Feature Clustering and Visualization of High Dimensional Data using Clique Cover Theory

Master Thesis
Media Informatics

Abhijeet Das
Matriculation number 374125

2019-10-09

Supervisors:
Prof. Dr. Christoph Quix
Prof. Dr. Stefan Decker

Advisors:
Arnab Chakrabarti
Dr. Michael Cochez

Eidesstattliche Versicherung

Name, Vorname

Matrikelnummer

Ich versichere hiermit an Eides Statt, dass ich die vorliegende Arbeit/Bachelorarbeit/ Masterarbeit* mit dem Titel

selbständig und ohne unzulässige fremde Hilfe erbracht habe. Ich habe keine anderen als die angegebenen Quellen und Hilfsmittel benutzt. Für den Fall, dass die Arbeit zusätzlich auf einem Datenträger eingereicht wird, erkläre ich, dass die schriftliche und die elektronische Form vollständig übereinstimmen. Die Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.

Ort, Datum

Unterschrift

*Nichtzutreffendes bitte streichen

Belehrung:

§ 156 StGB: Falsche Versicherung an Eides Statt

Wer vor einer zur Abnahme einer Versicherung an Eides Statt zuständigen Behörde eine solche Versicherung falsch abgibt oder unter Berufung auf eine solche Versicherung falsch aussagt, wird mit Freiheitsstrafe bis zu drei Jahren oder mit Geldstrafe bestraft.

§ 161 StGB: Fahrlässiger Falscheid; fahrlässige falsche Versicherung an Eides Statt

(1) Wenn eine der in den §§ 154 bis 156 bezeichneten Handlungen aus Fahrlässigkeit begangen worden ist, so tritt Freiheitsstrafe bis zu einem Jahr oder Geldstrafe ein.

(2) Straflosigkeit tritt ein, wenn der Täter die falsche Angabe rechtzeitig berichtigt. Die Vorschriften des § 158 Abs. 2 und 3 gelten entsprechend.

Die vorstehende Belehrung habe ich zur Kenntnis genommen:

Ort, Datum

Unterschrift

Acknowledgements

First of all, I would like to express my sincere admiration and gratitude to Prof. Dr. Stefan Decker for giving me the opportunity to work on this thesis in his department. I would like to thank Prof. Dr. Christoph Quix for his constant assistance, encouragement, and feedback during the entire phase of the thesis. I am deeply grateful to Dr. Michael Cochez for his valuable feedback and suggestions. Lastly, I am very much obliged to Arnab Chakrabarti. Without his guidance, motivation, support, and optimism this thesis would not have been possible.

I would also like to acknowledge the IT Center and IT Team of Informatik 5 for providing the computational and storage resources required for this study.

Finally, I would like to thank my dear parents and my friends for their trust, patience, and constant support. Their motivation helped me to finish the thesis and my studies successfully.

Abstract

The exponential growth of science and technology has led to a tremendous increase in data volume. With the technological development raising its bar every day, the devices are so well equipped that it is now possible to capture many aspects or features of data at a relatively low cost. Many real-life applications involve the analysis of high-dimensional data. As the number of dimensions increases, the challenge to process and visualize such data also increases. A phenomenon known as "The Curse of Dimensionality" is frequently observed in high-dimensional data. The approaches to reducing the dimensions either attempt to transform the features or select a subset of features with the aim to minimize the information loss. The feature transformation methods transform the feature space to create a reduced set of principal components with the aim of having the same discriminatory power as in the original space. The feature selection methods aim to maximize the relevance and minimize the redundancy of the selected features. They are more interpretable because they select a subset of the original feature without any transformation. Many feature selection methods have shown good results when the class label information is present in the data. Researchers have focused their attention on unsupervised feature selection methods because the real-world data mostly doesn't contain the class labels. Moreover, this field is less explored compared to supervised learning. Recently, many graph-based approaches have developed a considerable interest in the field of feature clustering and selection because of the flexibility of applying many graph concepts to the data.

In this work, we present an approach to perform unsupervised feature clustering and selection. The approach also provides a platform to visualize the features and the data points in the reduced feature space using standard visualization methods like Parallel Coordinates, Scatterplot Matrix, etc. The proposed approach develops a novel graph clustering algorithm based on Clique Cover Theory to perform unsupervised feature clustering. The number of clusters and the size of each cluster is determined dynamically without the need to specify any parameters. The unsupervised feature selection is then performed by selecting the most central feature from each cluster. It is carried out by using a graph centrality measure known as Eigenvector Centrality. An interactive User Interface is designed to facilitate the user to select the dataset, explore the intermediate results (feature graphs and correlation matrix), and to visualize the data points efficiently.

Contents

List of Figures

List of Tables

List of Algorithms

Chapter 1

Introduction

1.1 Introduction

With the ever-expanding amount of available computing resources, the ability to collect and generate a wide variety of complex, high-dimensional datasets continues to grow. High-dimensional datasets can be seen in numerous fields of study, such as economy, biology, chemistry, political science, astronomy, physics, etc. Dimensionality refers to the number of attributes or features in the data. In the domain of data visualization, a dataset with 10 or more features can be called as "high-dimensional" because of the limitations of the visualization methods to efficiently illustrate all the features. Moreover, the data is mostly unsupervised when dealing with real-world problems. One of the most prominent mining techniques in unsupervised data is clustering. In the case of high-dimensional data, feature clustering can help in a better understanding of the data and also in reducing the data size for efficient storage and processing [8]. For example, it is used in several machine learning tasks like image segmentation, information retrieval, text classification [9, 10, 11]. Feature selection is another mining technique that focuses on finding the most meaningful dimensions, thereby removing irrelevant and redundant features from the data [4, 12]. The result of the feature selection is a reduced feature set that can be effectively interpreted and visualized. The visualization can be less cluttered and can provide intricate details of the data which can be used for analysis. In our thesis, we have performed unsupervised feature clustering by using a novel graph clustering approach based on the Clique Cover Theory. Considering this clustering result into account, we have further performed unsupervised feature selection using Eigenvector Centrality and finally visualize the selected features using different standard visualization methods.

Clustering is the task of grouping a set of objects in such a way that objects in the same group (called a cluster) are more similar to each other than to those in other groups [1]. In high-dimensional data, features can be interpreted as objects, and feature clustering aims to group similar features together. Recently graph-based methods have shown good results in many areas that require feature clustering such as text classification [13], face

[1] `https://datavizproject.com/data-type/cluster-analysis/`

recognition [14]. They have played an essential role due to their ability to encode similarity relationships among data using the concepts of graph theory. In our approach, feature clustering is performed using the concepts of the Clique Cover Theory. A brief explanation of the concept is given below.

Clique Cover Theory- It is a graph clustering approach based on the underlying notion of maximal cliques [15]. Clique, Q, is defined as a subset of vertices of an undirected graph G such that every two distinct vertices are adjacent [2]. A Clique, which is not a subset of a larger clique, is known as maximal clique Q_i [3]. In our approach, we have extended the concept of maximal cliques to the edge-weighted cliques. A maximal clique having the maximum sum of edge-weights highlights the notion of a cluster. The recursive process of determining such cliques leads to the generation of clusters of different sizes(more details in Chapter 3, Section 3.5.3). In terms of graph theory, a cluster can be termed as a cover on the respective nodes of the graph such that the subset of nodes is strongly connected within the cover. The graph clustering algorithm is applied to the feature graph, which is created with features as nodes and correlation coefficient as edge-weights. We have used the Maximal Information Coefficient (MIC) [3] to determine correlations between numerical features and the Chi-square Test of Association [16] followed by Cramer's V to determine the correlations between categorical features. A detailed explanation of these correlations is given in Chapter 3, Section 3.5.1.

Feature selection, also known as variable selection or attribute selection is the process of selecting a subset of relevant features from the original feature space [17]. The unsupervised feature selection methods have raised considerable interest in many research areas. This is mainly due to their ability to identify and select relevant features without requiring the class label [18]. In our thesis, we have performed unsupervised feature selection as a succeeding step after the feature clustering. The relevant feature, also termed as the Representative Feature, is selected from each cluster using the concepts of Eigenvector Centrality. It is used as a measure to determine the importance of a feature in a cluster [4]. A brief explanation of the concept of Eigenvector Centrality is given below.

Eigenvector Centrality- It is a global-based centrality measure that captures the importance of a node by considering global information from all the nodes in the cluster. In [19], the authors have stated that Eigenvector centrality gives better results when clusters are formed by the determination of maximal cliques. It is based on the principle that "A node is important if it is linked to by other important nodes" [5](more details in Chapter 3, Section 3.5.4). It is used to give relative scores to all the nodes in the cluster, and the node with the maximum score is treated as the Representative Feature node.

In this thesis, we have developed an algorithm for unsupervised feature clustering and selection. The algorithm constructs a feature graph using the features and the above-

[2] `https://courses.cs.washington.edu/courses/cse527/01au/oct25/oct25.html`
[3] `http://mathworld.wolfram.com/MaximalClique.html`
[4] `https://www.sci.unich.it/~francesc/teaching/network/eigenvector.html`
[5] `http://djjr-courses.wikidot.com/soc180:eigenvector-centrality`

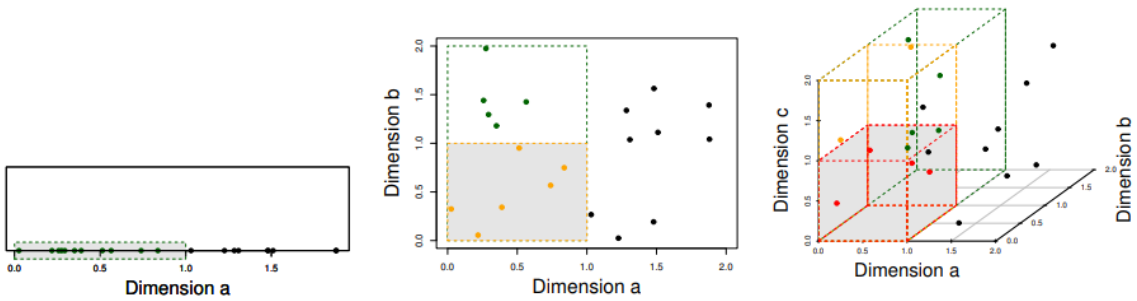
mentioned correlation measures. The graph clustering algorithm based on the Clique Cover Theory is applied to perform unsupervised feature clustering. The Representative Features are selected from each cluster using the Eigenvector Centrality. Finally, the data points corresponding to the selected features or the reduced feature space is visualized using different visualization methods such as parallel coordinates, scatterplot matrix, grouped bar charts, etc. An interactive User Interface is designed to explore the intermediate steps, visualize the feature graphs and their correlations and coherently visualize the data in the reduced dimensions.

1.2 Motivation

Many real-life applications involve the analysis of high dimensional data. As the number of dimensions increases, more and more scientific challenging problems appear. Moreover, the methods of visualization become limited, clumsy, and difficult to interpret. In this section, various factors that motivated to develop our approach are discussed. The challenges of the existing approaches and a relative comparison of our approach are also described.

1.2.1 The Curse of Dimensionality

As the datasets possess high dimensions, there emerges a phenomenon known as "The Curse of Dimensionality". It was Richard Bellman, who coined this phrase in his book on control theory [20]. In general terms, as dimensionality increases, the data points become increasingly sparse.



(a) 11 Objects in One Unit Bin (b) 6 Objects in One Unit Bin (c) 4 Objects in One Unit Bin

Figure 1.1: The Curse of Dimensionality [1]

As seen in Figure 1.1, the data points in only one dimension are relatively tightly packed. Adding a dimension stretches the points across that dimension, pushing them further apart. Additional dimensions make the data points sparse, and the notion of distance is no longer valid.

This calls for a technique known as "Dimensionality Reduction". It refers to the process of converting a set of data having high dimensions into lower dimensions, ensuring that

it conveys similar information concisely [21]. In the language of statistics and machine learning, it is the process of reducing the number of random variables by obtaining a set of principal variables [6]. The principal variables could be a new set of features or a subset of original features. Dimensionality reduction techniques can be broadly classified into Feature Transformation and Feature Selection Methods.

Feature transformation aims to find the principal variables of the dataset by creating linear or non-linear combinations of features in order to uncover the latent structure [22, 23, 24]. The intuition is that a high-dimensional dataset may exhibit interesting patterns on a low dimensional subspace because of the correlations among features. Principal Component Analysis (PCA) [22] and Linear Discriminant Analysis (LDA) [23] are prominent linear feature transformation methods, whereas Locally Linear Embedding (LLE) [24] is one such non-linear feature transformation method. Following are some of the limitations of these methods :

- The resultant features obtained from PCA are often less interpretable as it implements linear combinations of many features in high-dimensional data. It also suffers from information loss if the data points are depicted in the form of a circle(2D) or a hypersphere in high dimensions [25]. Moreover, it is unable to perform non-linear combinations of features.

- LDA requires labeled data, which makes it more situational. It fails to preserve the structure of the data if the distributions are non-Gaussian. It can give incorrect results when the discriminative information is not present in the mean of classes but in the variance [26].

- LLE is sensitive to noisy data. Moreover, it is known to have difficulty with a non-convex manifold structure [27].

Feature selection aims to find the principal variables by selecting a small subset of features in order to minimize redundancy and maximize relevance to the task [17]. It involves searching through different feature subsets and evaluating each of these subsets using certain criteria [28, 29]. With respect to the different selection strategies, feature selection can be broadly classified into 3 models: Wrapper, Filter, and Hybrid. Following are some of the limitations of each model:

- Filter model ranks each feature separately and thus does not take the feature dependencies into account. They are known to perform well concerning supervised data only where class labels are present.

- Wrapper model uses predictive models like clustering algorithms to score feature subsets. They can model feature dependencies, but they possess the risk of overfitting and high computational costs depending on the clustering algorithm.

[6] `https://urlzs.com/Tf1XS`

- The hybrid model combines the benefits from both filter and wrapper models, but they are also dependent on the computational performance of the clustering algorithm.

1.2.2 Challenges of Existing Approaches

In this section, the limitations of some of the existing methods related to unsupervised feature clustering and selection are described.

- In [30], the authors have introduced an algorithm for feature clustering based on the Expectation-Maximization (EM) clustering algorithm. The major drawback is that it has a slow convergence rate. It also tends to yield poor estimation to the asymptotic Maximum Likelihood Estimator (MLE).

- In [4], the authors have introduced a correlation measure known as the maximum information compression index to determine feature similarity. The correlation measure is known to capture only the linear relationship between features.

- In [31], the authors have used a graph clustering algorithm known as Louvain community detection to cluster the features. Also, they have used Pearson product-moment correlation to determine the edge-weights. The Louvain community detection algorithm depends on modularity optimization, and have trouble detecting small communities in large networks. On the other hand, the Pearson correlation is unable to capture non-linear relationships.

- In [32], the authors have used Mutual Information as a measure to find edge-weights between graph nodes. Although Mutual Information can capture non-linear relationships, it is sensitive to bin size.

- In [33], the authors have performed Minimum Spanning Tree (MST) construction by applying the Prims algorithm to identify the clusters. The complexity of MST construction increases when dealing with high-dimensional data. Moreover, it uses the uncertainty coefficient as a correlation measure, which is asymmetric in nature. The authors in [6] have used Spectral graph clustering to determine the clusters. The Spectral clustering requires a prior estimation of the number of clusters.

1.2.3 Comparison of Our Approach with the Current State-of-the-Art

In the previous section, various limitations of the existing approaches in processing high-dimensional data have been discussed. In this section, a relative comparison of our approach with regard to feature clustering, correlation, and selection are described.

- The Clique Cover approach determines the number of clusters and the size of each cluster dynamically based on the intrinsic properties of the graph, such as maximal cliques and edge-weights. Unlike Spectral Clustering, it doesn't require to estimate the number of clusters in advance.
- In our approach, the clusters are generated recursively until all the nodes are traversed. Therefore, unlike the Louvain community algorithm, it can determine small isolated clusters as well.
- The algorithm can be applied directly to the feature graph. Unlike MST, there is no need to transform the graph into a tree structure.

For creating the feature graph, we have used a couple of correlation measures; the Chi-square Test of Association and the Maximal Information Coefficient(MIC). The Chi-square tests are used to determine correlations in the categorical feature groups, whereas MIC is used in the numerical feature groups. Following are some of their properties:

- Unlike the Pearson correlation, MIC can capture a wide range of associations. It can identify linear, non-linear, functional as well as non-functional relationships between features.
- MIC gives similar scores to equally noisy relationships of different types without being biased and provides the score in a normalized range between 0 and 1. The coefficients resulting from MIC are more interpretable compared to the Mutual Information.
- Chi-square test of association is efficient while dealing with any type of categorical data; nominal, ordinal, or dichotomous.
- Unlike the uncertainty coefficient, the Chi-square test is symmetric in nature and is invariant with respect to the order of the categories in a categorical variable.

In our approach, we have not used the class label information to perform clustering and selection. So, it can work on both supervised and unsupervised datasets. It doesn't rank the features independently but models the dependency by clustering them. The selected features are the subset of the original features. Unlike PCA, it doesn't transform the features and thereby making them more interpretable. Besides providing clustering and feature selection results, the approach provides an interface to visualize the Representative Features. The interface allows the user to explore the intermediate results, visualize the feature graph, and visually inspect the results using standard techniques.

1.3 Use Case Scenario

To further clarify the motivation of developing such an approach, let's consider a real-world use case scenario. The scenario can help us to understand the problem definition and demonstrate how the results from our approach can be used to overcome the problem.

Let us start by assuming that a user, say; John wants to purchase a new automobile. He downloads an automobile dataset from a trusted source, like www.fueleconomy.gov. The dataset has 42 columns representing various features related to automobiles such as Mfr Name, Division, Carline, Annual Fuel Cost, City Fuel Consumption, etc. A screenshot of the dataset is shown in Figure 1.2.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
1	Mfr Name	Division	Carline	Verify Mfr Cd	Transmissic	Air Aspir M	Air Aspiration Method	Battery Type	Guzzler?	Index..Mc	Eng.Displ	X..Cyl	City.FE..G	Hwy.FE..G	Comb.FE..	X..Gears
2	Honda	Acura	NSX	HNX	Auto(AM-S	TC	Turbocharged			57	3.5	6	21	22	21	9
3	FCA US LLC	ALFA ROMEO	4C	CRX	Auto(AM6)	TC	Turbocharged			410	1.8	4	24	34	28	6
4	Volkswagen Grc	Audi	R8 AWD	VGA	Auto(AM-S	NA	Naturally Aspirated		G	65	5.2	10	14	22	17	7
5	Volkswagen Grc	Audi	R8 RWD	VGA	Auto(AM-S	NA	Naturally Aspirated		G	71	5.2	10	14	25	18	7
6	Volkswagen Grc	Audi	R8 Spyder AV	VGA	Auto(AM-S	NA	Naturally Aspirated		G	66	5.2	10	14	22	17	7
7	Volkswagen Grc	Audi	R8 Spyder RV	VGA	Auto(AM-S	NA	Naturally Aspirated		G	72	5.2	10	14	25	18	7
8	Volkswagen Grc	Audi	TT Roadster c	VGA	Auto(AM-S	TC	Turbocharged			46	2	4	23	30	26	6
9	BMW	BMW	M4 DTM Char	BMX	Auto(AM-S	TC	Turbocharged			488	3	6	17	24	20	7
10	Volkswagen Grc	Bugatti	Chiron	VGA	Auto(AM-S	TC	Turbocharged		G	38	8	16	9	14	11	7
11	General Motors	Chevrolet	CORVETTE	GMX	Auto(S8)	NA	Naturally Aspirated			278	6.2	8	15	25	18	8
12	General Motors	Chevrolet	CORVETTE	GMX	Auto(S8)	SC	Supercharged		G	223	6.2	8	13	23	16	8
13	General Motors	Chevrolet	CORVETTE	GMX	Manual(M7	SC	Supercharged			285	6.2	8	15	22	18	7
14	General Motors	Chevrolet	CORVETTE	GMX	Manual(M7	NA	Naturally Aspirated			276	6.2	8	16	25	19	7
15	Ferrari	Ferrari North A	488 gtb	FEX	Auto(AM7)	TC	Turbocharged			142	3.9	8	15	22	18	7
16	Ferrari	Ferrari North A	488 gtb	FEX	Auto(AM7)	TC	Turbocharged			143	3.9	8	16	22	18	7
17	Ferrari	Ferrari North A	488 Spider	FEX	Auto(AM7)	TC	Turbocharged			145	3.9	8	16	22	18	7
18	Ferrari	Ferrari North A	488 Spider	FEX	Auto(AM7)	TC	Turbocharged			144	3.9	8	15	22	18	7
19	Ferrari	Ferrari North A	812 Superfas	FEX	Auto(AM7)	NA	Naturally Aspirated		G	154	6.5	12	12	16	13	7
20	FCA US LLC	FIAT	124 Spider	CRX	Auto(A6)	TC	Turbocharged			405	1.4	4	25	36	29	6
21	FCA US LLC	FIAT	124 Spider	CRX	Manual(M6	TC	Turbocharged			406	1.4	4	26	35	30	6
22	Ford Motor Com	Ford	FORD GT	FMX	Auto(AM7)	TC	Turbocharged		G	322	3.5	6	11	18	14	7
23	Jaguar Land Rov	Jaguar	F-TYPE Conve	JLX	Auto(S8)	TC	Turbocharged			185	2	4	23	30	26	8
24	Jaguar Land Rov	Jaguar	F-TYPE Conve	JLX	Auto(S8)	SC	Supercharged			161	3	6	20	28	23	8
25	Jaguar Land Rov	Jaguar	F-TYPE Conve	JLX	Manual(M6	SC	Supercharged			171	3	6	16	24	19	6

Figure 1.2: The Automobile Dataset

John is not aware of all the features and their characteristics that are required to make a purchase decision. It can be time-consuming and baffling for John to understand all the features. On the other hand, it would be much simpler if he could filter out a few important features and make a decision based on that. In other words, a feature selection method that could provide a subset of features from the original feature set can make the dataset more comprehensible. It can also help John to identify relations between different automobile features.

Given the high-dimensional data, our approach performs a series of operations to clean and pre-process the data. The raw data available online may contain a lot of empty and duplicate values that are needed to pre-process first. In the automobile dataset, some of the

feature fields like "Fuel_cell_desc, Battery Type" doesn't contain any values. There are certain fields like "Guzzler, MaxBiodiesel" that are partially filled, whereas "Air Aspiration Method" is a duplicate field. During the pre-processing, the missing values are replaced with appropriate values, whereas the duplicate fields are removed. After cleaning and pre-processing, the dataset is segregated into categorical and numerical feature groups. In this case, there is a group of 11 categorical features and a group of 14 numerical features.

The feature groups are internally processed by applying various operations like feature correlations, feature graph construction, and clustering. The technical details of these methods are given in the Solution chapter Section 3.5. After processing, the approach gives the cluster results, as shown in Table 1.1 and 1.2.

Clusters	Node ID	Node Desc
Cluster 1	4	Verify Mfr Cd
	1	Mfr Name
	3	Division
	2	Carline
	10	Max Ethanol...Gasoline
Cluster 2	5	Transmission
	9	X..Gears
	7	Trans Desc
Cluster 3	8	X..Cyl
	11	X.1.Mfr.Smog.Rating
Cluster 4	6	Air Aspiration Method Desc

Table 1.1: Categorical Cluster Table

As seen, there are 4 categorical clusters and 5 numerical clusters. The number of clusters and the size of each cluster is determined automatically without providing any prior information. Moreover, all the nodes in the respective feature groups are part of the cluster, meaning it contains all the information from the original data. The clusters are used to identify the similarities between the features. For example, suppose John is aware that the price of the automobile increases with the number of gears. He can straightway apply the same analogy to Transmission and price relationship because Gears and Transmission are part of the same cluster. After providing the clustering results, the approach internally determines the centrality of each feature node in the cluster and provides a set of Representative Features from all the clusters. They are the most central and important feature that can describe the cluster in the most coherent way. The table of selected Representative Features is shown in Table 1.3 and 1.4.

Clusters	Node ID	Node Desc
Cluster 1	14	Comb Unadj FE- Conventional Fuel
	11	MFR.Calculated Gas Guzzler MPG
	9	Comb Co2 Rounded Adjusted
	5	Comb FE Guide- Conventional Fuel
Cluster 2	3	City FE Guide- Conventional Fuel
	7	City Co2 Rounded Adjusted
	12	City Unadj FE- Conventional Fuel
	6	Annual Fuel Cost
Cluster 3	4	Hwy FE Guide- Conventional Fuel
	8	Hwy Co2 Rounded Adjusted
	13	Hwy Unadj FE- Conventional Fuel
Cluster 4	2	Eng Displ
	10	Carline Class
Cluster 5	1	Index Model Type

Table 1.2: Numerical Cluster Table

Clusters	Node ID	Node Desc
Cluster 1	3	Carline
Cluster 2	5	Transmission
Cluster 3	11	X.1.Mfr.Smog.Rating
Cluster 4	6	Air Aspiration Method Desc

Table 1.3: Categorical Representative Features

As seen, there are 4 categorical Representative Features and 5 numerical Representative Features. It implies that we have reduced the total feature set from 42 to 9, considering only significant features that can provide a better interpretation of the data. It would be much simpler and effective for John to go through the selected 9 features in order to make a decision. Besides giving the textual output, our approach also provides an interface to visualize the data points corresponding to the selected features using standard visualization techniques. The relationships between the features can also be examined using the feature graphs. Visualization is effective in understanding the pattern of the data and explore relationships. An example of visualization using a parallel coordinate is shown in Figure 1.3.

Clusters	Node ID	Node Desc
Cluster 1	14	Comb Unadj FE- Conventional Fuel
Cluster 2	3	City FE Guide- Conventional Fuel
Cluster 3	4	Hwy FE Guide- Conventional Fuel
Cluster 4	10	Carline Class
Cluster 5	1	Index Model Type

Table 1.4: Numerical Representative Features

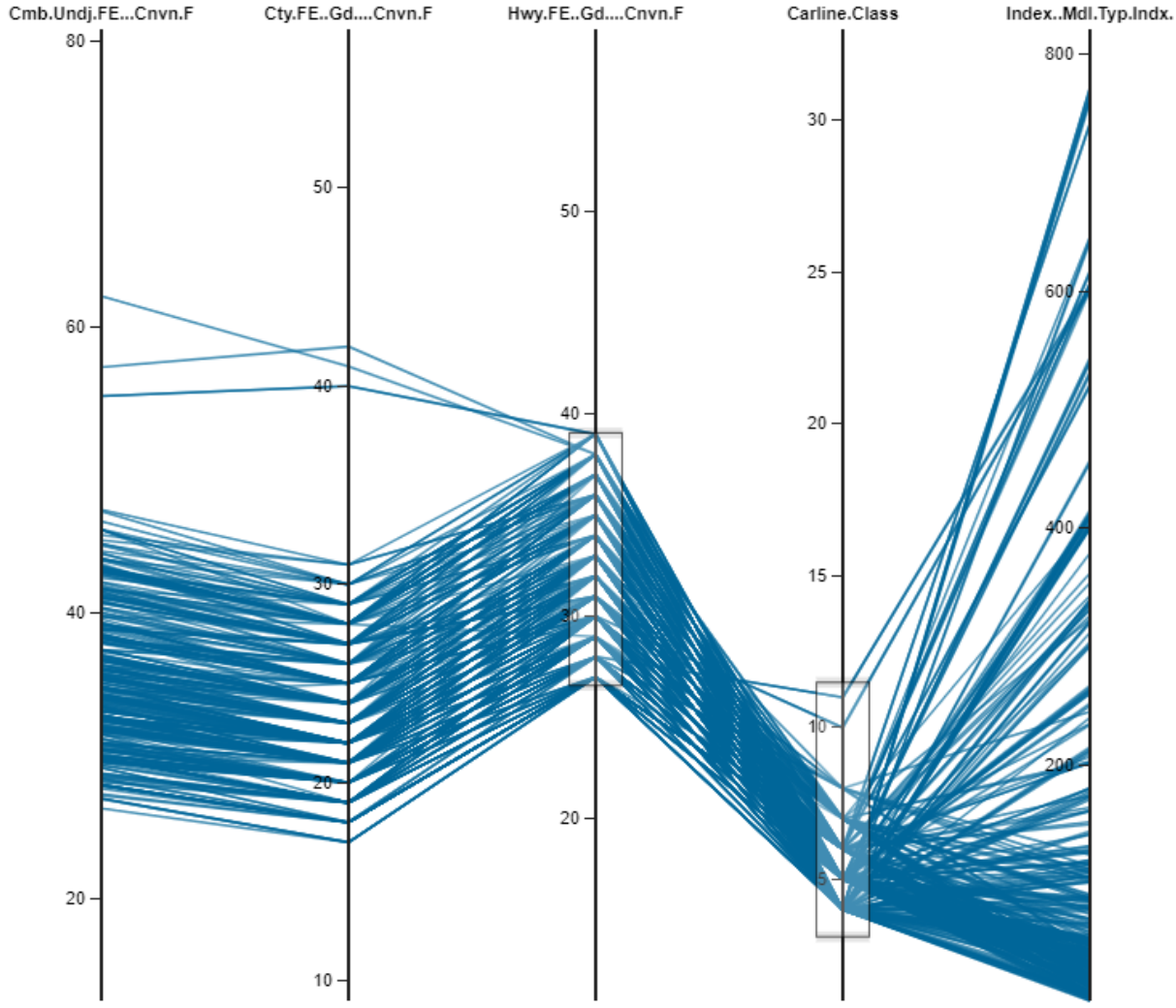


Figure 1.3: Parallel Coordinate Plot

1.4 Goal of the Thesis

In this section, the major goals of the thesis to develop the proposed approach are described.

1.4.1 Define Data Model

The data model consists of separate sets of categorical and numerical feature groups. After the data is cleaned and pre-processed, it is segregated to create the data model. The reason is that we want to capture the maximum trends of association in the respective feature groups. The creation of the data model is the first step before processing the features.

1.4.2 Determine Feature Clustering

Once the data model is created, feature clustering is performed on each feature group. This step involves several sub-tasks:

- Determining the correlation between feature pairs- This step embarks the generation of the feature graph. A couple of correlation measures, namely the Maximal Information Coefficient and Chi-square test of association are applied to the numerical and categorical groups, respectively. The output of this step is the square weighted adjacency matrix that contains the corresponding correlation coefficient between the feature pairs. It can be interpreted as a complete feature graph, where every node is connected to every other node.

- Identifying threshold coefficient - This step is used to determine the strong connections for every node in the complete feature graph. The concepts of the K-Nearest Neighbour (KNN) is used to determine the threshold connections for each node. This step prunes out weak relations from the complete feature graph and creates a feature dependency graph consisting of only strong connections.

- Applying the Clique Cover clustering algorithm- Once the feature dependency graph is generated, a graph clustering algorithm based on the Clique Cover Theory is applied. This step transforms the complete feature set into clusters of feature subset of varying sizes.

1.4.3 Select the Representative Feature

This step selects a Representative Feature from each cluster obtained from the previous step. A graph centrality measure known as Eigenvector Centrality is applied to determine the most central feature from each cluster. In other words, one feature per cluster is identified that can represent the cluster in the most coherent way.

1.4.4 Visualize the Selected Features

The final part of our approach is visualization. It is characterized by 2 parts. In the first part, the features are visualized using feature graphs like cluster feature graphs, representative feature graphs. The correlation between features is explored using correlation heatmaps. In the second part, the data points in the reduced feature space are visualized using standard methods. The reduced dimensional space provides many visualization techniques to demonstrate various characteristics of the data. Visualization can be very effective in understanding the intricate details of the data and can, therefore, help the user to make an efficient and prompt decision. Parallel Coordinate plot, Scatterplot Matrix, 3D Scatter plot, Grouped, and Stacked Bar Charts are some of the visualization methods used by our approach. Various interactive modes like brushing, tooltips, and highlighting are used to provide better visual cognizance.

1.4.5 Evaluate the Proposed Approach

For the evaluation, we have selected 9 datasets from different categories to compare our results with the existing approaches. The datasets are chosen with varying numbers of features and properties that enable us to test our approach in different circumstances. The evaluation using the classification accuracy is performed using different classifiers. The clustering accuracy metric is used for assessing the clustering quality. Representation Entropy is used as a metric to evaluate the redundancy rate of the selected features. The class separability and the cost-sensitive analysis is performed by plotting and comparing ROC curves. The Friedman test is performed to compare different results on all the selected datasets statistically. The computational complexity of different steps in the algorithm is determined. The qualitative evaluation is carried out by plotting Parallel Coordinate Plot and comparing the clutter observed in reduced feature space with respect to the original feature space.

Chapter 2

Related Work

In this chapter, some of the preliminary works in the field of feature clustering, dimensionality reduction, and unsupervised feature selection methods are discussed. An account of work related to feature correlation measures and visualization methods for high-dimensional data are also described.

2.1 Related Work on Feature Clustering

Feature clustering [1] seeks to find clusters of features in different subspaces within a dataset. These clustering algorithms localize the search for relevant dimensions or features and thus attempts to find clusters that exist in multiple, possibly overlapping subspaces. There are two major branches of subspace clustering based on their search strategy, namely the Top-down approach and the Bottom-up approach. The bottom-up algorithms find dense regions in low dimensional spaces and combine them to form clusters. The top-down algorithms start with the full set of dimensions as the initial cluster and evaluate the subspaces of each cluster, iteratively improving the results.

2.1.1 Bottom-up Algorithms

The bottom-up algorithm uses the grid-based clustering approaches [34] to determine clusters. The grid-based approach (Figure 2.1) involves the following steps:

1. Divide the feature space into (hyper) rectangular cells- It works by partitioning the range of values in each dimension into equally sized cells.

2. Discard low-density grid cells- A density-based definition of clusters is assumed, i.e., that high-density regions represent clusters, while low-density regions represent noise.

3. Combine adjacent high-density cells to form clusters.

The bottom-up method uses the downward closure property of density using the APRIORI algorithm to reduce the search space. The downward closure property of density means

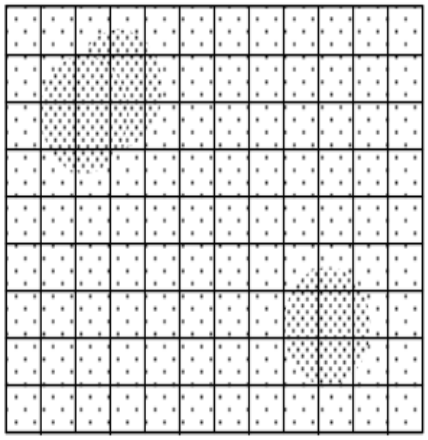


Figure 2.1: 2-dimensional Grid for Cluster Detection [1]

that if there are dense units in K dimensions, then there are dense units in all (K-1) dimensional projections [1]. The bottom-up methods are categorized as the way in which the grids are defined. The first group uses a static sized grid to divide each dimension into bins. The second group uses a dynamic or adaptive grid size.

Clustering In QUEst (CLIQUE) [35] is one of the first algorithms that use static grid size to determine the clusters within the subspace of the dataset. It combines density and grid-based clustering and uses an APRIORI style technique to identify the clusters in the subspaces. It creates the notion of the *coverage*. The coverage is defined as the fraction of the dataset covered by the dense units in the subspace. The subspaces with the maximum coverage are kept, and the rest are pruned. The clusters are formed by combining these dense units. A CLIQUE can find clusters of any shape and can also result in overlapping clusters. On the contrary, tuning the parameters for grid size and the density threshold are difficult as both could significantly affect the quality of the clustering. Moreover, it does not scale well with the number of dimensions.

Merging Adaptive Finite Intervals Algorithm (MAFIA) [36] is the extension of CLIQUE that uses an adaptive grid based on the distribution of data. In this algorithm, the dimension is partitioned based on the data distribution, and the resulting cell boundaries capture the cluster perimeter more accurately than fixed-sized grid cells. After the bins are created, it uses a similar APRIORI style algorithm to generate the clusterable subspaces. It requires giving the parameters like density threshold value and a threshold for merging adjacent windows. Although MAFIA works faster than CLIQUE and scales linearly with the number of instances, its running time grows exponentially with the number of dimensions in the clusters.

2.1.2 Top-down Algorithms

The top-down subspace algorithms start by finding an initial approximation of the clusters in the full feature space. It assigns the weight for each dimension and then iteratively

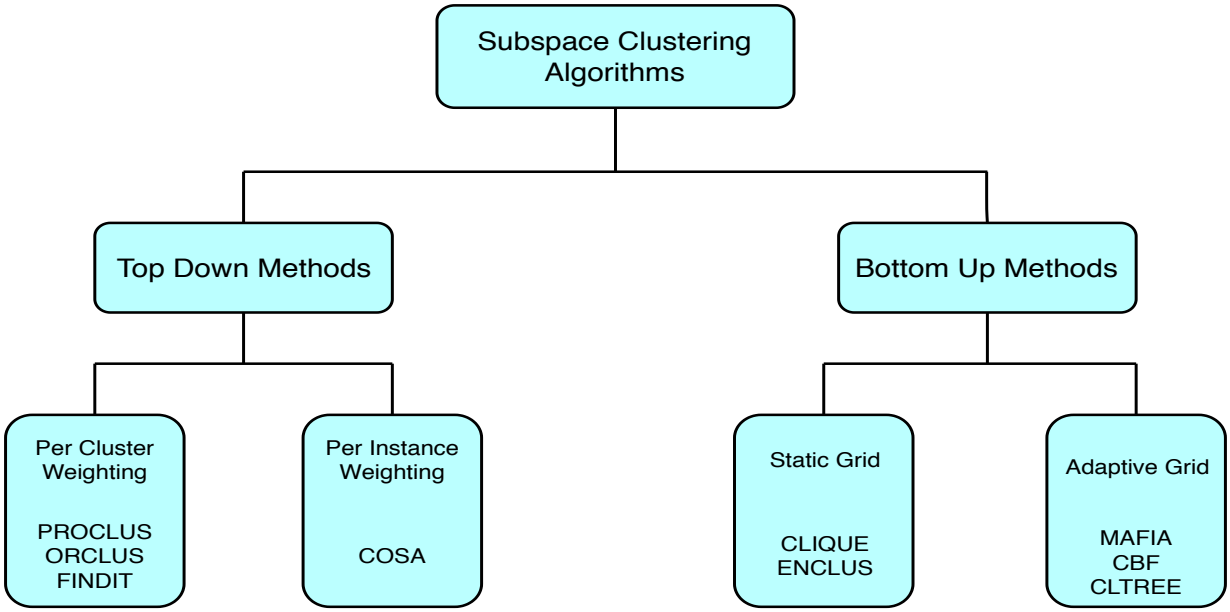


Figure 2.2: Subspace Clustering Methods [1]

updates the weights by running a clustering algorithm. The approach requires multiple iterations of expensive clustering algorithms in the full set of dimensions. The algorithms also use the sampling technique to improve performance.

PROjected CLUStering (PROCLUS) [37] is the first top-down subspace clustering algorithm. It samples the data, selects a set of K medoids, and iteratively improves the clustering. The algorithm proceeds as a three-phase approach, initialization, iteration, and cluster refinement. The initialization phase selects a set of potential medoids using a greedy approach that is far apart from each other. The iteration phase selects a random set of K medoids from this reduced dataset. It determines the cluster quality and replaces the bad medoids with newly selected ones. Cluster quality is based on the average distance between instances and the nearest medoid. The refinement phase reassigns points to the new medoids based on the clusters formed. PROCLUS depends on the input of critical parameters like the number of clusters and the size of the subspaces, which are difficult to determine ahead of time.

Clustering On Subsets of Attributes (COSA) [38] is another top-down iterative algorithm that assigns weights to each dimension for each instance. It starts by examining the K-Nearest Neighbors (KNN) of each instance, and these neighborhoods are further used to determine the respective dimension weights for each instance. The weights for pairs of instances are calculated that are, in turn, used to update the distances used in the KNN calculation. The process is repeated until the weights stabilize. Finally, the neighborhoods for each instance become increasingly enriched with instances belonging to its own cluster. It doesn't need to specify the number of dimensions in clusters; however, the value of K needs to be adjusted, which is used in the KNN calculations.

2.2 Related Work on Dimensionality Reduction

As discussed in the Motivation Section 1.2, a phenomenon is known as "The Curse of Dimensionality" is observed in high-dimensional data. It means that as the number of dimensions increases in a dataset, the points spread out far apart, and the notion of distance is no longer very meaningful. It implies that the distance-based measures like K-means clustering may not give an appropriate result. To cope with this, many dimensionality reduction measures are developed for a long time. These techniques can be broadly classified into two categories: Feature Transformation and Feature Selection Methods.

2.2.1 Feature Transformation

Feature Transformation methods summarize the dataset by creating linear or non-linear combinations of features with the aim to uncover latent structure [22, 23, 24]. They are often used as a pre-processing step before applying a predictive model or a clustering algorithm. These methods generate a new set of reduced features or principal variables.

Principal Component Analysis (PCA) [22] is one of the prominent linear feature transformation methods. PCA has the notion of finding the eigenvectors of a covariance matrix with the highest eigenvalues and uses those to project the data into lower dimensions. The transformation is defined in such a way that the first principal component has the largest possible variance. The succeeding component has the highest variance possible under the constraint that it is orthogonal to the preceding components (Figure 2.3). It is often used as a tool in exploratory data analysis and used for making predictive models. It is also used to visualize genetic distance and correlations between populations.

Linear Discriminant Analysis (LDA) [23] is also a linear transformation method, but it aims to maximize the separation between multiple classes. The direction of maximum variance, as used in the PCA, is not a good estimation for classification. LDA is a "supervised" method that explicitly attempts to model the difference between the classes of data by finding the best direction to separate classes. It projects the dataset onto a lower-dimensional space with good class-separability in order to avoid overfitting.

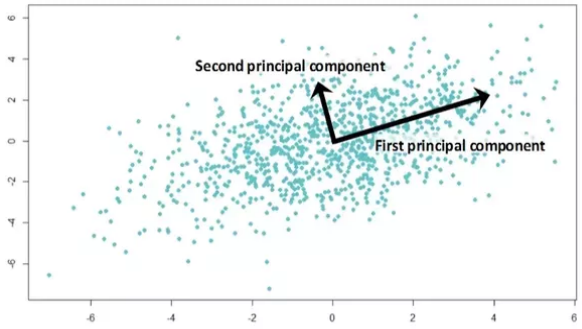


Figure 2.3: Principal Components in PCA

Locally Linear Embedding (LLE) [24] is a non-linear dimensionality reduction technique. It works by embedding the objects in a low dimension while preserving their neighborhoods. It first finds a set of the nearest neighbors of each point. Then it computes a set of weights for each point in such a way that the point can be best described as a linear combination of its neighbors. Finally, it finds the low-dimensional embedding of points, such that each point is still described with the same linear combination of its neighbors. LLE can project non-linear trends to lower dimensions but tends to give poor results when handling non-convex manifold structure.

2.2.2 Feature Selection

Feature Selection attempts to discover the features of a dataset that are most relevant to the data mining task at hand [39]. It is a commonly used technique for reducing the dimensionality of a problem to more manageable levels. Rather than creating new features by combination, feature selection aims to select a subset of features from the original feature space making it more interpretable. It involves searching through various feature subsets and evaluating each of these subsets using some criterion [17, 28, 29]. Depending on selection strategies, it can be broadly classified into 3 models: wrapper, filter, and hybrid.

- Filter methods select the most relevant features from the data itself without using any clustering algorithm. The features are evaluated based on the intrinsic properties of the data. The main properties of filter methods are their speed and scalability. However, they are unable to model feature dependencies and yield better results mostly with supervised data. Relief [40], Laplacian Score [5], Spectral feature selection [6] are some of the filter methods.

- Wrapper methods use the results of a specific clustering algorithm to evaluate feature subsets. They are characterized by finding features subsets that contribute to improving the quality of the results in the clustering algorithm. Although it can model feature dependencies, the main disadvantage of wrapper methods is that they have a high computational cost, and are dependent on a particular clustering algorithm. Feature Subset Selection using Expectation Maximization (FSSEM) [30] is an example of wrapper method.

- Hybrid methods exploit the qualities of both approaches, filters, and wrappers. It attempts to achieve a compromise between efficiency (computational effort) and effectiveness. Boosting Based Hybrid Feature Selection (BBHFS) [41] is a hybrid based approach.

2.3 Related Work on Feature Correlations

The determination of the correlation between the features is one of the initial steps in clustering. Correlation is a statistical measure that indicates the extent to which two or more variables are related to each other [1]. Feature correlation measures are used to determine the extent of feature similarity between features. Recently many papers have focused on determining similarity based on correlation measures [42]. This focus is because the distance measures like Euclidean [2] can only measure the space distance between data points, so they are unable to reveal the underlying dependency.

In [33], the authors have used a well-known measure of the correlation between two random variables known as the Pearson correlation coefficient [43].

$$\rho = \frac{\text{cov}(X, Y)}{\sigma_x \sigma_y}$$

It is defined as the covariance of the two variables divided by the product of their standard deviations. It measures the linear correlation between two random variables. It can take the value ranging from -1 to +1, implying negative and positive correlations, respectively. The major limitation is that it cannot interpret non-linear correlations.

In [44], the authors have used the Spearman correlation coefficient [45] to measure the relationship between two variables. Spearman's rho can be interpreted as a rank-based version of Pearson's correlation coefficient, which can be used for variables that are not normal-distributed. It can identify both the linear as well as monotonic relations.

$$\rho = 1 - \frac{6 \sum d_i^2}{n(n^2 - 1)}$$

where d is the pairwise distances of the ranks of the variables x_i and y_i, and n is the number of samples.

The linear or monotonic relationships approximate only a fraction of the true relationship types. The authors in [32] have circumvented this problem by using Mutual Informataion [46], which is capable of identifying non-linear relationships. It is a measure of the mutual dependence between the two variables. It quantifies the amount of information obtained about one random variable by observing the other random variable. The concept of Mutual Information is linked to the entropy of a random variable [47]. Entropy measures the amount of uncertainty of an unknown or random quantity [3]. Based on that, the Mutual Information between two variables is the reduction in uncertainty in one variable given another variable. Formally, it can be defined as:

[1] `https://whatis.techtarget.com/definition/correlation`
[2] `https://www.definitions.net/definition/euclidean+distance`
[3] `https://towardsdatascience.com/the-intuition-behind-shannons-entropy-e74820fe9800`

$$I(X,Y) = \sum_{x,y} P(x,y) \ln \frac{P(x,y)}{P(x)P(y)}$$

where P(x), P(y) are the individual probabilities, and P(x,y) is the joint probability.

Although Mutual Information can capture non-linear trends, it is known to be computationally expensive and sensitive to bin size. In [48], the authors have used both Mutual Information and Hoeffding inequality [49] as a constraint in determining feature correlations. Hoeffding's inequality provides an upper bound on the probability that the sum of bounded independent random variables deviates from its expected value by more than a certain amount [4].

The notion of correlation is termed as an association in the case of categorical variables. As discussed, Spearman correlation can also be used when there are ordinal variables, i.e., the variables that can be ranked. Like Spearman's, Goodness and Kruskal's Gamma test [50] is also a non-parametric measure that is recommended when the data has many tied ranks. Gamma tests for an association between points and also tells us the strength of association. It is also particularly useful when the data has outliers, as they don't affect the results much. Both Spearman and Gamma test is used only in ordinal variables; they cannot be used in the case of nominal variables. The uncertainty coefficient [51] is a measure of the nominal association. It is based on information theory. In [33], the authors have used the uncertainty coefficient to determine feature correlations. The uncertainty coefficient, also called the entropy coefficient, is a measure of the entropy in a column variable Y, which a row variable X explains. It is given by the formula.

$$U(X|Y) = \frac{H(X) - H(X|Y)}{H(X)}$$

where, H(X) is the entropy of a single distribution and the relative entropy of X given Y, H(X|Y). Although the uncertainty coefficient works for both ordinal and nominal data; it is non-symmetric and gives a different output when the order of variables is changed.

2.4 Related Work on Unsupervised Feature Selection Methods

Since our approach deals with the unsupervised data, some of the related works in the field of unsupervised feature selection are discussed in this section.

2.4.1 Unsupervised Feature Selection using Feature Similarity

In [4], the authors have described an unsupervised feature selection algorithm suitable for large datasets. A new feature similarity measure, known as the "Maximum Information

[4] https://medium.com/@ODSC/understanding-the-hoeffding-inequality-a4bb801a05a7

Compression Index", is introduced. It is based on the principles of the Pearson product-moment correlation and proposed an extension to capture more trends. The Pearson correlation is given by :

$$\rho = \frac{\text{cov}(X, Y)}{\sigma_x \sigma_y}$$

where cov() denotes the covariance and $\sigma()$ denotes the standard deviation.
The Maximum Information Compression Index λ_2 is given by the formula :

$$2\lambda_2 = \sqrt{Var(x) + Var(y) - 4Var(x)Var(y)(1 - \rho(x, y)^2)}$$

The value of λ_2 is zero when the features are linearly dependent and increases as the amount of dependency decreases. The λ_2 is the amount of the reconstruction error committed if the data is projected to a reduced dimension in the best possible way. It is a measure of the minimum amount of information loss or the maximum amount of information compression possible.

The feature sets are partitioned based on the K-Nearest Neighbor principle using the above similarity measure. The k-nearest features of each feature are computed. Among them, the most compact subset is selected. The process is repeated for the remaining features until all the features are either selected or discarded. In the first iteration, a constant threshold ϵ is assigned, which is equal to the distance of the kth nearest neighbor of the feature selected. In the subsequent iterations, the value of λ_2 is checked corresponding to ϵ, and the value of K is adjusted accordingly. Appendix A.1, Algorithm 6 gives the algorithm for Unsupervised Feature Selection using Feature Similarity.

2.4.2 Laplacian Score for Feature Selection

In [5], the authors have used the filter-based approach of unsupervised feature selection by ranking the importance of the features in a dataset. For each feature, the Laplacian score is calculated to reflect its locality, preserving power. The score is based on the observation that two data points are probably related to the same topic if they are close to each other [52]. The nearest neighbor graph is constructed in order to model the local geometric structure. The Laplacian Score (LS) is fundamentally based on two concepts; Laplacian Eigenmaps [53] and Locality Preserving Projection [54]. The importance of a feature can be considered as the degree to which it respects the graph structure. A "good" feature should be the one on which two data points are close if and only if there is an edge between these two points.

A reasonable criterion for choosing a good feature is to minimize the following objective function:

$$L_r = \frac{\sum_{ij}(f_{ri} - f_{rj})^2 S_{ij}}{Var(f_r)}$$

Where Var(f_r) is the estimated variance of the r-th feature. For a feature to be considered as a good feature, the S_{ij}, the $f_{ri}f_{rj}$, and thus the Laplacian Score tends to be small. The step-wise description of the algorithm for selecting features based on the Laplacian score is given in Appendix A.1, Algorithm 7.

2.4.3 Spectral Feature Selection for Supervised and Unsupervised Learning

In [6], the authors have performed an unsupervised feature selection by exploiting the intrinsic properties of the graph based on Spectral Theory. A similarity measure known as the RBF Kernel function is used:

$$S_{ij} = e^{-\frac{|x_i - x_j|^2}{(2\delta)^2}}$$

Where δ is a free parameter set according to the number of clusters involved. Given G, its adjacency matrix W is defined as W(i, j) = w_{ij}. Also, the degree matrix D of the graph G is defined by: D(i, j) = d_i, if i = j, and 0 otherwise. Here d_i can be interpreted as an estimation of the density around x_i, because the more data points that are close to x_i, the larger the d_i. Given the adjacency matrix W and the degree matrix D of G, the Laplacian matrix L and the normalized Laplacian matrix L are defined as: $L = D - W$ and $\mathcal{L} = D^{-0.5}LD^{0.5}$.

Spectral Feature Selection According to graph theory, the structural information of a graph can be obtained from its spectrum. Spectral feature selection studies to select features according to the structure of the graph induced from S. A feature that is consistent with the graph structure assigns similar values to instances that are near each other on the graph. It ranks the features of the graph obtained from S.

There are 2 main function used to calculate the rank of the feature vectors. Let $\tilde{f}_i = (D^{\frac{1}{2}}f_i)$ denote the weighted feature vector of F_i and $\hat{f}_i = \frac{\tilde{f}_i}{|\ f_i|}$ denote the normalized weighted feature vector. The score of F_i can be evaluated by the following function:

$$\varphi_1(F_i) = \hat{f}_i^{\ T}\mathcal{L}\hat{f}_i$$

The value of $\varphi_1(F_i)$ can be small, if $\hat{f}_i$ is very similar with eigenvector ξ_0. In this case, a small $\varphi_1(F_i)$ value does not indicate better separability. To handle this, another ranking function is used :

$$\varphi_2(F_i) = \frac{\hat{f}_i^T \mathcal{L} \hat{f}_i}{1 - \hat{f}_i^T \xi_0}$$

According to spectral clustering theory, the leading K eigenvectors of $\mathcal{L}$ form the optimal soft cluster indicators that separate G into K parts. Therefore, if K is known, we can also use the following function for ranking: $\varphi_3(F_i) = \sum_{j=1}^{k-1}(2 - \lambda_j)\alpha_j^2$

where λ_j are the Eigenvalues and $\sum_{j=0}^{n-1} \alpha_j^2 = 1$.

Based on the above-ranking functions, the spectral feature selection algorithm is given in Appendix A.1, Algorithm 8.

2.4.4 Unsupervised Feature Selection for Multi-Cluster Data

Traditional methods score each feature independently and neglect the possible correlation between different features and thus can not produce an optimal feature subset. To cope with such issues, the authors in [7] paper proposed a new approach, called Multi-Cluster Feature Selection (MCFS), for unsupervised feature selection. The features are selected such that the multi-cluster structure of the data can be best preserved.

Since naturally occurring data usually have multiple clusters structure, a good feature selection algorithm should consider the following two aspects:

1. The selected features should best preserve the cluster structure of the data.
2. The selected features should "cover" all the possible clusters in the data.

The flat embedding for the data points which "unfold" the data manifold can be found by solving the following generalized eigenproblem $Ly = \lambda Dy$, where Y = $[y_1...y_K]$, y_k's are the eigenvectors of the above generalized eigen-problem with respect to the smallest eigenvalue. L and D are the Laplacian and Degree matrix. Given y_k, a column of Y, the relevant subset of features can be determined by minimizing the fitting error as follows:

$$\min_{a_k} \left| y_k - X^T a_k \right|^2 + \beta \left| a_k \right|$$

where a_k is a M-dimensional vector and $|\ a_k| = \sum_{j=1}^{M} |\ a_{k,j}|$ denotes the L1-norm of a_k.

The above regression problem has the following equivalent formulation:

$$\min_{a_k} \left| y_k - X^T a_k \right|^2$$

$$|a_k| \leq \gamma$$

The Least Angel Regression (LAR) algorithm [55] can be used to solve the above optimization problem. Instead of setting the parameter γ, LARs provides another choice

to control the sparseness of a_k by specifying the cardinality (the number of non-zero entries) of a_k, which is particularly convenient for feature selection. For every feature j, we define the MCFS score for the feature as :

$$MCFS(j) = max_k \, |a_{k,j}|$$

The algorithm to perform unsupervised feature selection using multi-cluster data is given in Appendix A.1, Algorithm 9.

2.5 Related Work on High-dimensional Data Visualization

One of the major challenges of high-dimensional data is to visualize them. As the dimensions grow, it becomes challenging to visualize them using traditional visualization techniques. The major hindrances are the physical limitations of the display screen(2D/3D) and the relatively small capacity to process complex data. In the past decade, several approaches [2] are introduced to visually convey high-dimensional structural information such as parallel coordinates [56], quality measures [57]. The process of visualization can be divided into three steps: Data transformation, visual mapping, and view transformation. The Visualization Flow diagram is shown in Figure 2.4.

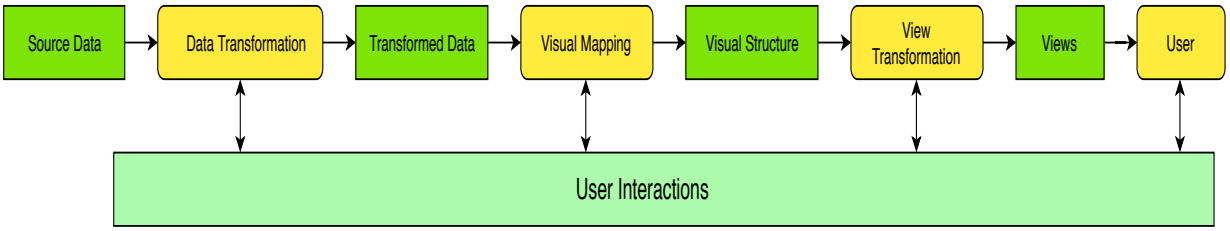


Figure 2.4: Visualization Flow Diagram [2].The yellow boxes represent different steps of the visualization process. The results of the individual steps are given in green.

Data transformation mainly consists of methods, namely feature clustering and dimensionality reduction. Some of the works related to these fields are presented in sections 2.1 and 2.2. The visualization methods are incorporated in the data transformation methods. For example, the interactive PCA (iPCA) [58] introduces a system that visualizes the results of PCA using multiple coordinated views. In [59], the authors have introduced a dual visual analysis model where both the dimension embedding and point embedding can be explored simultaneously.

Visual mapping converts the analysis result obtained from the data transformation or the original dataset into visual structures based on various visual encodings. There are various ways of visual mapping.

1. **Axis-Based Methods**- These methods refer to visual mappings where element relationships are expressed through axes representing the dimensions. Some mappings in this category are:

    - Scatterplot Matrix- A scatterplot matrix is a collection of bivariate scatterplots in a matrix format that allows users to view multiple bivariate relationships simultaneously. One of the major drawbacks of the Scatterplot Matrix is the scalability. The number of bivariate scatterplots increases quadratically as the dimensionality of the dataset increases. Scagnostics [60] is a set of measures designed for identifying interesting plots and is also extended to handle time-series data [61]. In [62], the authors have introduced an interactive feature selection method by evaluating the maximum conditional entropy of all possible axis-parallel scatterplots.
    - Parallel Coordinate Plot- Parallel Coordinate Plot (PCP) [63] allows patterns that highlight multivariate relations to be revealed by showing all the axes at once. The major drawback is that as the number of points increases, the line density in the PCP increases dramatically, which can lead to overplotting and visual cluttering. The authors in [64] were able to reduce clutter for Parallel Coordinate Plot without altering the information content simply by reordering the dimensions and adding a brushing axis feature.
    - Radial Layout- Radviz [65] uses a circular pattern. In Radviz, n-dimensional anchors are placed along the perimeter of a circle. Each circle represents one of the dimensions of an n-dimensional dataset.

2. **Glyphs**- Chernoff faces [66] are one of the first attempts to map a high-dimensional data point into a single glyph. Recently, the authors have utilized glyphs to provide statistical and sensitivity information to present trends in the data.

3. **Pixel-Oriented Approaches**- Several works have targeted dense pixel displays in order to encode maximum information. The pixel concept can be applied to bar charts to create pixel bar charts [67].

4. **Animation**- Recently, many techniques have used animated transitions to enhance the perception of point and structure correspondences among multiple relevant plots.

View transformation is the final step that decides what is to be appeared on-screen. The 3 main categories of rendering process are Illustrative Rendering, Visual Representation, and Color Blending.

User interaction is integrated with each of the processing steps. The amount of user interaction is categorized into 3 parts: Computation-Centric, Interactive Exploration, and Model Manipulation.

Chapter 3

Solution

As seen in the previous sections, there are many challenges in processing and visualizing high-dimensional data. The feature selection methods mostly perform well in the presence of the class labels. The wrapper model of unsupervised feature selection depends on the clustering algorithms that may have high complexity. On the other hand, the filter models fail to undertake the feature dependency into account. The graph-based feature selection methods rely on the proper tuning of the parameters to give accurate results. Many correlation measures are incapable of capturing different trends of association, thereby generating misleading edge-weights in the graph. Moreover, the methods to visualize become limited and yield clumsy results as the number of dimensions increases.

Fulfilling these limitations, it is tempting to develop an approach that aims to perform feature selection in unsupervised high-dimensional data. In this chapter, we have described various steps to create the data model, determine feature correlations and cluster features using Clique Cover Theory. Considering the feature clusters into account, we have determined EigenVector Centrality scores to select Representative Features. The chapter also describes various standard visualization methods to visualize data points in the reduced feature space.

3.1 Selection of Dataset

In our approach, the unsupervised high-dimensional datasets are used. If the dataset has class labels, we have removed them first and then used it in our approach in an unsupervised way. We have chosen datasets from different categories in order to carry out the feature clustering and selection. The selected datasets have an increasing number of features. Such datasets enable us to perform the stress test of our approach. The number of categorical and numerical features also varies in the selected datasets. Table 3.1 shows the name and description of the selected datasets.

Data Sources List		
Data Description	**Dimensions**	**Data Source**
Heart Dataset	13	kaggle
Australian Credit Dataset	14	openml
Hepatitis Dataset	20	kaggle
Automobile Dataset	25	UCI Machine Learning Lab
Breast Cancer Dataset	30	UCI Machine Learning Lab
Auto Univ Dataset	39	openml
QSAR Biodegradation Dataset	41	openml
Right Heart Catheterization Dataset	54	bigml
Sonar Dataset	60	kaggle
Emotions Dataset	78	openml
Robot Failure Dataset	91	openml
Yeast Dataset	116	UCI Machine Learning Lab
Spectrometer Dataset	125	openl
Musk Dataset	168	openml
Arrhythmia Dataset	280	kaggle
Airline Ticket Price Dataset	417	openml

Table 3.1: Selected Datasets

3.2 Define Data Model

The data model consists of separate sets of categorical and numerical feature groups. The reason to segregate is that we wanted to apply the most suitable sets of correlation measures that can capture maximum trends of association in the respective groups. As discussed in Sections 1.2.2 and 2.3, there are many drawbacks of specific correlation measures. Also, a single correlation measure cannot work well with both the categorical and numerical groups. The selection of appropriate measures is required for the proper construction of the feature graph, which we will see in the next sections. We have defined a set of constraints for all the selected datasets in order to clean and segregate the data into categorical and numerical feature groups. For example, a constraint is given to remove features like "ID" or "Name", which is unique for all the data points. Another constraint is to set a number of factors to segregate the feature groups(details in Section 3.4.4 and 4.1.4).

For example, consider the automobile data described in the Use Case scenario(Chapter 1, Section 1.3) describing various features of a car like model name, brand, engine size, displacement, horsepower, carline, fuel tank capacity, etc. In this case, our data model is a collection of numerical feature groups that is "engine size, displacement, horsepower, fuel capacity" and a separate collection of categorical feature groups like "model name, brand, carline."

3.3 The System Architecture

The system architecture is laid into 3 phases (Figure 3.1) :

1. Ingestion phase - The first phase is to ingest and parse the dataset. The initial operations on the data like cleaning, pre-processing, imputation, and segregation are carried out in this phase. It creates the data model, i.e., a set of processed categorical and numerical feature groups that is required in the subsequent phases.

2. Processing phase - This is the central and most important phase that takes the input from the ingestion phase and output the reduced feature set. It contains all the functions and algorithms required to process the data model and produces the result for visualization. The two prominent operations carried out in this phase are Feature Clustering and Representative Feature Selection.

3. Visualization phase - This phase is divided into 2 parts. The data points in the reduced feature set are visualized using various standard visualization methods. The features and their correlations are visualized using feature graphs like cluster feature graphs, representative feature graphs, and correlation heatmap. The visualizations provide interactive modes that enable the users to zoom, pan, brush or rotate the graphs. It creates a clear impression of a multi-dimensional graph in a coherent way.

3.4 Ingestion Phase

This phase ingests the dataset, parse it, and creates the data model as needed to process in the next phases. The dataset can be in CSV or JSON format. For describing the ingestion steps, it is assumed to be in the form of a spreadsheet or table in which the rows represent the data points, and the columns represent the features. The pictorial flow representation of the steps involved in this phase is shown in Figure 3.2.

3.4.1 Data Cleaning

Data Cleaning is the critical first step in any data science project. In our approach, the following steps have been used to clean the data.

1. Removing all NA or blank columns.

2. Removing all columns with more than 40% missing values [1].

3. Removing the columns that contain unique values for every row. For example, a column like “Id” or “URL”, which is unique for every row, is insignificant because it doesn’t possess any correlation with respect to other features and therefore can not contribute to feature clustering and selection.

[1] `https://www.dataquest.io/blog/machine-learning-preparing-data`

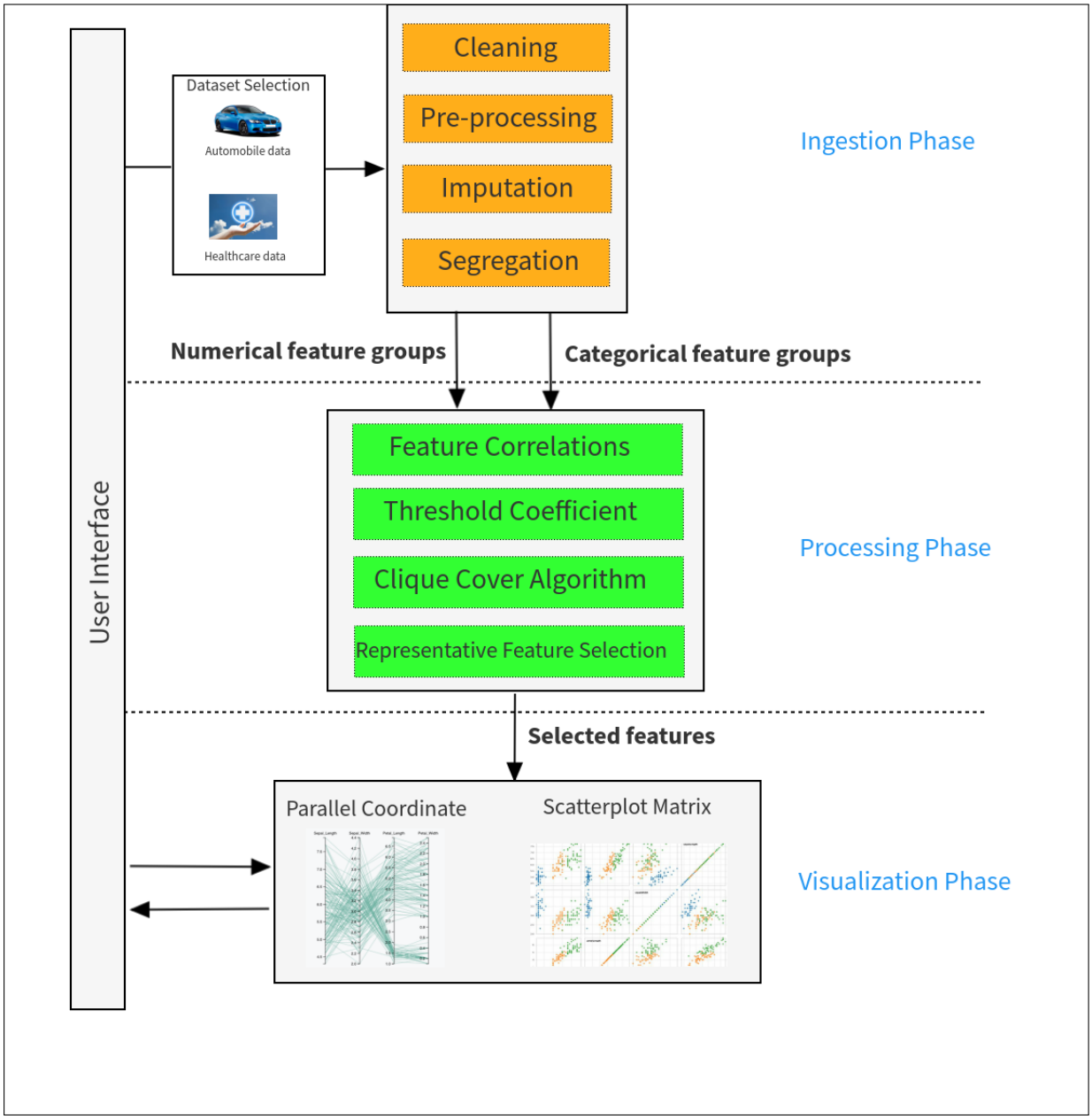


Figure 3.1: The System Architecture

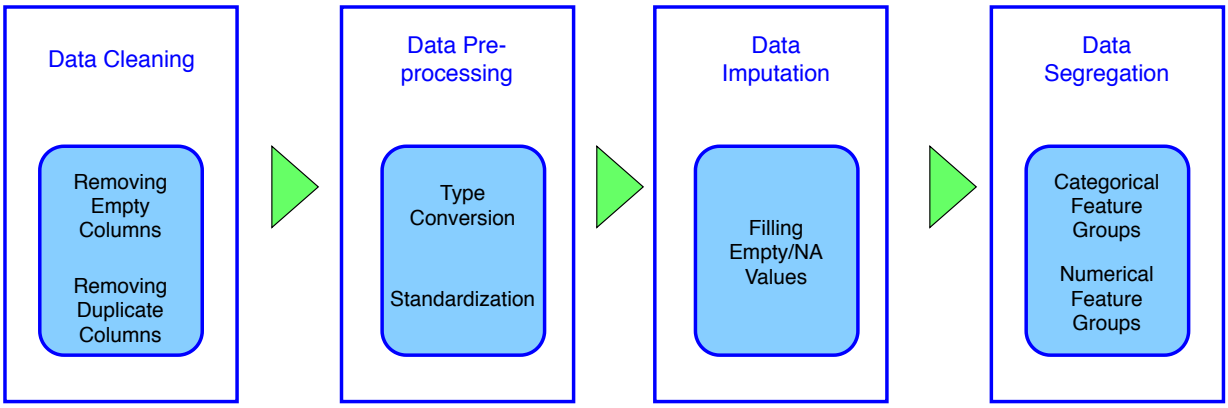


Figure 3.2: The Steps in Ingestion Phase

4. Removing the columns that contain non-interpretable feature values. For example, a column containing special symbols or non-ASCII characters has to be removed.

5. Removing duplicate columns.

3.4.2 Data Pre-processing

Data Pre-processing is a mining technique that involves transforming raw data into an understandable format as needed to create the data model [2]. In our approach, the following steps have been used to pre-process the data.

1. Type conversion- This step ensures that the numbers are stored as numerical data types, and the categorical variables are stored as string data types. For example, the numerical feature, say weight(Kg), is given as "1595". Due to the double quotes, this value is interpreted as a string. This step converts such values into an interpretable numeric format.

2. Rectifying syntax errors- It involves several steps like removing extra spaces at the beginning of the string and fixing typos or structural errors [3]

3. Standardization- It involves standardizing both numerical and categorical data. For strings, it ensures all values are either in the lower or upper case. For numbers, it ensures standardizing the numerical values like converting scientific notation to interpretable decimal format.

[2] `https://urlzs.com/d5BPo`
[3] `https://urlzs.com/r9qo1`

3.4.3 Data Imputation

Data imputation is the process of replacing missing data with substituted values [4]. There are various methods of data imputation. The most basic method is imputation with mean/-median. It works by calculating the mean/median of the existing values in a column and then replacing the missing values by the generated mean/median independently. It has the following disadvantages: it can only be used with numeric data, and it does not model correlations between features. The imputation by mean/median also gives less realistic and inaccurate results. Some other imputation methods are Multivariate Imputation by Chained Equation (MICE) or Imputation using Deep Learning. Although these methods are more accurate, they are computationally expensive.

In our solution, we have used a method known as "Fast imputation of missing values" [68]. It performs efficient data imputation by using a chained random forest method [69, 70]. It works on the principle of predictive mean matching that tries to raise the variance in the resulting conditional distributions to a realistic level and performs multiple imputations as an iterative chaining process. This method can be used to impute the missing values to both categorical and numerical data types in a computationally efficient manner.

3.4.4 Data Segregation

This is the final step of the ingestion phase. After the data is cleaned, pre-processed, and imputed, the segregation of data is carried out. It is done by considering the data type of the values present in the columns. For example, the string data types like "Model Name" or "Car Name" come in the categorical feature group, whereas the numerical data types like "Mileage" or "Fuel Tank Capacity" come in the numerical feature group. The categorical variables are also distinguished by having a certain number of factors or levels. Factors are the variables that take on a limited number of different values; such variables are often referred to as categorical variables [5]. Based on the dataset values, a constraint on the number of factors is applied to carry out segregation. Figure 3.3 shows different types of data after segregation.

3.5 Processing Phase

This is the central and most important phase consisting of functions and algorithms that perform the unsupervised feature clustering and selection. It involves the steps to create the feature graph, apply the graph clustering algorithm, and perform feature selection. The process flow diagram of the steps is shown in Figure 3.4.

[4] `https://www.theprojectdefinition.com/data-imputation/`
[5] `https://www.guru99.com/r-factor-categorical-continuous.html`

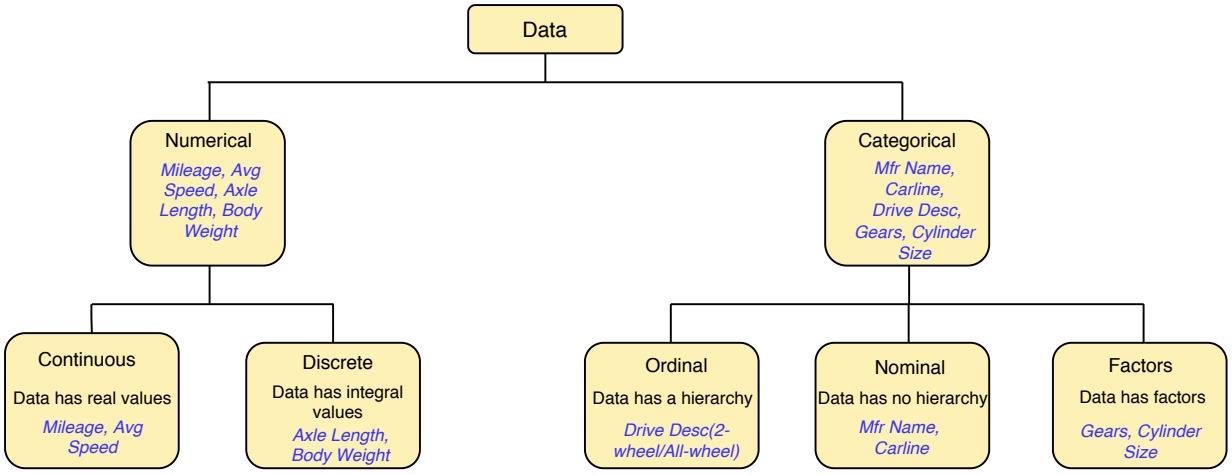


Figure 3.3: The Types of Data

3.5.1 Feature Correlation

To construct the feature graph, we need to determine the feature correlations first. The feature graph is the graphical representation of the data where the nodes are the features, and the edge-weights are the correlation or association coefficients between the features. From the ingestion phase, we have the data model as the categorical and numerical feature groups. The features in the respective group become the nodes of the feature graph, and the correlation coefficient becomes the edge-weight between the nodes. In our approach, a couple of correlation measures are used:

1. Chi-square test of association followed by Cramer's V for categorical feature groups.

2. Maximal Information Coefficient (MIC) for numerical feature groups.

Chi-square Test and Cramer's V Chi-square test [16] is used to determine relation or association between variables at the categorical level, i.e., at nominal or ordinal levels. It can be used at both univariate and bivariate levels. In its univariate form, it is known as "Chi-square goodness of fit test" [71]. For example, if we wanted to explore whether the frequency of weddings across months of the year are evenly distributed, the univariate form can be used. The bivariate form analyses two variables in conjunction with one another. It is known as the "Chi-square test of association" [71]. For example, it can be used to examine whether gender is associated with the fear of crime. In our approach, we have used the "Chi-square test of association" to find the correlation between categorical variables.

The Chi-squared test of association, also written as χ^2, is a statistical hypothesis test that finds the association between categorical variables. It determines whether a set of observed frequencies deviate significantly from a set of expected frequencies [6]. It works on

[6] `https://www.statisticshowto.datasciencecentral.com/probability-and-statistics/chi-square/`

the contingency table. A contingency table (also known as a cross-tabulation or crosstab) is a type of table in a matrix format that displays the (multivariate) frequency distribution of the variables. It provides a basic picture of the interrelation between two variables and can help find interactions between them. It is often used in business intelligence, engineering, and scientific research. A sample contingency table is shown below in Figure 3.5.

The formula for calculating Chi-square test of association is given as:

$$\chi^2 = \frac{1}{d} \sum_{k=1}^{n} \frac{(O_k - E_k)^2}{E_k}$$

where: O_k is the Observed frequency, E_k is the Expected frequency and d is the number of rows in the contingency table.

The expected frequency can be calculated by multiplying row total and column total divided by the grand total. For example, referring to the Figure 3.5, the expected frequency with respect to the first block (where gender is Female, and Sport preference is Archery) is ((100 * 45)/200).

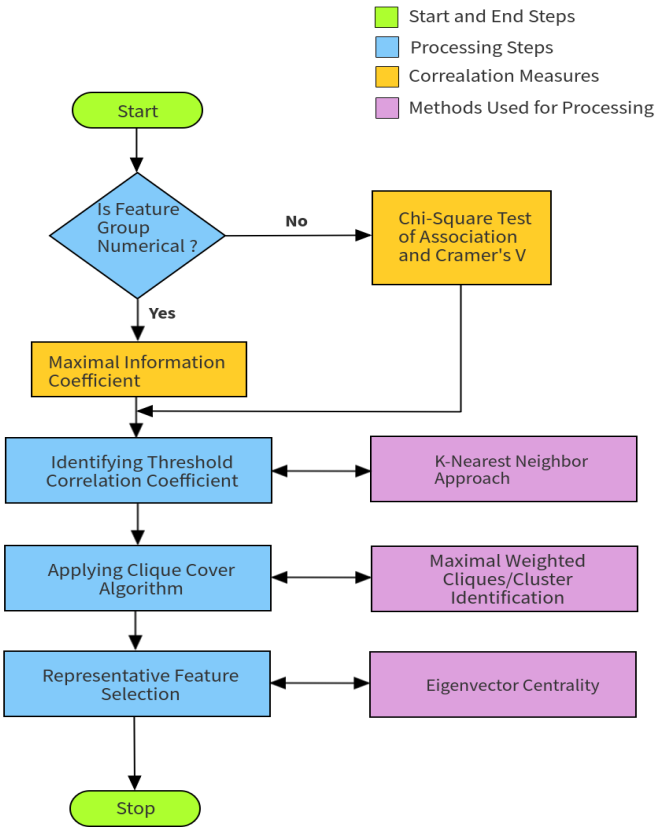


Figure 3.4: The Process Workflow

		Sport Preference			
		Archery	Boxing	Cycling	
Gender	Female	35	15	50	100
	Male	10	30	60	100
		45	45	110	200

Figure 3.5: The Contingency Table [7]

While Chi-square is advantageous in the sense that it is symmetric in nature and invariant with respect to the order of the categories, it suffers from a few weaknesses. It tells nothing about the strength of the association between the variables; it just signifies that there is one. Another drawback is that it is sensitive to sample size.

In order to solve these problems, some post-hoc tests are needed like the Phi coefficient [8] and Cramer's V [9]. While both of them are immune to sample size, the Phi coefficient is limited to the contingency table of size 2, whereas Cramer's V can be used on the table of any size. Moreover, Cramer's V provides a normalized value between 0 and 1: where 0 implies little association, and 1 implies a strong association between the variables.

Cramer's V is computed by taking the square root of the chi-squared statistic divided by the sample size and the minimum dimension minus 1.

$$V = \sqrt{\frac{\chi^2/n}{min(k-1, r-1)}}$$

where χ^2 is the Chi-square test, n is the grand total of observations, k being the number of columns and r being the number of rows.

In our approach, the Cramer's V coefficient is determined between every categorical variable pair and stored in a matrix format. The entities in the matrix correspond to the edge-weights in the categorical feature graph.

[8] http://www.pmean.com/definitions/phi.htm
[9] http://changingminds.org/explanations/research/analysis/cramers_v.htm

Maximal Information Coefficient (MIC)- MIC [3] is used to find correlations between features in the numerical feature groups. Unlike Pearson and Spearman's coefficient that can capture only linear and monotonic relations, MIC can capture a wide range of associations like linear, non-linear, functional, and non-functional. MIC belongs to a larger class of Maximal Information-based Non-parametric Exploration (MINE) [10]. MINE statistics can be used not only to detect interesting associations but also to identify the properties such as non-linearity and monotonicity.

Two significant aspects of MIC are generality and equitability. Generality implies that given sufficient sample size, the statistic can capture a wide range of interesting associations, not limited to specific function types (such as linear, exponential, or periodic). Equitability means that the statistic should give similar scores to equally noisy relationships of different types. For example, the statistic should not give a higher coefficient to a noisy linear relationship compared to strong sinusoidal relationships. An equitable statistic should give similar scores to functional relationships. Figure 3.6 shows the equitability results of different relationships.

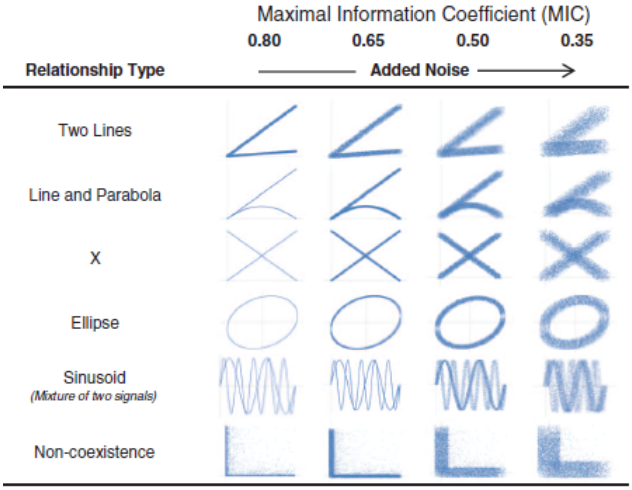


Figure 3.6: MIC Equitability Results [3]

MIC is based on the concepts of Mutual Information. The Mutual Information is a measure of the mutual dependence between the two variables and is based on the concepts of Information Entropy [11].It can capture non-linear trends, but it is computationally expensive to determine Mutual Information in high-dimensional data [72]. To overcome this, MIC calculates the normalized Mutual Information obtained from a grid of points in the data.

[10] http://www.exploredata.net/
[11] http://www.scholarpedia.org/article/Mutual_information

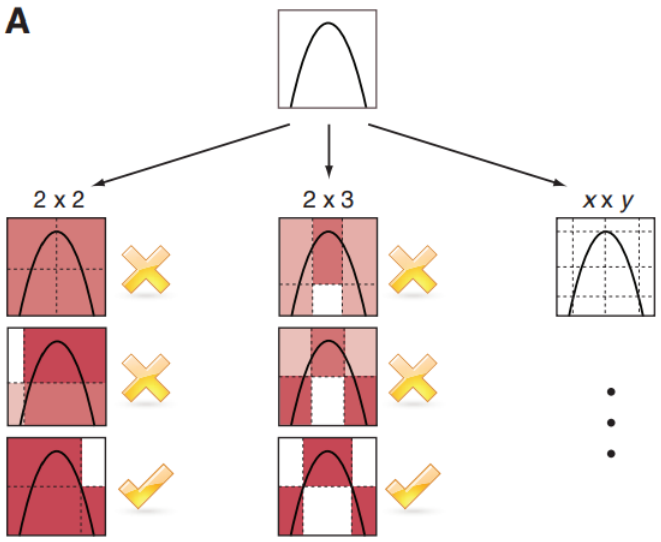


Figure 3.7: MIC Grid Resolution [3]

Intuitively, MIC is based on the idea that if a relationship exists between two variables, then a grid can be drawn on the scatterplot that partitions the data to encapsulate that relationship. Consider Figure 3.7; the plot shows a non-linear relationship between 2 variables. In the case of the 2x2 grid, the bottom grid structure captures the maximum grid resolution because it covers the entire non-linear relationship with the least possible area. Similarly, in the 2x3 grid structure, the last one has the maximum grid resolution. The grid resolution is determined by the mutual information present in that grid. Therefore, the first step to calculate MIC is to explore all grids, computing for every pair of integers (x,y), and to determine the maximum grid resolution or the largest possible mutual information obtained in the x-by-y grid.

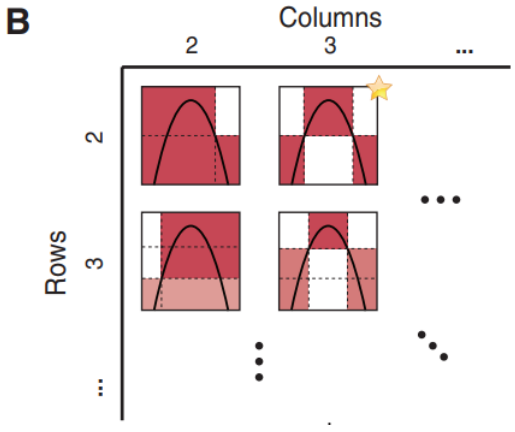


Figure 3.8: MIC Characteristic Matrix [3]

$$I^*(D, x, y) = maxI(D|_G)$$

where I*(X|Y) represents Mutual Information taken maximum overall grids G with x columns and y rows.

The next step is to normalize the Mutual Information values to ensure a fair comparison between grids of different dimensions. A characteristic matrix (Figure 3.8) is defined containing the highest normalized mutual information achieved by any x-by-y grid.

$$M(D)_{x,y} = \frac{I^*(D, x, y)}{\log min(x, y)}$$

Finally, MIC is defined to be the maximum value present in the characteristic matrix.

$$MIC(D) = max_{x,y} M(D)_{x,y}$$

In our approach, MIC is used as a measure to determine the correlation between the numerical feature groups. A square matrix is generated to contain the correlation coefficients between the feature pairs.

This ends the first step of the processing phase. The square weighted matrix can be interpreted as the complete graph. A complete graph is an undirected graph in which every pair of distinct vertices is connected by a unique edge [12]. By the end of this step, we obtain a couple of complete feature graphs with nodes and edges corresponding to the features and correlations in the respective feature groups(details and feature graphs are given in Chapter 4, Section 4.2.1).

3.5.2 Identifying Threshold Coefficient

A graph cluster possesses the property that the internal nodes are strongly connected with each other. The complete feature graph obtained from the previous step may contain a few weakly connected edges between the nodes. In this step, we have determined a threshold correlation coefficient for each feature node and prune out the weak edges (edges below the threshold). The threshold is determined using the concepts of K-Nearest Neighbors [13]. The value of K is determined based on the number of features in the complete feature graph. The K strongest connections for each feature are retained, and the rest are pruned.

K-NN approach is commonly used in graph algorithms to determine the proximity of the nodes [73]. It is often used in search applications where we are looking for similar items. For example, searching for semantically similar documents, also referred to as Concept Search. It is also extensively used in Recommender systems. The reason for using the K-NN algorithm is that it doesn't require any assumptions or training steps to build the model. Moreover, it is easy to implement and robust to noisy data as well. K-NN algorithm has been used in spectral clustering to determine the clusters of nodes present in

[12] `http://mathworld.wolfram.com/CompleteGraph.html`
[13] `https://urlzs.com/J4F1y`

the graph [74]. The authors have used Euclidean distance as a measure to determine edge-weights. There are various methods to determine the value of K for the K-NN algorithm. In our approach, the value of K is determined using the standard rule of thumb formula given as the square root of the number of features [14]. After determining the value of K, the weak edges are pruned by replacing their values with zeros in the weighted adjacency matrix. The resultant feature graph, which we called the feature dependency graph, is represented in the form of an affinity matrix (details in Section 4.2.2). An affinity matrix is a matrix representation of a graph containing all the nodes, but edges correspond only to the strong associations. It is also symmetric in nature. The feature dependency graph obtained in this step is used as a base to perform the graph clustering algorithm.

3.5.3 Clique Cover Algorithm

In this step, a graph clustering approach is known as "Clique Cover Theory" is applied to the feature dependency graph to cluster the features. It is based on the concept of maximal cliques. The cluster identification using a maximal clique is a part of the family of dominant set clustering [15], which is used in the domain of computer vision and bioinformatics.

The dominant set theory has intriguing links to game theory, graph theory, and optimization theory. From the game-theoretic perspective, clusters are regarded as equilibria of non-cooperative clustering games. From the graph-theoretic context, it generalizes the notion of the maximal clique to edge-weighted graphs. From an optimization point of view, it can be characterized in terms of solutions to a simplex-constrained, quadratic optimization problem.

In our approach, the graph-theoretic aspect of the dominant sets is used that is based on the concepts of maximal cliques. A brief definition of clique Q and maximal clique Q_i are given in the Introduction Chapter (section 1.1). Given a graph G, a subgraph H of a graph G is a maximal clique if H is isomorphic to a complete graph, and there is no vertex v ∈ V(G) such that v is adjacent to each vertex of H [75]. In other words, a subgraph H of a graph G is a maximal clique if H is a clique, and there is no vertex in G that sends an edge to every vertex of H.

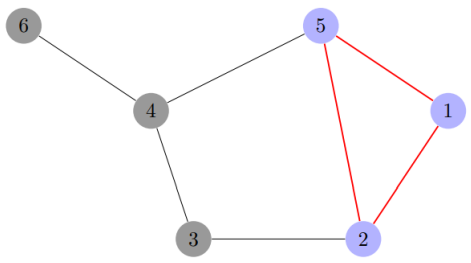


Figure 3.9: Graph with Clique and Maximal Clique

[14] https://saravananthirumuruganathan.wordpress.com/2010/05/17/a-detailed-introduction-to-k-nearest-neighbor-knn-algorithm/

To understand the concept of clique and maximal cliques, let us consider the Figure 3.9. The following are the cliques in the graph: {4,6}, {1,2}, {1,5}, {2,3}, {2,5}, {3,4}, {4,5}, {4,6} and {1,2,5}. Every pair of vertices are connected with each other in all of these cliques. The maximal cliques in this graph are {4,6}, {2,3}, {3,4}, {4,5}, {4,6} and {1,2,5}, as all other cliques can be extended by a vertex to form {1,2,5}.

In our approach, the concept of maximal cliques is extended to edge-weighted graphs because the feature dependency graph is an undirected and edge-weighted graph. The maximal cliques with respect to the maximum sum of the edge-weights are identified from the feature dependency graph. Such a subgraph satisfies 2 properties:

1. Internal homogeneity- Elements belonging to a group have high associations with each other.

2. Maximality- A maximal clique cannot be further extended by introducing external elements.

These properties emphasize the notion of a cluster. The cluster can be termed as Clique Cover [15] in graph theory. A Clique Cover means partitioning an undirected graph into cliques of various sizes. Using dynamic programming, the feature clusters are determined recursively from the feature dependency graph.

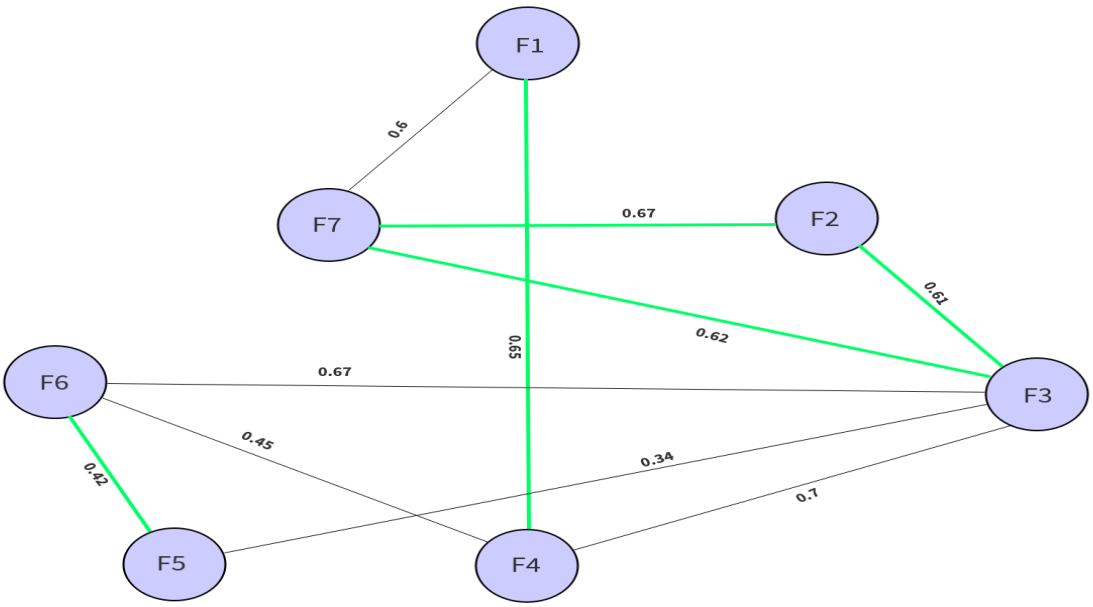


Figure 3.10: Sample Feature Dependency Graph

[15] https://pdfs.semanticscholar.org/e94e/f10d4889ace7b0b17eb9fffad4cf7547eefb.pdf

Let's consider the feature dependency graph shown in Figure 3.10 to understand the clustering approach based on the Clique Cover Theory. Here we have 7 features F1 to F7 representing the nodes or features of the graph, and the edge-weight corresponds to the correlation coefficient between the feature pairs.

The algorithm initially determines the cliques from the graph and further determines the maximal cliques. It then proceeds to incorporate the edge-weights of the maximal cliques. The maximal clique with respect to the maximum sum of edge-weight is identified as the cluster. From the graph depicted in Figure 3.10, we can see that there exists many cliques such as {3,6}, {1,7},{5,6},{3,5,6} etc. But there 5 maximal cliques namely, {3,4,6} ,{1,4}, {2,3,7}, {3,5,6} and {1,7}. The weight of a maximal clique is the sum of the weight of the edges in that clique. So the corresponding weight of the 5 maximal cliques is 1.82, 0.65, 1.90, 1.42, and 0.60, respectively. In this case, {2,3,7} is the maximal clique with respect to the maximum weight of 1.90. This clique satisfies "internal homogeneity" and "maximality" properties because it has strong interconnections and is maximal in nature. This can be termed as the first cluster or the Clique Cover. The algorithm removes the clustered nodes and edges from the existing graph by dynamically truncating the affinity matrix and updating the dimensions. This implies that the new feature dependency graph contains the remaining 4 nodes with features F1, F4, F5, F6 respectively. The recursive process of cluster identification is applied internally, and it outputs 2 more clusters {1,4} and {5,6} from the recursively generated feature dependency graphs. Therefore, the Clique Cover graph clustering algorithm generates 3 clusters {2,3,7}, {1,4} and {5,6} of sizes 3, 2 and 2 respectively.

As seen, the Clique Cover produces the non-overlapping/exclusive clusters, which is evident from the result that none of the features are present in more than one cluster. Moreover, the approach doesn't require a prior estimation of the number of clusters. The number of clusters and the size of each cluster is determined dynamically from the intrinsic properties of the graph. We have recursively applied the Clique Cover Theory on the feature dependency graph to generate feature clusters of different sizes.

The algorithm, "Unsupervised feature clustering using Clique Cover Theory", requires the weighted adjacency matrix obtained from the Feature Correlation step as the input argument. It initializes two output lists; one for storing the cluster node IDs and the other for storing the cluster node labels. Initially, the remaining dimensions are set to the total dimensions of the feature graph. The algorithm proceeds by identifying the threshold coefficient for each node and generate a feature dependency graph. Next, the algorithm determines the cliques, maximal cliques, and the total number of maximal cliques in the feature dependency graph. It then iterates through all of the maximal cliques and identifies the maximal clique having the maximum weight by summing up the edge-weights in the maximal clique. This maximal clique having maximum weight is the first cluster or Clique Cover. It updates the output list with the corresponding node Ids and labels of the first cluster. The algorithm then removes the clustered nodes and edges and updates the feature graph with the remaining dimensions. It recursively calls the *FeatureClustering* procedure to generate more clusters. This way, the algorithm recursively reduces the size of the remaining dimensions and assigns the feature nodes as part of some clusters. The

terminating condition for the recursive process is reached when there is a single node. It terminates by updating the output list with node Ids and labels of the last node. The different steps of the algorithm are shown in Algorithm 1.

Algorithm 1: Unsupervised Feature Clustering using Clique Cover Theory

Procedure : *FeatureClustering*;
Input : Weighted Adjacency Matrix obtained from the Feature Correlation step.
Output : ClusterNodeIds and ClusterNodeLabels.
Initialize: ClusterNodeIds ← (); ClusterNodeLabels ← (); remainDim ← totalDim;
Step 1: Identify threshold coefficient.
Step 2: Generate a feature dependency graph.
Step 3: Determine Cliques Q, Maximal Cliques Q_i and number of Maximal Cliques.
Step 4: Determine the weight of all the Maximal Cliques.
Step 5: Determine the Maximal Clique with maximum weight and set it as the first cluster or the Clique Cover.
Step 6: Update the output with the cluster node Ids and labels.
Step 7: Remove the clustered nodes and edges from the feature graph.
Step 8: Update the feature graph with remaining dimensions.
Step 9: Recursive call to *FeatureClustering* procedure.
Step 10: If there is one feature node present, then update the output with the last node.

3.5.4 Representative Feature Selection

This is the last step in the processing phase, where the Representative Features are selected from each cluster. The selection is made using the concepts of graph centrality. Centrality in social networks is an important measure of the influence of a node in the network [76]. One application in healthcare is discovering the central nodes that transmit disease among the nodes that represent connected communities. Another application is identifying the most influential person(s) in a social network.

There are different types of centrality measures:

- Local-Based Centrality Measures- It captures the importance of the node through the partial information around it. Degree centrality is an example of Local-based Centrality.
- Global-Based Centrality Measures- It considers global information giving better ranking results. Betweenness centrality, Closeness centrality, Eigenvector Centrality are examples of Global-based centrality.

In our approach, we have used the Eigenvector Centrality [16] as a measure to determine the importance of a node in a cluster. The reason being it is a Global-based central-

[16] `https://www.sci.unich.it/~francesc/teaching/network/eigenvector.html`

ity measure and gives better results when used on the clusters obtained from maximal cliques [19]. The other centrality measures like Closeness centrality or Edge betweenness works well when the distance between the nodes is present. In our case, the correlation between nodes can be well estimated by Eigenvector Centrality.

The brief description of Eigenvector Centrality is given in the Introduction Chapter (section 1.1). It is an extension of degree centrality. It assigns relative scores to all nodes in the network based on the concept that connections to high-scoring nodes contribute more to the score of the node than connections to low-scoring nodes. It does not depend on the number of neighbors but rather on how important its neighbors are. The importance of its neighbors, in turn, depends on how important their neighbors are, and so on [77]. A node with a few important neighbors has larger Eigenvector Centrality than a node with various neighbors of limited importance. The Eigenvector Centrality x_i of a vertex in an unweighted network is defined to be proportional to the sum of the centralities of the vertex's neighbors. In matrix notation, it is defined as becomes $\lambda x = Ax$, such that x is an eigenvector of the adjacency matrix A. A vector that can be multiplied by the adjacency matrix for a graph and return itself multiplied by a scalar is known as Eigenvector. Following this premise, for a weighted undirected graph G = (V, E, W) with adjacency matrix A where weights denote correlations, we may write for every node x :

$$C_E(x) = \frac{1}{\lambda} \sum_{(x,x') \in E} W(x, x') C_E(x')$$

where $C_E(x)$ is the Eigenvector Centrality of the node x, and λ is some constant. The centrality value of a node is defined as a weighted average of the centrality values of its neighbors. In terms of the adjacency matrix, we can write it as :

$$C_E(x_i) = \frac{1}{\lambda} \sum_j A_{ij} C_E(x_j)$$

In matrix form, the above equation can be written as :

$$\lambda C_E = A C_E$$

where $C_E = (C_E(x_1), ..., C_E(x_n))^T$ and represents the Eigenvector of adjacency matrix A. The Eigenvector Centrality to a weighted network is the leading Eigenvector of the adjacency matrix. It is useful for ranking search results in a citation network. Using the citation frequencies as edge-weights, the Eigenvector Centrality would give papers high scores either if they are cited by many others or if they are cited with high weight by a few others.

In our approach, the process of determining the Eigenvector Centrality score is carried out for all nodes in each cluster. The maximum score corresponds to the node, which is the most central node in the graph. The central node is termed as the Representative Feature.

This concludes the processing phase of the solution approach. After getting the data model from the ingestion phase, a series of operations are performed in this phase. The

construction of the feature dependency graph starts from determining the feature correlations and identifying the threshold to prune out weak edges. The unsupervised feature clustering is performed using the Clique Cover Theory. The Representative Feature set is determined using the concepts of Eigenvector Centrality. The reduced feature set is now carried to the visualization phase.

3.6 Visualization Phase

This is the last phase of the system architecture. The visualization is characterized by 2 parts. In the first part, the features are visualized by using feature graphs like cluster feature graphs, representative feature graphs. The correlation between features is explored using correlation heatmaps. The feature graphs are depicted in the Implementation chapter (section 4.2.1 to 4.2.4). The second part of the visualization is performed by plotting data points of the reduced feature space using different standard multi-dimensional visualization methods. The following are some of the visualization methods used in our approach.

3.6.1 Parallel Coordinate Plot

A parallel coordinate is used for plotting multivariate, numerical data. They are ideal for comparing many variables or features together and identifying the relationships between them. In this plot, each feature is given its own axis, and all the axes are placed in parallel to each other. The axes are normalized to keep the scales uniform. Feature values are plotted as a series of lines that connected across all the axes. It means that an entire path going through all its axes represents a data point.

The parallel coordinate plot can quickly become cluttered if there are a lot of data points spread across the axis. The best way to remedy this problem is through interactivity and is done by a technique known as "Brushing" [17]. Brushing highlights a selected line or collection of lines while fading out all the others. It enables us to isolate sections of the plot a user is interested in while filtering out the noise. Another interactivity mode is "Dragging". It enables to drag a feature axis from one position to another while rearranging the other axes automatically. Reordering by dragging helps to perceive the relationships between adjacent variables and also facilitate in discovering patterns or correlations across variables.

[17] `https://datavizcatalogue.com/methods/parallel_coordinates.html`

An example of the parallel coordinate plot is shown in Figure 3.11.

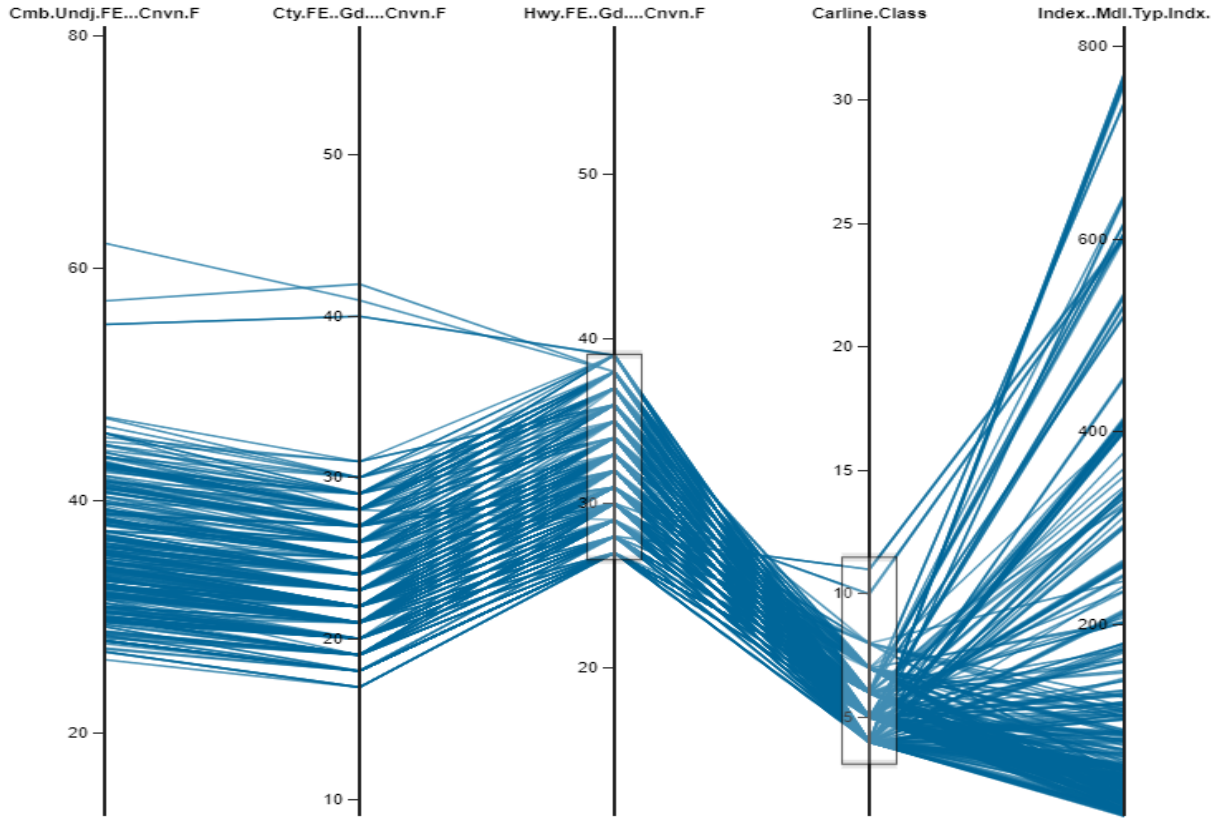


Figure 3.11: The Parallel Coordinate Plot

3.6.2 Scatterplot Matrix

A scatterplot matrix is a collection of scatterplots organized in a matrix format. Each scatterplot shows the relationship between a pair of variables. If there are K variables, the scatter plot matrix will have K rows and K columns, and the ith row and jth column of the matrix is a plot of X_i versus X_j [18]. Brushing is an interaction technique that highlights points based on linked selections across multiple views in a scatterplot matrix. It can be particularly useful for exploring relationships in multi-dimensional data.

[18] https://www.jmp.com/support/help/14-2/scatterplot-matrix.shtml

An example of the scatterplot matrix is shown in Figure 3.12.

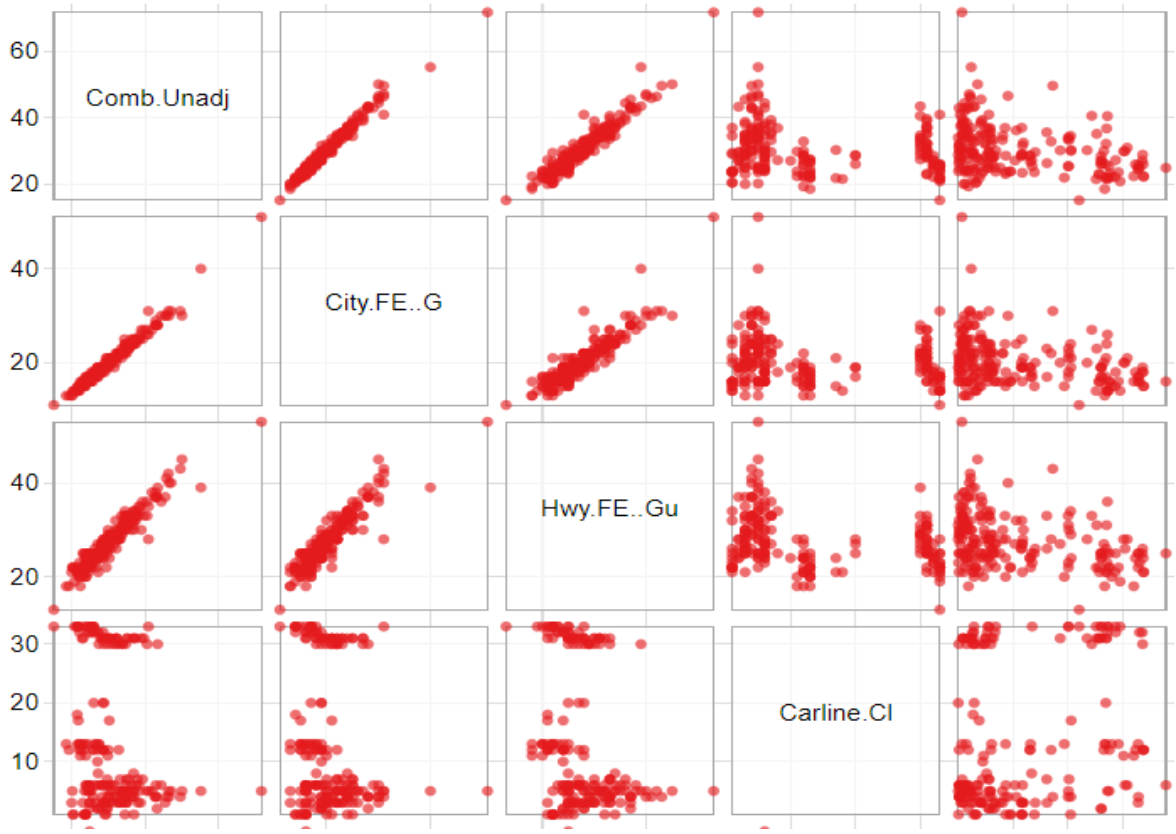


Figure 3.12: The Scatterplot Matrix

3.6.3 Grouped and Stacked Bar Charts

A grouped bar chart, also known as clustered bar graph or multi-set bar chart, is a type of bar graph that is used to represent and compare different categories of two or more groups [19]. The categories are grouped and arranged side-by-side. The bar clusters make it easy to interpret the differences inside a group, and even between the same category across groups. The interactive bar chart enables zoom, scroll, and pan different sections of the chart. The tooltips help to determine the exact values of the bar plots, and the color scheme facilitates interpreting the bars inside the group. The grouped bar chart is shown in Figure 3.13.

A stacked bar chart is another variant of the bar chart. In the stacked bar chart, parts of the data are stacked on top of each other, where each bar displays a total amount, broken down into sub-amounts [20]. Equivalent subsections are plotted in the same color in

[19] https://datavizproject.com/data-type/grouped-bar-chart/
[20] https://datavizcatalogue.com/methods/stacked_bar_graph.html

each bar. This makes it easy to compare both the whole picture and the components of each bar. An example stacked bar chart is shown in Figure 3.14.

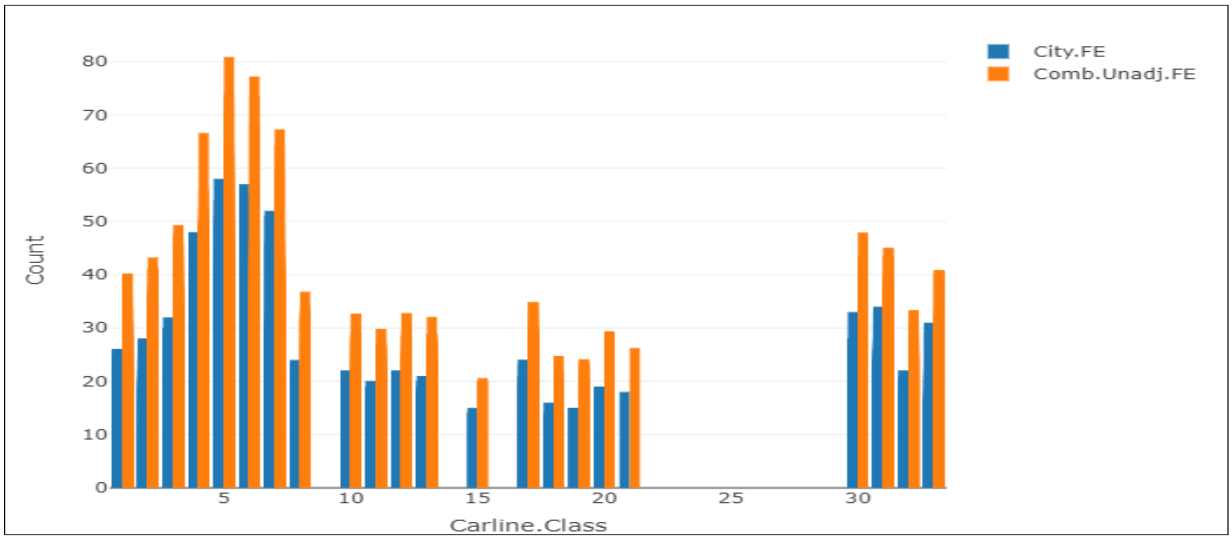


Figure 3.13: The Grouped Bar Chart

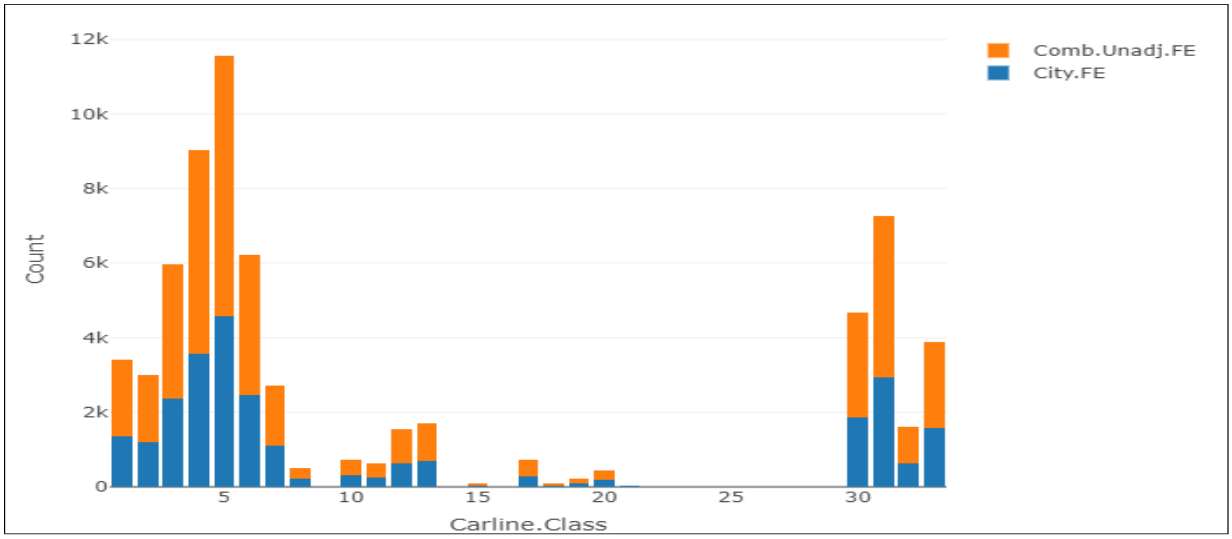


Figure 3.14: The Stacked Bar Chart

3.6.4 3D Scatterplot

The 3D scatter plot displays trivariate points plotted in an X-Y-Z grid. It is particularly useful for investigating the relationships among these variables [21]. A fourth variable can be set to correspond to the color of the markers, thus adding yet another dimension to the plot. It enables us to visualize correlations between different variables in a 3D space. The interactivity is achieved by rotating the axes so that the data points can be investigated from different angles. The basic interactive modes like zoom, expand, tooltips allow the user to understand different aspects of the data in a visually cognizant way. A sample 3D scatterplot is shown in Figure 3.15.

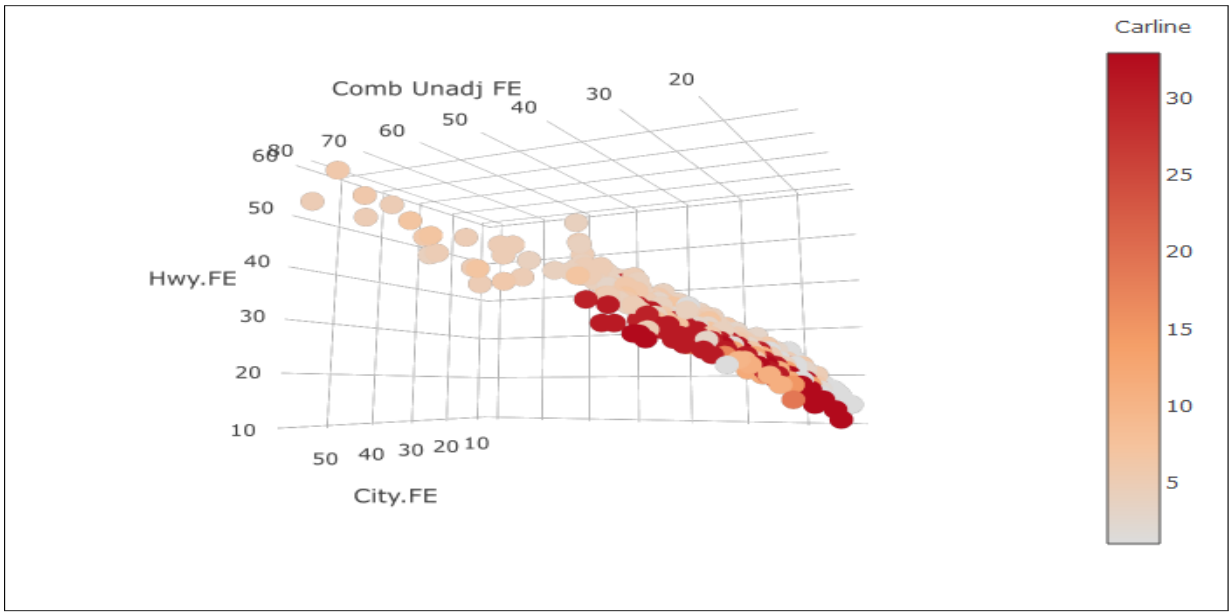


Figure 3.15: The 3D Scatterplot

[21] https://ncss-wpengine.netdna-ssl.com/wp-content/themes/ncss/pdf/Procedures/NCSS/3D_Scatter_Plots.pdf

Chapter 4

Implementation

This chapter describes the implementation details of all the 3 phases in the proposed solution. It consists of various functions, algorithms, and their descriptions required to carry out the steps in different phases of the solution. The entire implementation is carried out in R version 3.4.3 and uses the packages of R to perform different sub-operations. An Integrated Development Environment (IDE) for R known as Rstudio is used to write and organize the code. Throughput this chapter, we have used the automobile dataset, which is described in the Use case Scenario (Chapter 1, Section 1.3) to carry out the implementation.

As described in the previous chapter, the 3 phases of the system architecture are :

1. Ingestion Phase
2. Processing Phase
3. Visualization Phase

4.1 Ingestion Phase

This phase ingests and parses the dataset to create the data model as required by our approach. Since we have considered the automobile dataset, the functions described here are used to deal with the corresponding automobile data. There are 4 main steps in the ingestion phase:

1. Data Cleaning
2. Data Pre-processing
3. Data Imputation
4. Data Segregation

4.1.1 Data Cleaning

Before cleaning the data, it is necessary to read the data and store it in a format that can be used for processing. Since we have the automobile data in CSV format, we can read it and store it as a dataframe. A dataframe in R is a table or a two-dimensional array-like structure in which each column contains values of one variable, and each row contains one set of values from each column [1]. A screenshot of the automobile dataframe in R is shown below:

	Mfr.Name	Division	Carline	Verify.Mfr.Cd	Transmission	Air.Aspir.Method
1	Honda	Acura	NSX	HNX	Auto(AM-S9)	TC
2	FCA US LLC	ALFA ROMEO	4C	CRX	Auto(AM6)	TC
3	Volkswagen Group of	Audi	R8 AWD	VGA	Auto(AM-S7)	*NA*
4	Volkswagen Group of	Audi	R8 RWD	VGA	Auto(AM-S7)	*NA*
5	Volkswagen Group of	Audi	R8 Spyder AWD	VGA	Auto(AM-S7)	*NA*
6	Volkswagen Group of	Audi	R8 Spyder RWD	VGA	Auto(AM-S7)	*NA*
7	Volkswagen Group of	Audi	TT Roadster quattro	VGA	Auto(AM-S6)	TC
8	BMW	BMW	M4 DTM Champions Edition	BMX	Auto(AM-S7)	TC
9	Volkswagen Group of	Bugatti	Chiron	VGA	Auto(AM-S7)	TC
10	General Motors	Chevrolet	CORVETTE	GMX	Auto(S8)	*NA*
11	General Motors	Chevrolet	CORVETTE	GMX	Auto(S8)	SC
12	General Motors	Chevrolet	CORVETTE	GMX	Manual(M7)	SC
13	General Motors	Chevrolet	CORVETTE	GMX	Manual(M7)	*NA*
14	Ferrari	Ferrari North America, Inc.	488 gtb	FEX	Auto(AM7)	TC
15	Ferrari	Ferrari North America, Inc.	488 gtb	FEX	Auto(AM7)	TC

Figure 4.1: The Automobile Dataframe

The major challenge in data cleaning is to deal with missing values. The percentage of missing values varies throughout the features. We have used the rules mentioned in Chapter 3, Section 3.4.1, to clean the dataset. The other challenge is to identify the features that are unique for all data points but is insignificant and can be removed. For example, the automobile dataset has a value for every row known as rowID. This column can be removed because it won't have correlations with any other column.

For data cleaning, we have used 3 built-in R packages; tidyverse, dplyr and janitor. The tidyverse is a collection of R packages designed for data science [2]. It is the most powerful collection of R packages for preparing, wrangling, and visualizing data. The dplyr package is a subset of tidyverse package which is used for data wrangling and transformation. It contains various operations like select, filter, count the number of rows and columns in the dataframe. The janitor package [3] contains a function called `"remove_empty"`, which is used to remove all empty or NA rows and columns. The rest of the function implements user-defined functions to clean the data.

[1] `https://www.tutorialspoint.com/r/r_data_frames.htm`
[2] `https://urlzs.com/rsDdF`
[3] `https://www.rdocumentation.org/packages/janitor/versions/1.2.0`

The following function "dataClean" is used to perform data cleaning:

```
# Load the required libraries
library(tidyverse)
library(dplyr)
library(janitor)

dataClean <- function(){
  # Read the data
  data <- read.csv("AutoData.csv", header=T, na.strings=c("","NA"))
  total_rows <- nrow(data) # Total rows in the dataframe
  total_cols <- ncol(data) # Total Columns in the dataframe
  filtered_cols <- data

  # Remove all empty/NA rows and columns
  removeAllEmptyCols <- remove_empty(filtered_cols, which = c("rows
     ", "cols"))

  # Remove columns that contains more than 60 percent NA values
  removeNACols <- removeAllEmptyCols[, colMeans(is.na(
     removeAllEmptyCols)) <= .4]

  # Remove Columns that contains all Unique values
  removeUniqueCols <- removeNACols[, sapply(removeNACols, function(
     col) length(unique(col)) > 1)]

  # Loop to identify duplicate values and removing them.
  allCols <- colnames(removeUniqueCols)
  dupCol <- c()
  for(i in 1:ncol(allCols)) {
    colName <- allCols[i]
    value <- "Desc"
    if(grepl(value, colName)){

      if( nlevels(allCols[,i]) == nlevels(allCols[,i-1])  )
        dupCol <- c(dupCol, i-1)
    }

  }
  allCols <- allCols[,-(dupCcol)]
  cleanData <- allCols

  # return the Cleaned Data
  return(cleanData)
}
```

4.1.2 Data Pre-processing

The raw data often contains special symbols or extra spaces. The challenge in data pre-processing is to identify the uninterpretable the special symbols and extra spaces in the data and remove them. The type conversion is also an important aspect in the pre-processing as it ensures the segregation can be performed properly in the next step. The steps in data pre-processing are mentioned in section 3.4.2.

The data pre-processing is performed by the user-defined "dataPreprocess" function. It takes the cleanData dataframe as an argument which is obtained from the previous step. The dplyr package loaded earlier is applied on the columns to check the datatype. It has a built-in function known as "select_if" that selects a value based on the condition. In our code, we have specified 2 conditions; "as.numeric" and "as.factor" to check for numerical and categorical datatypes respectively. The base package [4] is used to detect special characters as given by the regular expression. The built-in "gsub" function is given an argument [:punct:] that stands for punctuation characters. It removes the special characters and returns the dataframe. The "trimws" is a built-in function also present in the base package used to remove whitespaces. The argument "l" in the function is used for the leading whitespaces. The "tolower" function given by stringr package [5] is used to convert the categorical values to lower case. The processed dataframe is combined and returned, as shown in the following function snippet below:

```
# Load the required libraries
library(base)
library(stringr)

dataPreprocess <- function(cleanData) {
  # Check the data type, as.numeric for numerical and as.factor for
      categorical
  num_cols <- select_if(cleanData, as.numeric)
  cat_cols <- select_if(cleanData, as.factor)

  # Regular expression to remove the special characters
  cat_cols <- as.data.frame(gsub("[[:punct:]]", "", as.matrix(cat_
     cols)))
  # Removes leading whitespace from character strings.
  trimws(cleanData, "l")
  # Converts the categorical values into lower case
  cat_cols <- tolower(cat_cols)
  # Bind and return the processed Data
  processedData <- cbind(num_cols, cat_cols)
  return(processedData)

}
```

[4] https://www.rdocumentation.org/packages/base/versions/3.6.1
[5] https://www.rdocumentation.org/packages/stringr/versions/1.4.0

4.1.3 Data Imputation

The data imputation is carried out by the method known as "Fast imputation of missing values". This method is given by the "missRanger" package [6] in R. It takes the processData as an argument and performs the imputation. A few more parameters are also set to get accurate results. The "seed" parameter is set to initialize the random generator. Since the imputation method uses chained random forests, the "num.trees" parameter specifies the number of decision trees to be used. The "pmm.k" parameter is used to specify the number of candidate non-missing values to sample from in the predictive mean matching step. The imputation function is shown below:

```
# Load the required library
library(missRanger)

imputeData <- function(){
   imputedData <- missRanger(processedData, pmm.k = 3, seed =
      75757,  num.trees = 80)
}
```

4.1.4 Data Segregation

The dplyr package loaded earlier is used to segregate the dataframe based on the datatype. The "`select_if`" function is based on two conditions namely; "is.numeric" and "is.factor". Additionally, a constraint is defined to check the number of factors in each column of the dataframe. Factors are variables in R, which take on a limited number of different values and are also called categorical variables. In our case, we have specified an upper bound of 10, which means the factors lower or equal to 10 are also considered as categorical variables. Finally, the dataframe is segregated to generate two dataframes "finalCatCols" and "finalNumCols" respectively. The segregation function "`segData`"is shown below:

```
segData <- function(){
  # Segregate the data based on data type
  num_cols <- select_if(imputedData, is.numeric)
  cat_cols <- select_if(imputedData, is.factor)

  finalCatCols <- cat_cols
  #Specify the upper bound on the number of factors(10) to be
     considered as numerical
  finalNumCols <- num_cols[, sapply(num_cols, function(col) length(
     unique(col))) > 10]
```

[6] `https://www.rdocumentation.org/packages/missRanger/versions/2.1.0/topics/missRanger`

```
  # The factors lower than or equal to 10 are considered as
     categorical
  if( ncol(finalNumCols) !=  ncol(num_cols)){
    finalCatCols  <- cbind(cat_cols,  num_cols[, sapply(num_cols,
       function(col) length(unique(col))) <= 10])
    }
}
```

This completes the description of functions in the ingestion phase. The data model consisting of categorical and numerical feature groups is created in the form of 2 dataframes. The "finalCatCols" dataframe consists of categorical features, whereas the "finalNumCols" dataframe consists of numerical features. The two dataframes are shown in Figure 4.2 and 4.3.

	Mfr.Name	Division	Carline	Verify.Mfr.Cd	Transmission	Air.Aspiration.Method.Desc	Trans.Desc
1	Honda	Acura	NSX	HNX	Auto(AM-S9)	Turbocharged	Automated Manual- Selectable (e.
2	FCA US LLC	ALFA ROMEO	4C	CRX	Auto(AM6)	Turbocharged	Automated Manual
3	Volkswagen Group of	Audi	R8 AWD	VGA	Auto(AM-S7)	Naturally Aspirated	Automated Manual- Selectable (e.
4	Volkswagen Group of	Audi	R8 RWD	VGA	Auto(AM-S7)	Naturally Aspirated	Automated Manual- Selectable (e.
5	Volkswagen Group of	Audi	R8 Spyder AWD	VGA	Auto(AM-S7)	Naturally Aspirated	Automated Manual- Selectable (e.
6	Volkswagen Group of	Audi	R8 Spyder RWD	VGA	Auto(AM-S7)	Naturally Aspirated	Automated Manual- Selectable (e.
7	Volkswagen Group of	Audi	TT Roadster quattro	VGA	Auto(AM-S6)	Turbocharged	Automated Manual- Selectable (e.
8	BMW	BMW	M4 DTM Champions ...	BMX	Auto(AM-S7)	Turbocharged	Automated Manual- Selectable (e.
9	Volkswagen Group of	Bugatti	Chiron	VGA	Auto(AM-S7)	Turbocharged	Automated Manual- Selectable (e.
10	General Motors	Chevrolet	CORVETTE	GMX	Auto(S8)	Naturally Aspirated	Semi-Automatic
11	General Motors	Chevrolet	CORVETTE	GMX	Auto(S8)	Supercharged	Semi-Automatic
12	General Motors	Chevrolet	CORVETTE	GMX	Manual(M7)	Supercharged	Manual
13	General Motors	Chevrolet	CORVETTE	GMX	Manual(M7)	Naturally Aspirated	Manual
14	Ferrari	Ferrari North A...	488 gtb	FEX	Auto(AM7)	Turbocharged	Automated Manual
15	Ferrari	Ferrari North A...	488 gtb	FEX	Auto(AM7)	Turbocharged	Automated Manual

Figure 4.2: The finalCatCols Dataframe

4.2 Processing Phase

This phase contains the functions and algorithms to perform unsupervised feature clustering and selection. It takes the segregated dataframes (finalCatCols and finalNumCols) from the ingestion phase and proceeds to carry out the following main steps:

1. Feature Correlations
2. Identifying Threshold Coefficient
3. Applying Clique Cover Algorithm
4. Representative Feature Selection

	Index..Model.Type.Index.	Eng.Displ	City.FE..Guide....Conventional.Fuel	Hwy.FE..Guide....Conventional.Fuel
1	57	3.5	21	22
2	410	1.8	24	34
3	65	5.2	14	22
4	71	5.2	14	25
5	66	5.2	14	22
6	72	5.2	14	25
7	46	2.0	23	30
8	488	3.0	17	24
9	38	8.0	9	14
10	278	6.2	15	25
11	223	6.2	13	23
12	285	6.2	15	22
13	276	6.2	16	25
14	142	3.9	15	22
15	143	3.9	16	22

Figure 4.3: The finalNumCols Dataframe

4.2.1 Feature Correlations

This is the first step to construct the feature dependency graph. Feature correlations are used as edge-weights in the feature graph. As described in the Section 3.5.1, a couple of correlation measures are used in this step. The Maximal Information Coefficient (MIC) is used on the "finalNumCols" dataframe consisting of numerical feature groups. The Chi-square test of association followed by Cramer's V is used on the "finalCatCols" dataframe consisting of categorical feature groups. The output of this step is a couple of square weighted adjacency matrices consisting of correlations between the features in the respective feature group. We have named "resultMIC" and "resultCramer" as the two resultant matrices. The algorithm for the feature correlations is shown in Algorithm 2.

Algorithm 2: Algorithm for Feature Correlations

Procedure : *featureCorrelations*;
Input : Numerical and Categorical feature groups (finalNumCols and finalCatCols).
Output : resultMIC and/or resultCramer.
if finalNumCols has more than 1 feature **then**
  resultMIC <- numCorr(finalNumCols)
end
else if finalCatCols has more than 1 feature **then**
  resultCramer <- catCorr(finalCatCols)
end

Maximal Information Coefficient (MIC)

To apply MIC on every pair of numerical feature groups, a package in R known as minerva [7] is used. It is a wrapper for "minepy" implementation of Maximal Information-based Nonparametric Exploration statistics (MINE). One of the built-in function present in minerva is "mine". The mine function takes the dataframe as an argument and produces a number of results like MIC, Maximum Asymmetry Score (MAS), Maximum Edge Value (MEV). The process of pairwise determining MIC on the entire dataframe can be expensive. To cope with that, a package called parallel [8] is used, that provides the support for parallel computation. The "detectCores" function in the parallel package is used to assign the maximum number of CPU cores on the current host to carry out the MIC process. The code snippet of "numCorr" function to pass the "finalNumCols" dataframe is shown below:

```
# Load the required libraries
library(minerva)
library(parallel)

numCorr <- function(finalNumCols){
   allMine <- mine(finalNumCols, n.cores= detectCores() , var.thr
      =0.0, use="pair", est="mic_e")
   resultMIC <- allMine$MIC

   return resultMIC
}
```

The resultMIC matrix obtained after applying the above function is shown in Figure 4.4.

	Index..Model.Type.Index.	Eng.Displ	City.FE..Guide....Conventional.Fuel	Hwy.FE..Guide....Conventional.Fuel	Comb.FE..Guide....Conventional.Fuel
Index..Model...	1.0000000	0.2255489	0.1131736	0.1270602	0.1180084
Eng.Displ	0.2255489	0.9981902	0.6246488	0.4960230	0.5960401
City.FE..Guid...	0.1131736	0.6246488	0.9965252	0.6778872	0.8895654
Hwy.FE..Guid...	0.1270602	0.4960230	0.6778872	0.9996537	0.8059559
Comb.FE..Gu...	0.1180084	0.5960401	0.8895654	0.8059559	0.9991448

Figure 4.4: The resultMIC Matrix

[7] https://cran.r-project.org/web/packages/minerva/minerva.pdf
[8] https://www.rdocumentation.org/packages/parallel/versions/3.6.1

Chi-Square Test of Association and Cramer's V

To carry out Chi-square tests, a package in R known as MASS [9] needs to be installed. This package contains various functions and datasets to carry out many applied statistics operations. It has a built-in "table" function to generate the contingency tables. It contains a function "chisq.test" that takes the contingency table as an argument and outputs the results of the Chi-square test of association. Since we are interested in determining the Cramer's V between every pair of categorical variables, another package is known as DescTools [10] is used. This package is a collection of basic statistic functions, association measures, and wrappers for efficiently describing data. The association function "PairApply" takes the dataframe and the operation to be applied as a parameter. When passed "CramerV" as the operation parameter, the PairApply function performs the Chi-square test of association followed by Cramer's V on the entire dataframe and produces the result in the matrix format. The code snippet for the function "catCorr" is shown below:

```
# Load the required libraries
library(MASS)
library(DescTools)

catCorr <- function(finalCatCols){
   resultCramer <- PairApply(finalCatCols, CramerV, symmetric =
      TRUE)

   return resultCramer
}
```

The resultCramer matrix obtained after applying the above function is shown in Figure 4.5.

	Mfr.Name	Division	Carline	Verify.Mfr.Cd	Transmission	Air.Aspiration.Method.Desc	Trans.Desc	X..Cyl
Mfr.Name	1.0000000	1.0000000	1.0000000	1.0000000	0.4411541	0.6167915	0.5568196	0.3566101
Division	1.0000000	1.0000000	1.0000000	1.0000000	0.4769301	0.6404904	0.6098077	0.6140636
Carline	1.0000000	1.0000000	1.0000000	1.0000000	0.8774864	0.8566487	0.8665832	0.9720092
Verify.Mfr.Cd	1.0000000	1.0000000	1.0000000	1.0000000	0.4411541	0.6167915	0.5568196	0.3566101
Transmission	0.4411541	0.4769301	0.8774864	0.4411541	1.0000000	0.3511772	1.0000000	0.2986877
Air.Aspiration...	0.6167915	0.6404904	0.8566487	0.6167915	0.3511772	1.0000000	0.1989290	0.1735179
Trans.Desc	0.5568196	0.6098077	0.8665832	0.5568196	1.0000000	0.1989290	1.0000000	0.2242399
X..Cyl	0.3566101	0.6140636	0.9720092	0.3566101	0.2986877	0.1735179	0.2242399	1.0000000

Figure 4.5: The resultCramer Matrix

This completes the implementation of the feature correlation step for both the feature

[9] https://www.rdocumentation.org/packages/MASS/versions/7.3-47
[10] https://www.rdocumentation.org/packages/DescTools/versions/0.99.19

groups. Both the matrices obtained, i.e., resultMIC and resultCramer are square weighted adjacency matrices. The resultant graph obtained by considering such a matrix is a complete weighted feature graph because every pair of feature nodes is connected by a unique edge. The complete feature graph corresponding to the resultCramer matrix is shown in Figure 4.6.

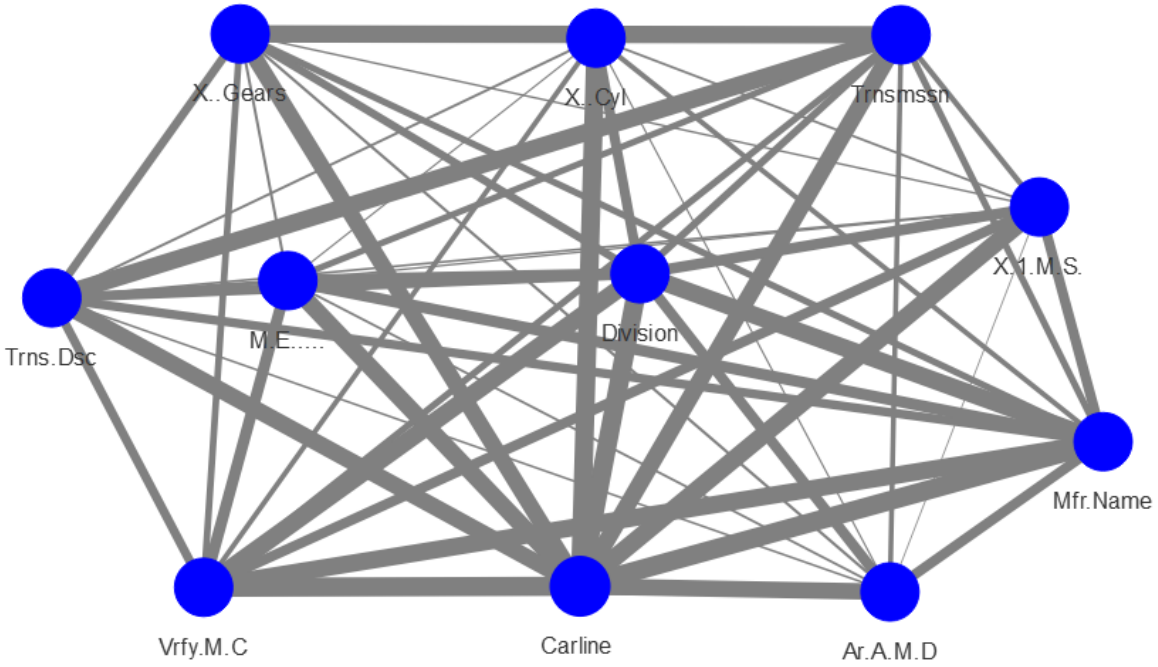


Figure 4.6: The Categorical Complete Feature Graph

4.2.2 Identifying Threshold Coefficient

This step is used to prune out the weak associations from the complete feature graph. The K-Nearest Neighbour algorithm is implemented to determine the K strongest neighbors, but instead of using the distance, the correlation coefficient is used as a way to determine the threshold. The value of K is determined by taking the square root of the number of features present in the respective feature groups. A user-defined function called `"makeAffinity"` is defined. It is supposed to take a couple of parameters; the correlation matrix(resultCramer/resultMIC) and the value of K. The function returns an affinity matrix. The affinity matrix contains all the nodes, but the connections below the threshold are replaced by zeros and are symmetric in nature. It is the matrix representation of the feature dependency graph, which is obtained by pruning the weak edges from the complete graph. The algorithm for identifying the threshold coefficient is given in Algorithm 3.

Algorithm 3: Algorithm for Identifying Threshold Coefficient

Procedure : *makeAffinity*;
Input : result (resultMIC and/or resultCramer) and K.
Output : affinity matrix (affMat).
Step 1: totalNodes ←length(result).
Step 2: **if** $K >$ totalNodes **then**
  affMat <- result
end
else
  /* Determine strong connections for every feature node */
  For each i-th feature in totalNodes.
    strongConnections ←sort(result[i,], decreasing=TRUE)[1:K]
  /* Make the affinity matrix symmetric in nature */
    For each s-th feature in strongConnections.
      j <- which(result[i,] == s)
      affMat[i,j] <- result[i,j]
      affMat[j,i] <- result[i,j]
end

The code snippet for "`makeAffinity`" function is shown below:

```
makeAffinity <- function(result, K) {
  totalNodes <- length(result[,1])
  if (K >= totalNodes) {  # fully connected
    affMat <- result
  } else {
    affMat <- matrix(rep(0, totalNodes^2), ncol= totalNodes)
    for(i in 1:totalNodes) { # for each line
      # only connect to those points with larger similarity
      strongConnections <- sort(result[i,], decreasing=TRUE)[1:K]
      for (s in strongConnections) {
        j <- which(result[i,] == s)
        affMat[i,j] <- result[i,j]
        affMat[j,i] <- result[i,j] # to make an undirected graph,
           ie, the matrix becomes symmetric
      }
    }
  }
  return affMat
}
```

The feature dependency graph obtained by pruning the edges from the complete graph is shown in Figure 4.7.

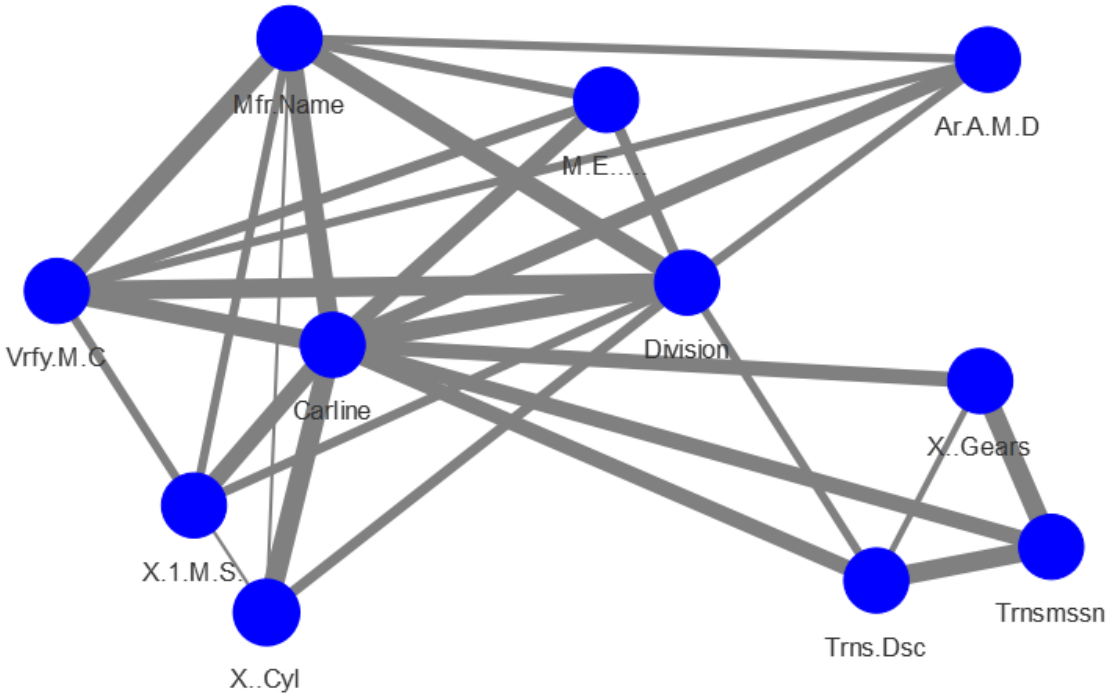


Figure 4.7: The Categorical Feature Dependency Graph

4.2.3 Applying Clique Cover Algorithm

The Clique Cover is a graph clustering algorithm used to cluster the nodes in the feature dependency graph. The algorithm works on the graph object in order to determine cliques and maximal cliques in the graph. The graph object is a vector representation of the adjacency matrix where each element of the vector denotes the connections between the nodes [11]. The algorithm proceeds by generating the graph object from the affinity matrix. Then it implements different functions to determine cliques, maximal cliques, number of maximal cliques. It then calculates the weight of each maximal clique and determines the maximal clique having the maximum weight. The output of the algorithm is the set of clustered nodes. The "makeAffinity" and the "cliqueCover" algorithm are recursively applied to generate clusters of features of different sizes, as described in Chapter 3 Algorithm 1. The clique Cover algorithm is given in Algorithm 4.

[11] `https://www.r-bloggers.com/r-graph-objects-igraph-vs-network/`

Algorithm 4: Clique Cover Algorithm

Procedure : *cliqueCover*;
Input : affMat.
Output : clusterNodes.
/* Create the graph object */
Step 1: graphObj ← generate_graph(affMat)
/* Determine the cliques */
Step 2: getCliq ← cliques(graphObj)
/* Determine the maximal cliques */
Step 3: getMaxCliq ← max_cliques(graphObj, getCliq)
/* Determine the number of maximal cliques */
Step 4: countMaxCliq ← count(getMaxCliq)
/* Call the initialize function to set all weights as null */
Step 5: initialize()
/* Call find_weights function to determine the weight of each maximal clique */
Step 6: weightOfCliq ← find_weights(countMaxCliq, affMat)
/* Determine the maximal clique having maximum weight */
Step 7: maxWeight ← get_max_weight(weightOfCliq)
/* Get cluster nodes Ids from the maximal clique having the maximum weight */
Step 8: clusterNodes ← getMaxCliq[maxWeight]

The igraph [12] package is used to generate graph objects. It provides a set of data types and functions for the implementation of graph algorithms, handling large graphs, and performing network analysis. The package contains a built-in function called "graph_from_adjacency_matrix" that generates the graph object. The code snippet to generate graph objects is shown below:

```
# Load the required libraries
library(igraph)

generate_graph <- function(affMat){
  graphObj <- graph_from_adjacency_matrix(affMat, mode = "
     undirected", weighted = TRUE)
  return(graphObj)
}
```

The package also contains the functions to determine the cliques and maximal cliques. The "clique" function takes the graph object, returned by the above function, as an argument. It also asks for two parameters "min" and "max", which denotes the lower and upper limit on the size of the cliques to find. The min and max parameters are set to NULL by default, which means to find cliques of all sizes. The "max_clique" function takes the graph object along with a "subset" parameter as an argument. The subset parameter is

[12] https://www.rdocumentation.org/packages/igraph/versions/0.3.1

a vector of nodes which is returned by the clique function. It implies that the maximal clique is the subset of the clique. The function template is shown below:

```
# Load the required libraries
library(igraph)

getCliq <- cliques(graphObj, min = NULL, max = NULL)
getMaxCliq<- max_cliques(graphObj, min = NULL, max = NULL, subset =
    getCliq)
countMaxCliq <- count(getMaxCliq)
```

A few more functions are defined, which are needed to carry out the Clique Cover clustering algorithm. The "find_weights" function is used to determine the weight of all the maximal cliques present in the graph. The function iteratively calls another function "find_each_weight" to determine the weight of each maximal clique and update the "wt_sum" vector to keep a track of the weights.

```
find_weights <- function(countMaxCliq,affMat){
  for ( maxCliq in c(1: countMaxCliq)) {
    find_each_weight(maxCliq,affMat)
  }
  return(wt_sum)
}
```

The "initialize" function is used to reset the counter and create an empty vector "wt_sum" for each iteration of maximal clique determination.

```
initialize <- function(){
   Counter <- 0
   wt_sum <- vector()
}
```

The "find_each_weight" function considers each maximal clique and determines the pair of vertices that are connected by an edge. It sums up the weight of the edges identified in the maximal clique. The summed up weight is then appended to empty vector iteratively.

```
find_each_weight <- function(c, mat){
  cliq <- get_cliq[[c]]
  vertex_mat <- combn(cliq, 2)
  transposeVer <- t(vertex_mat)
  cliqSum <- sum(mat[transposeVer])
  }
   Counter <- .GlobalEnv$Counter + 1
   wt_sum[[Counter]] <- cliqSum
}
```

Once the weight of all the maximal cliques are determined, the "get_max_weight" function determines the maximum weight. It takes vector of weights as an input argument and returns the index having maximum weight.

```
get_max_weight <- function(wt_sum){
 maxWeight <- max(wt_sum)
  return(maxWeight)
}
```

The "get_cluster_nodes" takes the index of the maximum weight from the previous function and returns the maximal clique nodes with the maximum weight.

```
get_cluster_nodes <- function(maxWeight){
 clusterNodes <- getMaxCliq[maxWeight]
  return(clusterNodes)
}
```

The cluster graph obtained after after applying Clique Cover algorithm is shown in Figure 4.8.

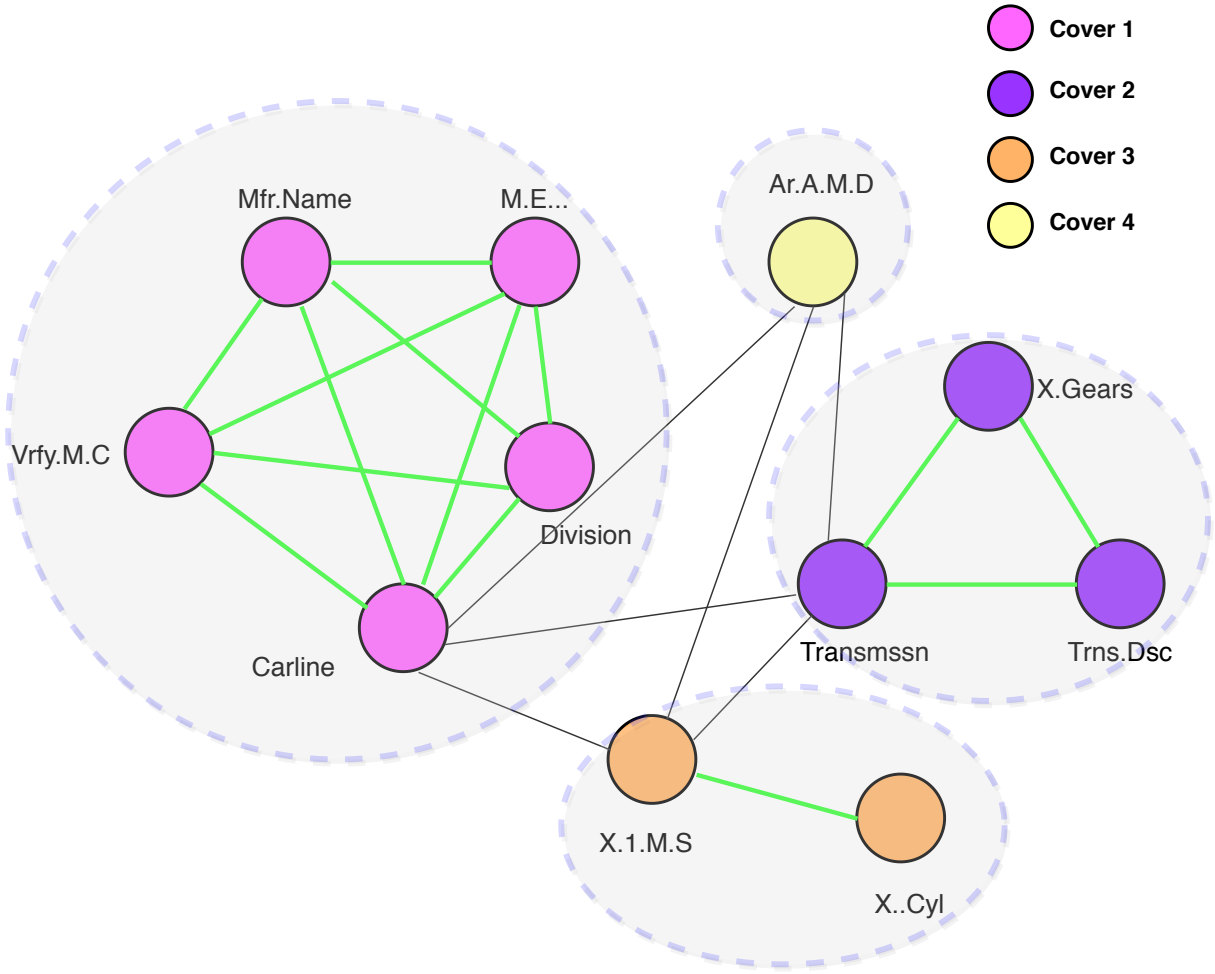


Figure 4.8: The Categorical Cluster Graph

The Figure 4.8 shows the cluster graph made from the categorical features. After applying the Clique Cover algorithm, the complete feature graph (Figure 4.6) is transformed into the cluster graph having 4 clusters or covers of different sizes.

4.2.4 Representative Feature Selection

The Representative Feature is selected by determining the Eigenvector Centrality score of each node present in the cluster. The node having the maximum score is termed as the Representative Feature. The algorithm requires the clusterNodes(obtained from the previous step) and the initial feature correlation matrix as the two input parameters. It proceeds by generating the graph object of individual clusters. It then determines the Eigenvector Centrality scores of each node present in the clustered graph object. It determines the node having the maximum score and returns the Representative Feature. The algorithm for Representative Feature selection is given in Algorithm 5:

Algorithm 5: Algorithm for Representative Feature Selection

Procedure : *repFeatSelection*;
Input : clusterNodes and result (resultMIC and/or resultCramer).
Output : Representative Feature (repFeature).
/* Generate graph object for each cluster */
Step 1: clusGraphObj <- individualClusterGraph(clusterNodes, result)
/* Determine the Eigenvector Centrality score and identify the Representative Feature */
Step 2: repFeature <- repFeatureSelection(clusGraphObj)

In our approach, a function called "`individualClusterGraph`" is defined. It takes two arguments; the node Ids of each cluster and the result from the Feature correlation matrix. It first creates an empty matrix having the same size as the number of nodes in each cluster. It then populates the matrix by using a subset of the feature correlation matrix(resultCramer/resultMIC) corresponding to the node Ids of each cluster. The function "`graph_from_adjacency_matrix`" provided by the igraph package, is used to generate the graph object from the adjacency matrix. The pseudocode of the function is shown below:

```
individualClusterGraph <- function(clusterNodeIds, result){

# Initialize an empty matrix of same size as each cluster.
currClus <- matrix(data=NA, nrow=length(clusterNodeIds), ncol=
   length(clusterNodeIds)
```

```
# Populate the matrix with the entities from correlation matrix
for (r in 1:nrow(currClus)) {
   for (c in 1:ncol(currClus))
        currClus[r,c] = result[currClus[r],currClus[c]]
  }

#  Generate the graph object from the matrix
clusGraphObj <- graph_from_adjacency_matrix(currClus, mode = "
   undirected", weighted = TRUE)

return(clusGraphObj)

}
```

The Eigenvector Centrality is calculated on the graph object determined from the above function. The igraph package of R provides a function called "eigen_centrality" that calculates the centrality scores. It takes the graph object as an argument and requires to set a few parameters. The "directed" parameter is set as False because we have the undirected feature graph. The "scale" parameter is set to True in order to scale the result to have a maximum score of one. The "weights" parameter is used to give the edge-weights for calculating the weighted Eigenvector Centrality score of the nodes. The node having the maximum score is considered as the Representative Feature. The code snippet of "represFeatureSelection" is shown below:

```
represFeatureSelection <- function(clusGraphObj){

# Determine the Eigenvector Centrality scores of the nodes in the
   cluster graph
 eigenScores <- eigen_centrality(indiClusGraph, directed=FALSE,
    scale= TRUE, weights=E(indiClusGraph)$weight)$vector

 # Select the node with the maximum score
 repFeature <- which.is.max(eigenScores)

 return(repFeature)
}
```

The Representative Feature graph is shown in Figure 4.9. As seen, the graph is less cluttered as compared to the complete Feature graph (Figure 4.6). Both the graphs show the features from the categorical feature groups, but the Representative Feature graph only shows the important nodes and their corresponding edges, making it more clear and interpretable.

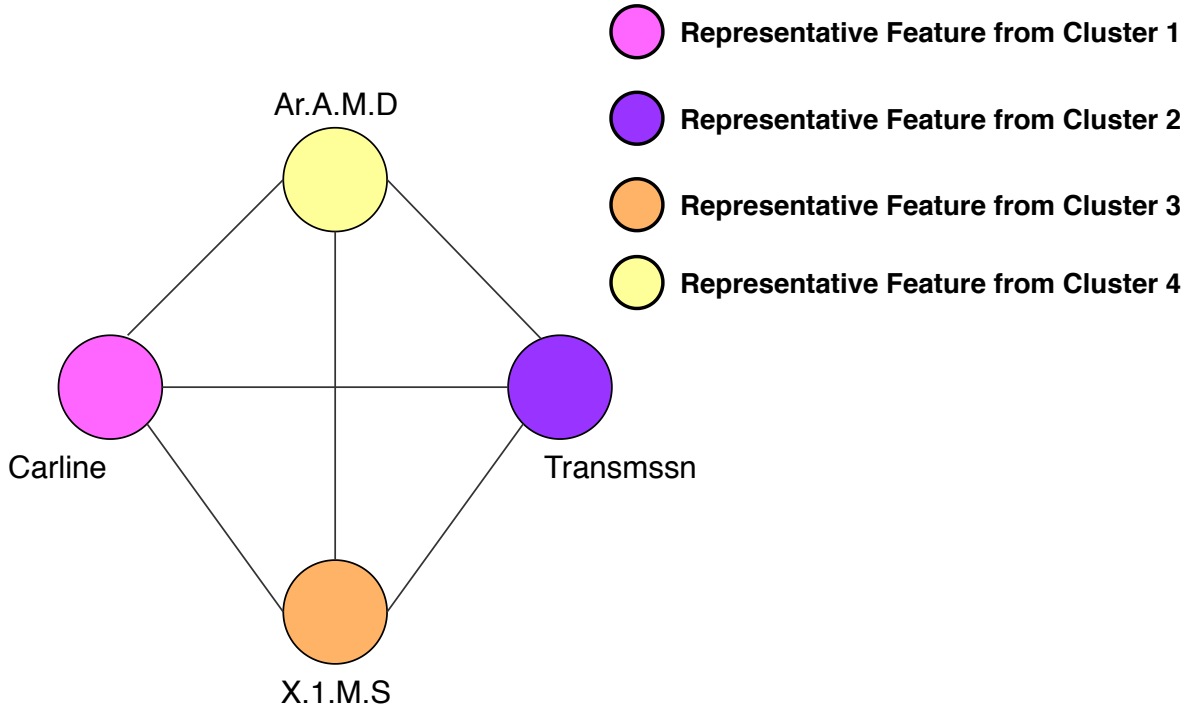


Figure 4.9: The Representative Feature Graph

4.3 Visualization Phase

The visualization phase is characterized by 2 aspects. First, we have visualized the features and their correlations using feature graphs such as the complete feature graph (Figure 4.6), cluster feature graph (Figure 4.8), and representative feature graph (Figure 4.9). In the second part, the data points corresponding to the set of Representative Features are visualized using various standard methods. In this section, the implementation of some of the visualization methods in R is described.

4.3.1 Parallel Coordinate Plot

The parcoords library [13] in R create a well designed and interactive parallel-coordinates chart. It is based on a similar fashion as the d3 parallel plot. It takes the dataframe as the argument and allows interactivity by setting the parameters. The "brushMode" parameter enables the brushing on the axis of the parallel coordinate. It mitigates the cluttering by highlighting a selected line or collection of lines while fading out all the others. The "reorderable" parameter allows dragging the axis while rearranging the other axes automatically. The function `"renderParcoords"` is a built-in function in the parcoords library that is used to render a parallel coordinate plot. The pseudocode for creating parallel coordinates is shown below :

[13] `https://github.com/timelyportfolio/parcoords`

```
library(parcoords)
renderParcoords(
    parcoords( selectedFeatureDF, rownames=F,
    brushMode="1d" ,reorderable = TRUE)
)
```

The Parallel Coordinate Plot generated for the selected datasets are shown in the Evaluation chapter (Section 5.7). We have used the Parallel Plot as a visualization method for performing qualitative evaluation of our proposed approach.

4.3.2 Scatterplot Matrix

It is a collection of scatter plots displayed in a matrix format. The pairsD3 library [14] is used to create an interactive matrix of scatterplots. The coordinates of points given as numeric columns of a matrix or dataframe. The built-in function "renderPairsD3" can take a few additional parameters for accurate rendering. The names of the variables are set in the "labels" parameter. The magnification of the plotting symbol is set by the "cex" parameter. There are several other parameters like theme, opacity, tooltip, width, margin, which are set to default values and can be easily customized.

```
library(pairsD3)
renderUI({pairsD3Output("pD3", width = "850px", height = "650px")})

output$pD3 <- renderPairsD3({pairsD3(selectedNumDF ,cex =3, labels=
    colnames(selectedNumFeat))
})
```

4.3.3 Grouped and Stacked Bar Charts

The grouped and stacked bar charts are the two variations of the bar chart that is used to represent and compare different categories of two or more groups. The grouped bar charts place the categories side by side, whereas the stacked bar chart places the categories on top of each other like the stack. The plotly library [15] in R can be used to plot both these charts. It provides a function known as "plot_ly" which maps the R objects to plotly.js, a web-based interactive charting library. To generate a grouped or stacked bar charts, a series of functions is pipelined and called. The first function is the default function "plot_ly" that takes the dataframe and sets the x-axis. In the vertical bar chart, the y-axis is set as count by default. The pipelined function "add_trace" takes the additional columns to plot along with the parameter to set the color combinations. The final pipelined function is "layout" that sets the title name, width, height, and margin. The change in "barmode" parameter in the layout function decides the layout. If the barmode is set to "group", it

[14] https://www.rdocumentation.org/packages/pairsD3/versions/0.1.0
[15] https://www.rdocumentation.org/packages/plotly/versions/4.9.0/topics/plot_ly

renders a grouped bar chart, and if it is set to “stack”, it renders a stacked bar chart. The pseudocode is shown below:

```
library(plotly)
plot_ly(finalCatCols, x = finalCatCols[,1], type = "bar") %>%
    add_trace(y = finalCatCols[,2], name = "y", z = finalCatCols
       [,3], , name = "z") %>%
      layout(yaxis = list(title = "Count"),
        xaxis = list(title = xTitle),
        barmode = group/stack)
})
```

4.3.4 3D Scatterplot

The 3D scatter plot uses the X-Y-Z grid to display trivariate points. The “plotly” package in R can be used to create 3D scatter plots. The default function"`plot_ly`" takes the dataframe and the 3 numerical axes to plot the points in an X-Y-Z plane. It is then pipelined with the "`add_markers`" function, which takes the 4th variable to set as the color of the points. The color scaling can also be done in this function. The final pipelined function is the "`layout`" function, which consists of the axis labels, width, height, margin, and annotations. The values are set by default, and it can be customized easily.

```
library(plotly)
plot_ly(finalNumCols, x = finalNumCols[,1], y = finalNumCols[,2], z
    = finalNumCols[,3])%>%
    add_markers(marker = list(color = finalCatCols[,4], colorscale
       = c('#FFE1A1', '#683531'), showscale = TRUE)) %>%
        layout(scene = list(xaxis = list(title = xTitle),
                            yaxis = list(title = yTitle),
                            zaxis = list(title = zTitle)),
                annotations = list( x = 1.13, y = 1.05,
                                    text=colorText,
                                     xref = '',
                                     yref = '',
                                     showarrow = FALSE
                ))
})
```

4.4 User Interface

An interactive UI is designed to carry out the processes and visualize the results. A good User Interface Design presents a seamless blend of visual design, interaction design, and information architecture. In our thesis, an interactive UI is designed in the form of a dashboard. We have used a couple of R packages shinydashboard [16] and dashboardthemes [17] to design the UI. Shinydashboard is made on top of another package called "shiny", which is used to build interactive web applications. The shinydashboard has a default layout consisting of the sidebar, header, and the body. In our approach, the sidebar panel contains various operation links such as selecting the dataset, visualizing the cluster graph, results of feature selection, and visualizing them using different modes. The results of the operations are displayed in the dashboard body. The dashboardtheme package is used to customize the appearance of the Shiny dashboard. It provides a number of pre-defined themes that create a decent visual appearance to the dashboard.

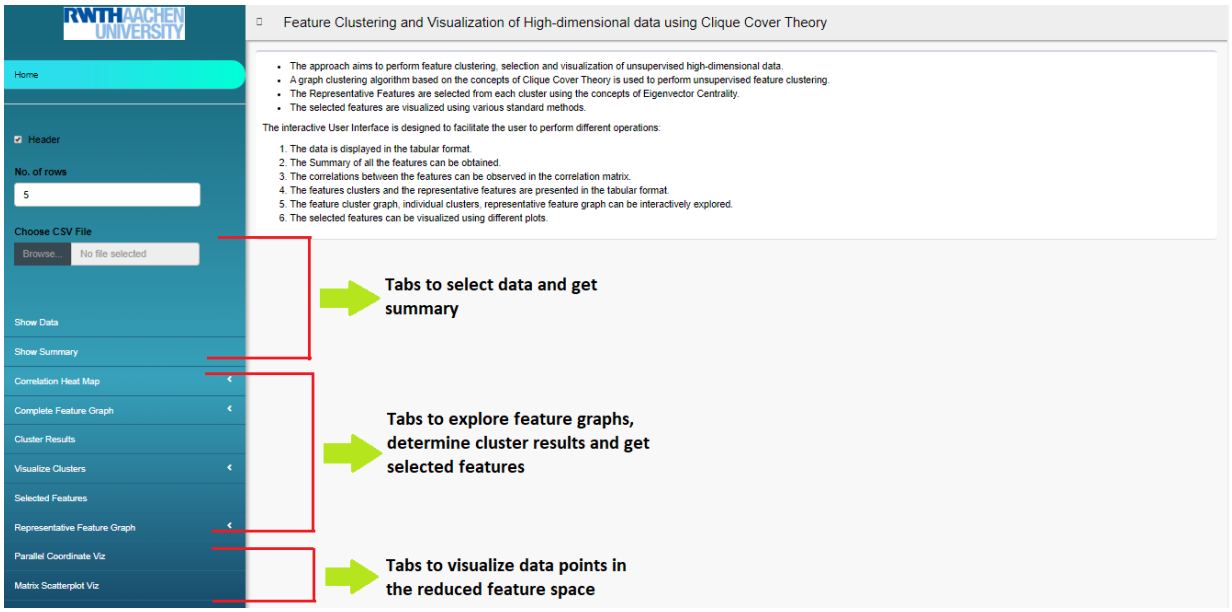


Figure 4.10: The Application Homepage

The shiny application is needed to be split into two separate files, ui.R and server.R. The ui.R contains the code for UI layout like the width of the sidebar, the description of the menu items, the items in the drop-down list, the checkbox and radio button, the size of the dashboardbody, etc. The code of ui.R is automatically parsed to a valid HTML-CSS code by the shiny application at the time of rendering. Each of the menu items or the buttons in UI is linked to a component in server.R, which is responsible for producing the result. The server.R contains all the algorithmic functions described in the processing and

[16] `https://www.rdocumentation.org/packages/shinydashboard/versions/0.7.1`
[17] `https://www.rdocumentation.org/packages/dashboardthemes/versions/1.0.5`

visualization phase and uses those to create the results which are finally rendered by the ui.R. Figure 4.10 shows the User Interface layout designed for our approach.

The UI starts with the home page that describes the operations performed by our approach. The layout is basically divided into 3 parts. The first part facilitates the user to select the dataset, view the data in tabular format, and to get a brief summary of the features in the data. The second part enables the user to explore the features using different feature graphs. The feature correlations can be visualized using correlation heatmap. A package called d3heatmap [18] is used to create the heatmap. The cluster results and the set of Representative Features can be visualized using tabular format and graphical format (Cluster feature graphs and Representative feature graphs). There are options to examine clustered features by visualizing individual clusters separately. The final part facilitates the user to visualize the data points in the reduced feature space corresponding to the Representative Features. The UI has tabs to display the data points using various plots like Parallel Coordinate Plots, Scatterplot Matrix, Grouped and Stacked Bar Charts, etc. The interactivity like zoom, brush, tooltips is enabled for all the plots by setting the parameters as described in section 4.3.

All the feature graphs are created using a package called VisNetwork [19]. It is a network visualization package created using vis.js, a javascript library. It provides various options to customize the graph like adding color, shape, opacity of the nodes, adjusting the thickness of the edges with respect to the values, adding tooltips on the nodes and edges, customizing legends, zoom and drag functionality.

The following figures show different stages of the User Interface. The correlation heatmap between the categorical features of the automobile dataset is shown in Figure 4.11. The Cluster feature graph and the Representative feature graph in the UI layout are shown in Figure 4.12 and 4.13, respectively. The Parallel Coordinate Plot in Figure 4.14 shows the data points in the reduced feature space.

[18] `https://www.rdocumentation.org/packages/d3heatmap/versions/0.6.1.2`
[19] `https://datastorm-open.github.io/visNetwork/`

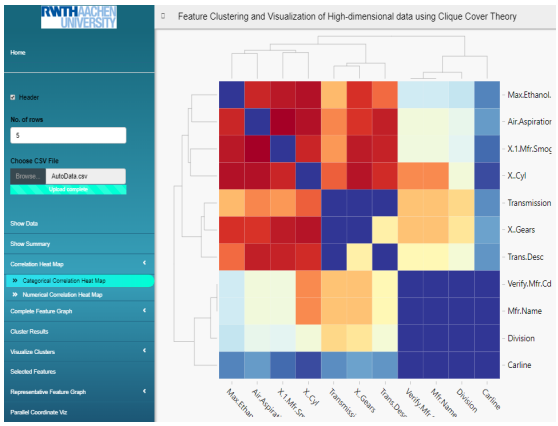


Figure 4.11: Categorical Correlation HeatMmap in UI layout

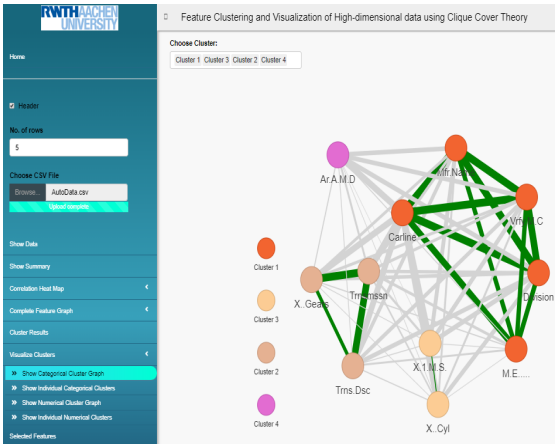


Figure 4.12: Cluster Feature Graph in UI layout

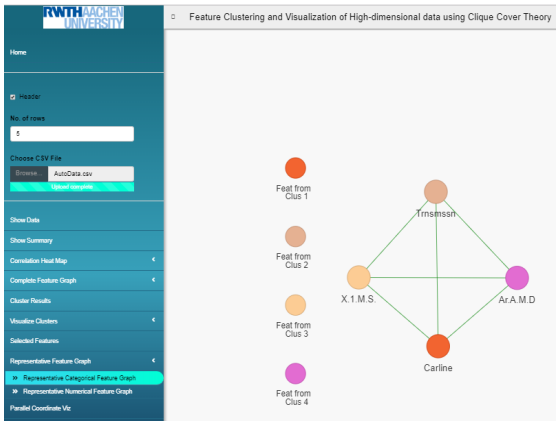


Figure 4.13: Representative Feature Graph in UI layout

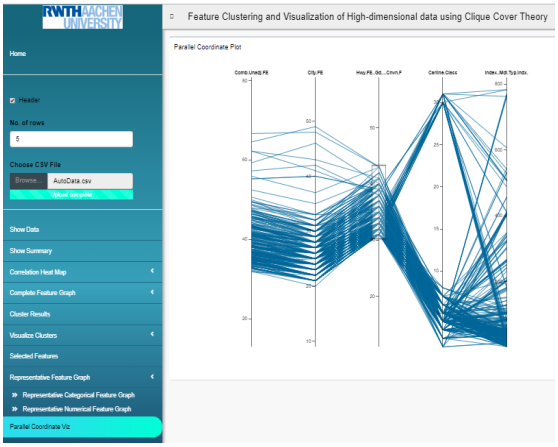


Figure 4.14: Parallel Coordinate Plot in UI layout

Table 4.1 shows the list and description of R packages used in the implementation of our proposed approach.

R Packages List	
R Package	**Description**
tidyverse	Preparing, Wrangling and Visualizing Data
dplyr	Data Transformation
janitor	Remove all empty or NA rows and column
base	Remove special characters and trim whitespace
stringr	String processing
missRanger	Fast Imputation of missing values
minerva	To determine Maximal Information Coefficient
parallel	Used for parallel computing
MASS	Used to create contingency Table and calculate Chi-square test
DescTools	Used to pairwise determine Cramer's V
igraph	Generate graph objects, determine cliques and maximal cliques
parcoords	Generate interactive Parallel Coordinate
pairsD3	Generate Scatterplot Matrix
plotly	Generate Grouped/Stacked Bar plots and 3D Scatterplot
Shinydashboard	Set the layout of dashboard
dashboardthemes	Customize the appearance of the Shiny dashboard
d3heatmap	Create correlation heatmap
visNetwork	Used for Graph Visualization

Table 4.1: The R Packages List used for Implementation

Chapter 5

Evaluation

In this section, our proposed approach is evaluated with respect to different existing approaches. We have considered several datasets and evaluation metrics to carry out the performance analysis. The results in the form of tables and graphs are presented in this chapter.

5.1 Experimental Setup

We have used a machine learning software, known as Weka [78], to carry out the evaluation. It is free software licensed under the GNU General Public License. It has a collection of machine learning algorithms written in Java for data mining tasks. The algorithms can either be applied directly to a dataset or called from the Java code. Weka contains tools for data pre-processing, classification, regression, clustering, feature selection, association rules, and visualization [1]. Its main user interface is the Explorer. The interface provides several panels to select and pre-process the dataset, apply different classification and clustering algorithms, visualize the results in different formats.

In our experiments, we have used the Weka version 3.6 [2]. It requires a Java environment to be set up. We have installed the Java version 1.8.0_211 on our host system in order to run the Weka. The proposed approach is implemented in R version 3.4.3. The host is a Windows 10 system having an Intel Core i5 processor with a collection of 4 cores clocking at 2.20GHz.

5.2 Selection of Datasets

For the evaluation, we have considered high-dimensional datasets from various categories having different numbers of features. The datasets also have different aspects like binary class, multi-class, missing values, and skewed classes. It enables us to perform the stress

[1] https://towardsdatascience.com/data-mining-tools-f701645e0f4c
[2] https://www.cs.waikato.ac.nz/ml/weka/

test of our approach corresponding to other existing approaches. The datasets are collected from open source repositories like UCI Machine Learning [3], kaggle [4], and openml [5].

For the quantitative evaluation, the supervised datasets are selected because the class labels are needed to evaluate the classification and clustering accuracy, and also for the cost-sensitive analysis. The following steps are taken to create the data model for the evaluation.

- Since our approach is unsupervised, we have removed the class labels from the selected datasets.
- Our proposed approach and the existing feature selection methods are applied to the unsupervised datasets.
- The class labels are then appended to the results obtained from each of the feature selection approaches.
- The supervised reduced feature sets thus obtained are then used for quantitative evaluation.

The following table shows the list of the selected datasets used for evaluation.

No	**Dataset Description**	**No. of Features**	**No. of Classes**	**Missing Values**	**Source**
1	Automobile Dataset	25	2	Yes	UCI ML
2	Breast Cancer Dataset	30	2 (Skewed Classes)	No	UCI ML
3	AutoUniv Dataset	39	8	No	UCI ML
4	QSAR-Biodegradation DataSet	41	2	Yes	openml
5	Sonar Dataset	60	2	No	Kaggle
6	Emotions Dataset	78	2	No	openml
7	Robot-Failures Dataset	91	5	No	openml
8	Spectrometer	125	2 (Skewed Classes)	Yes	openml
9	Musk	168	2	No	Kaggle

Table 5.1: Selected Datasets for Evaluation

[3] http://archive.ics.uci.edu/ml/datasets.php
[4] https://www.kaggle.com/datasets
[5] https://www.openml.org/

5.3 Existing Approaches

Our proposed approach performs unsupervised feature clustering and selection. Therefore, we have selected 5 existing unsupervised methods and evaluated our approach with respect to them. The description of some of the existing methods is mentioned in Section 2.4. The following are the 5 existing methods.

1. Laplacian Score for Feature Selection [5].
2. Spectral Feature Selection for Supervised and Unsupervised Learning [6].
3. l2,1-Norm Regularized Discriminative Feature Selection for Unsupervised Learning(UDFS) [79].
4. Unsupervised Feature Selection Using Nonnegative Spectral Analysis(NDFS) [80].
5. Unsupervised feature selection for multi-cluster data(MCFS) [7].

5.4 Evaluation Metrics

5.4.1 Quantitative Evaluation

The reduced feature sets obtained from each approach are quantitatively evaluated using the following metrics.

- **Evaluation using Classification Accuracy** - The accuracy of the reduced feature sets are evaluated using classifiers- Naive Bayes [6], Support Vector Machine (SVM) [7], Random Forests [8] and Logistic Regressions [9]. The K-fold cross validation [10] is used to evaluate the classifiers. The evaluation is carried out in the Weka Software corresponding to above mentioned classifiers and the value of K is set to 10 in the K-fold cross validation.
- **Evaluation using Clustering Accuracy** - The accuracy of the reduced feature sets are evaluated using 2 clustering algorithms; K-means clustering [11] and Expectation Maximization clustering [81]. The Clustering Accuracy (ACC) metric is used for assessing the clustering quality. The number of clusters is set to the number of classes present in the respective datasets. The Weka software has the meta panel known as "ClassificationViaClustering" to perform such measures. The K-fold cross-validation is again used, and the value of K is set to 10.

[6] http://www.statsoft.com/textbook/naive-bayes-classifier
[7] http://www.statsoft.com/textbook/support-vector-machines
[8] https://towardsdatascience.com/understanding-random-forest-58381e0602d2
[9] https://machinelearningmastery.com/logistic-regression-for-machine-learning/
[10] https://machinelearningmastery.com/k-fold-cross-validation/
[11] https://stanford.edu/~cpiech/cs221/handouts/kmeans.html

- **Evaluation using ROC Curves for cost-sensitive analysis** - AUC - ROC curve is a performance measurement for the classifier to represent the degree or measure of separability. It tells how much model is capable of distinguishing between classes. We have used Area Under The Curve (AUC) on the Receiver Operating Characteristics (ROC) [12] curve for cost-sensitive analysis. It is used to perform the sensitivity analysis when the class labels are skewed. For example, consider a reduced dataset where 90% of the class labels are positive, and the remaining 10% are negative. Even a bad classifier that predicts every instance as positive will also have at least 90% accuracy. To overcome this, the ROC curve plots the false positive rate with respect to the true positive rate and determines the Area Under the Curve (AUC). The higher AUC signifies the better performance of the classifier corresponding to relevant features in the dataset. We have considered all the classifiers mentioned above and plotted the ROC curve for each of the reduced feature sets obtained from different approaches.

- **Evaluation in terms of the redundancy of the selected features** - We have used "Representation Entropy" [4] as a metric to evaluate the redundancy of the selected features. The class labels are not needed to determine the Representation Entropy. Let the eigenvalues of the $d \times d$ covariance matrix of a feature set of size d be $\lambda_j, j = 1...d$. Let

$$\tilde{\lambda_j} = \frac{\lambda_j}{\sum_{j=1}^{d} \lambda_j}$$

$\tilde{\lambda}$ has similar properties like probability namely, $0 \leq \tilde{\lambda_j} \leq 1$ and $\sum_{j=1}^{d} \tilde{\lambda_j}$. The entropy function is defined

$$H_R = \sum_{j=1}^{d} \tilde{\lambda_j} log \tilde{\lambda_j}$$

The above measure is known as Representation Entropy. The function H_R attains a minimum value when all the information is present along a single coordinate direction. The value of the function is maximum when the information is equally distributed among the features, and so is the uncertainty involved in feature selection. The Representation Entropy is the property of the dataset as represented by the particular set of features and is a measure of the amount of information compression possible by dimensionality reduction. This is equivalent to the amount of redundancy present in the reduced feature set. It is expected that the entropy of individual clusters should be low, whereas the final reduced feature set should have low redundancy, i.e., a high value of Representation Entropy.

[12] `https://www.medcalc.org/manual/roc-curves.php`

- **Friedman Test** - It is a non-parametric statistical test used to detect differences in treatments across multiple test attempts [13]. We have used an ensemble learning method based on Voting Classifier to carry out the Friedman Test. The reduced feature sets obtained from the different approaches are scored using the Voting Classifier. The Friedman test ranks the scores, checks for the statistical difference, and determines which results are better based on the scores. The results of the Friedman test is discussed in Section 5.6.3.

For describing the results of the quantitative evaluation, we have used a couple of datasets from the table of selected datasets (Table 5.1). The evaluation metrics are explained with respect to the results and is given in Section 5.6.1 and 5.6.2.

The proposed approach doesn't require any parameter tuning like estimating the number of clusters or classes. The selected features are calculated based on the intrinsic properties of the data. The existing approaches, however, require some parameter tuning to yield the results. We have manually set the required parameters for some of the existing approaches. The Laplacian Feature Selection method and the Spectral Feature Selection method doesn't require any parameter tuning; however, they rank all the features separately. We have selected the top K features from both the approaches corresponding to the number of features returned by our proposed approach. The UDFS and NDFS methods require the number of clusters in advance before feature selection. We have put the number of classes present in the data as the number of clusters for optimal results. The MCFS method requires the number of clusters and the number of final features to be selected. The clusters are given corresponding to the classes in the data, and the number of final features is set corresponding to the number of features returned by our proposed approach.

5.4.2 Qualitative Evaluation

The qualitative evaluation is more subjective to quantitative evaluation. It is defined in science as any observation made using the five senses. In research and business, qualitative evaluations may involve value judgments and emotional responses [14]. In our thesis, the qualitative evaluation is performed by the visualization method. We have used the Parallel Coordinate plot as the primary method to observe clutter in different cases. Since plotting a Parallel Coordinate with many dimensions is clumsy, we have used a couple of datasets with relatively less number of features. The two selected datasets are namely; Heart dataset having 13 features and Australian Credit Approval dataset having 14 features. The following steps are used to perform the evaluation :

- The original dataset is plotted using Parallel Coordinate Plot without performing any dimensionality reduction.
- The unsupervised feature selection is carried out using the Laplacian Feature Selection method and our proposed approach.

[13] `https://www.statisticshowto.datasciencecentral.com/friedmans-test/`
[14] `https://urlzs.com/E2GQH`

- The reduced feature sets from both the methods are also plotted using Parallel Coordinate Plot and compared with each other and also with the original plot to observe the clutter present.

5.5 Computational Complexity

The computational complexity is calculated after the ingestion and the feature correlation phase. We have determined the computational complexity from the Complete feature graph (Figure 4.6) until we obtain the Representative feature graph (Figure 4.9). Our proposed approach first performs the unsupervised feature clustering on the entire feature set. The recursive process of determining the feature clusters mainly depends on three steps; identifying the threshold coefficient using K-NN method, maximal clique determination, and finding the weight of each maximal clique. In our case, the complexity of identifying the threshold coefficient depends on the number of nodes in the feature graph and the value of k. The complexity is given as O(kn). In [82], the authors have determined the complexity of maximal clique determination and is equal to $O(2^{n/3})$ or $O(1.2599^n)$, where n is the number of nodes. The complexity of finding the weights of each of the maximal clique depends on the number of edges, O(e), in the maximal clique. After the feature clusters are determined, the algorithm performs Representative Feature selection from each cluster based on the Eigenvector Centrality. The complexity of determining Eigenvector Centrality is O(qE), where q is the number of iterations needed before convergence, and E is the number of edges in each cluster [15]. The combined computational complexity for feature clustering and selection can be written as :

$$[O(kn) + O(1.2599^n) + (p \times O(e))] + O(qE)$$

where n is the total number of features in the feature graph, k is the number of nearest neighbors, p is the number of intermediate maximal cliques obtained, e is the number of edges in each maximal clique, q is the number of iterations required to determine Eigenvector Centrality and E is the number of edges in the cluster.

The computational function is recursively applied the number of times until all the clusters are formed. The core complexity of the problem or the step taking the maximum time is the exponential function to determine maximal cliques in the graph. It suggests that the time complexity to determine maximal cliques increases exponentially with the number of features. In other words, we can say that as the number of features in the complete feature graph increases, the time to determine maximal cliques increases exponentially. To justify this, we have calculated the time taken to process the feature graph and determine the Representative Features. We have selected 8 unsupervised datasets in different feature ranges to determine the time complexity. Table 5.2 shows the time required to process different datasets.

A line chart between the number of features and the time is shown in Figure 5.1. As seen in the figure, the time taken increases exponentially with respect to the number of features

[15] `https://urlzs.com/NyciK`

Dataset Name	**Dimensions**	**Time (in seconds)**
Automobile Dataset	25	0.10
QSAR Biodegradation Dataset	41	0.16
Emotions Dataset	78	0.97
Robot Failure Dataset	91	1.08
Yeast Dataset	116	2.42
Musk Dataset	168	11.61
Arrhythmia Dataset	280	22.52
Airline Ticket Price Dataset	417	43.59

Table 5.2: Time Measured on Different Datasets

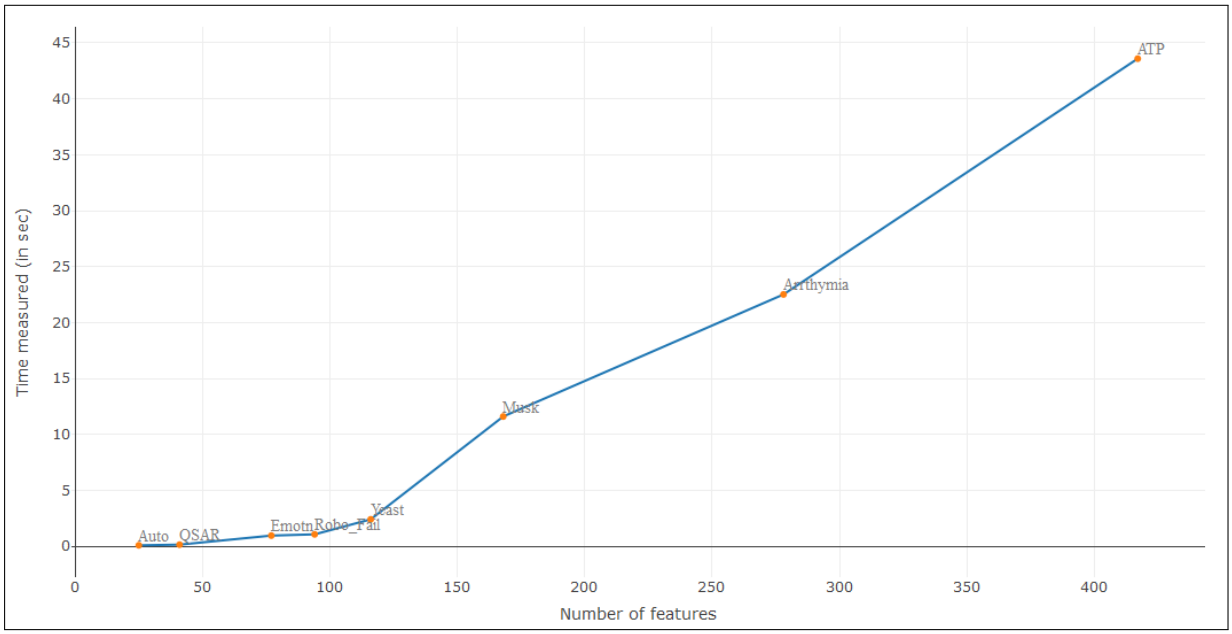


Figure 5.1: Time Complexity Analysis

in the dataset. It shows that the computational complexity of the proposed algorithm is of the exponential form. On analyzing the time taken for processing the features, the total computational complexity of the algorithm is found to be $O(1.38^n)$.

5.6 Quantitative Evaluation Results

In this section, the evaluation results with respect to the classification and clustering accuracy are presented in the form of grouped bar charts. The ROC curves are plotted side-by-side for each of the feature selection methods. The Representation Entropy is shown in the tabular format. We have discussed the evaluation results of two datasets; The Musk Dataset and The Robot-Failures Dataset. The results of the remaining datasets are given in Appendix A.2.

5.6.1 The Musk Dataset

The Musk dataset has 168 features and 477 instances. The dataset has 2 classes that are evenly distributed. Our proposed approach has selected 32 features from the Musk dataset. The quantitative evaluation results are shown below.

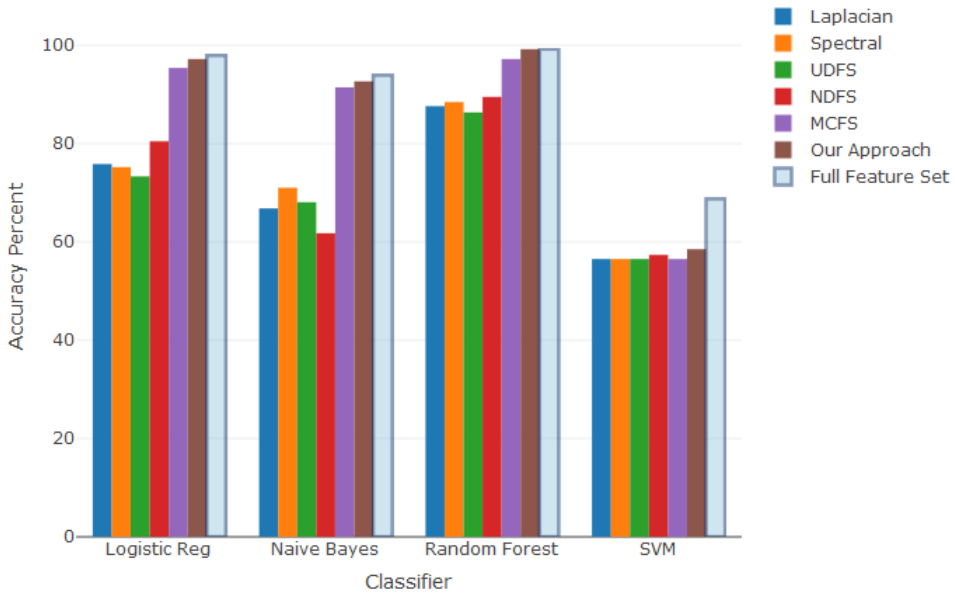


Figure 5.2: Classification Accuracy (Musk Dataset)

Figure 5.2 shows the comparison of classification accuracy of different feature selection methods. The data points corresponding to the reduced feature space obtained from each of the feature selection methods are divided into the train, and test sets, and the classification accuracy is determined using the 4 mentioned classifiers. The accuracy of the reduced feature space from our proposed approach has shown relatively better results in all the classifiers. The accuracy using the Random Forest classifier is the highest, whereas the accuracy using the SVM classifier is the lowest. The reason for showing the good result is that the dataset has a balanced proportion of categorical and numerical features. Since our approach segregates the features and applies feature clustering and selection to the

respective feature groups, the results obtained are relatively better. The figure also shows the accuracy when the classification is performed on the full feature set. As seen, the classification accuracy is more on the full feature set, but the relative accuracy difference is low when compared with our approach.

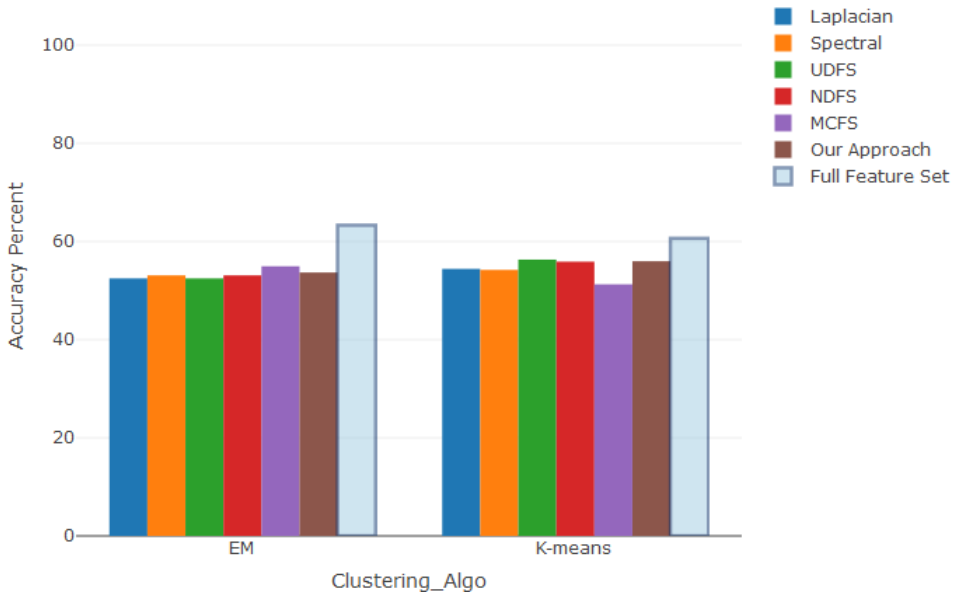


Figure 5.3: Clustering Accuracy (Musk Dataset)

Figure 5.3 shows the comparison of the clustering accuracy of different feature selection methods. Similar to determining the classification accuracy, the dataset is divided in the train, and test sets and the clustering accuracy are measured using 2 clustering algorithms; Expectation Maximization and K-means Clustering. Although the overall clustering accuracy is low as compared to the classification accuracy, the relative performance of our approach is good. Since we don't know the number of clusters in advance, we have assigned the number of classes as the number of clusters. The low accuracy could be because the number of clusters in the data are different from what we have assigned. The relatively good performance of our proposed approach is due to the balanced nature of the dataset, as described above. As seen, the accuracy using the full feature set is more compared to the reduced feature sets.

Figures 5.4 and 5.5 show the ROC curves of the reduced feature sets measured using different classifiers. The ROC curve is used to measure class separability. The higher Area Under The Curve (AUC), the better the performance of the model. In other words, higher the AUC, better the model is at predicting 0s as 0s and 1s as 1s. The maximum value of AUC is 1, and the dotted orange diagonal line denotes the 0.5, i.e., half of the AUC. As seen, the ROC curve plotted using the mentioned classifiers on all the reduced feature sets are beyond the orange line, thereby showing good performance. The ROC curve plotted using the SVM classifier has shown relatively average performance on all the approaches.

The ROC curve plotted using our proposed approach is significantly better compared to other approaches. The ROC curve for Logistic Regression, Naive Bayes, and Random Forest are far beyond the orange line and towards the left-top corner denoting larger AUC.

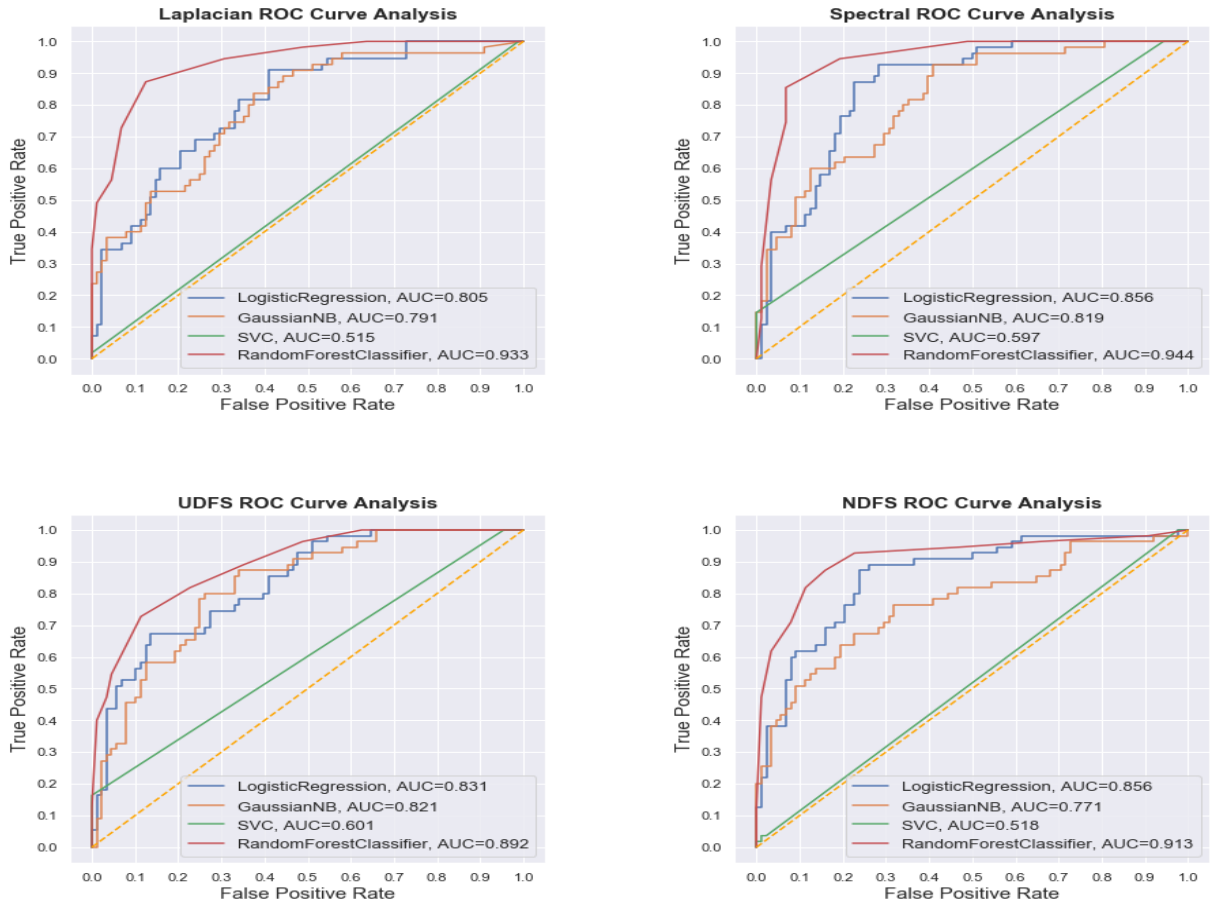


Figure 5.4: ROC Curve (Musk Dataset)(1)

	Laplacian	Spectral	UDFS	NDFS	MCFS	Our Approach
Rep Entropy	0.91315	0.90626	0.84794	0.7526	0.94620	1.0111

Table 5.3: Representation Entropy (Musk Dataset)

Table 5.3 shows the Representation Entropy obtained from the different reduced feature sets. It is a measure of the redundancy rate. The Representation Entropy H_R is calculated using the formula given in section 5.4.1 (Point 4). The Representation Entropy of the features belonging to the same cluster is low because they share the same information. In our proposed approach, we have first performed feature clustering and then selected the Representative feature from each cluster, ensuring that the selected features don't have the same information or possess low redundancy. In other words, the higher the Representation Entropy of the selected features, the lower is the redundancy rate. The Representation Entropy of our proposed approach is higher, meaning the selected features have a relatively low redundancy rate.

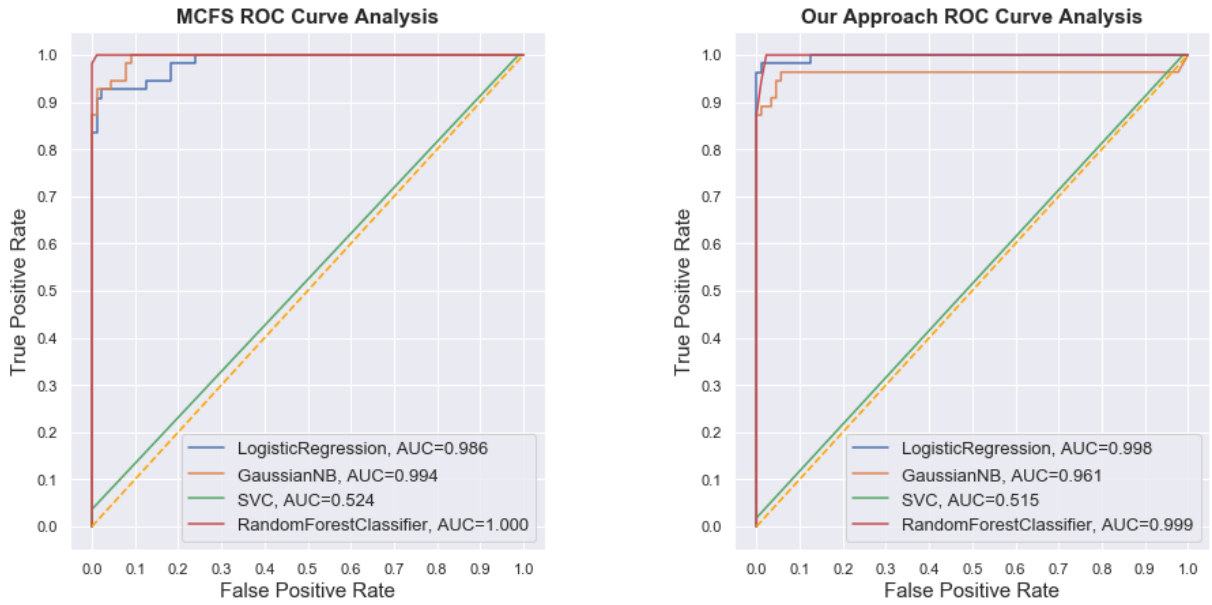


Figure 5.5: ROC Curve (Musk Dataset)(2)

5.6.2 The Robot-Fail Dataset

The dataset has 91 features and 268 instances. It contains force and torque measurements on a robot after failure detection. Each failure is characterized by 15 force/torque samples collected at regular time intervals. It is a multi-class dataset having 5 classes. Our proposed approach has selected 19 features from the Robot-Fail dataset.

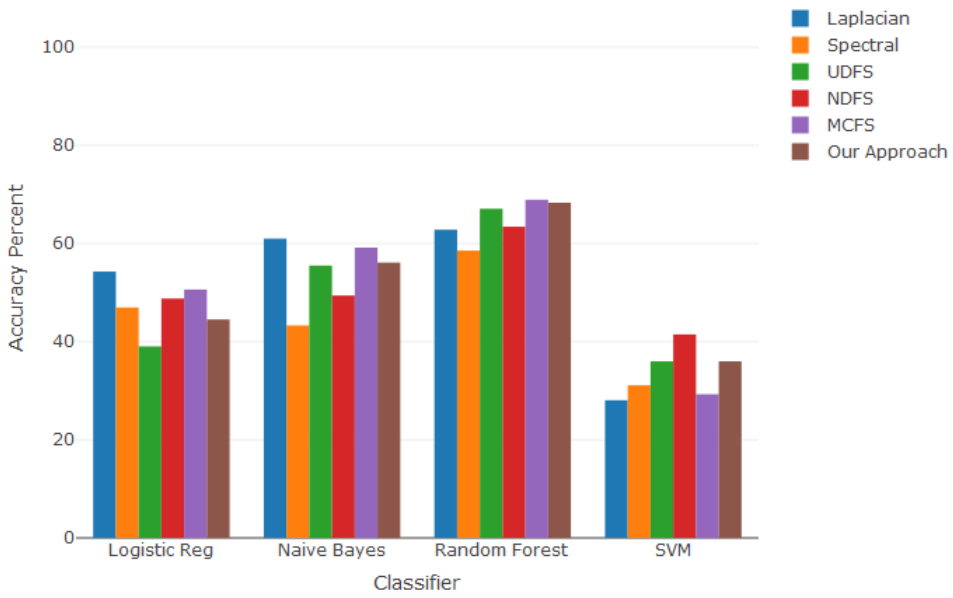


Figure 5.6: Classification Accuracy (Robot-Fail Dataset)

Figure 5.6 shows the comparison of classification accuracy of different feature selection methods using the Robot-Fail dataset. The steps involved in calculating the classification accuracy is the same as mentioned in the Musk dataset. As seen in the figure, the overall accuracy is low on all the reduced sets. The reason being it is a multi-class dataset and the classifier has to classify among 5 classes. As the number of classes increases, the chances of misclassification (error rate) also increases as a result of which the accuracy decreases. The accuracy our proposed approach corresponding to Random Forest has shown relatively good performance, whereas using other classifiers the approach has shown average performance. The feature selection methods that ranks the feature set have the advantage of selecting more features to improve the accuracy. For example, instead of selecting top 19 features using Laplacian method, if we select top 30 features, the accuracy might increase. We cannot set the number of selected features in our proposed approach, so we cannot increase the accuracy in such cases.

Figure 5.7 shows the clustering accuracy of different feature selection methods. The steps to determine accuracy is the same as mentioned in the Musk dataset. As seen, the clustering accuracy is very low on all the reduced feature sets obtained from different approaches. The reason is that it is a multi-class dataset, and we don't know the number

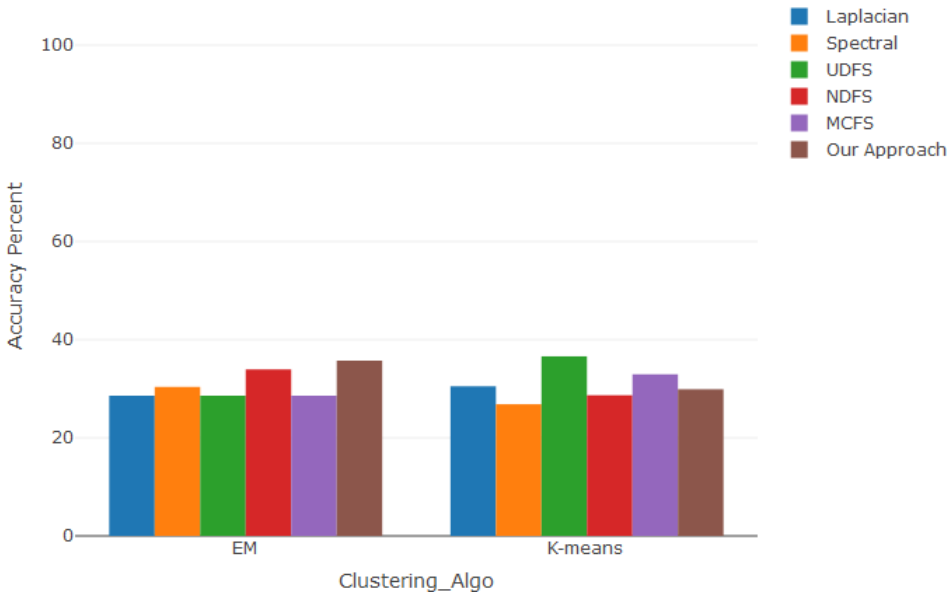


Figure 5.7: Clustering Accuracy (Robot-Fail Dataset)

of actual clusters in advance. We have set the number of clusters as the number of classes present in the data. Since the accuracy is low, we can expect the actual number of clusters to be different from what we have specified. As discussed in the classification accuracy, the ranking feature selection methods may have the advantage of selecting more top-ranked features to increase accuracy. Our proposed approach has a disadvantage since we cannot set the number of final selected features. The algorithm itself determines the number of clusters and so the number of Representative Features.

Figures 5.8 and 5.9 show the ROC curves plotted on the reduced feature sets obtained from different approaches. ROC curves are typically used in binary classification to study the output of a classifier. In order to extend the ROC curve and ROC area to multi-class or multi-label classification, it is necessary to binarize the output and use one verses rest classifiers. For example, the ROC curve is plotted between Class 0 and the rest Classes (Class 1,2,3 and 4). Similarly, the curves are plotted for all the classes. We have used the SVM classifier to generate different multi-class ROC curves. Similar to the way as mentioned in the Musk dataset, the black dotted diagonal line represents 50 percent of the Area Under the Curve (AUC). Although the ROC curves for all the classes on our proposed approach are beyond the black dotted line, the curves are not near the top-left corner, thereby making the AUC around 0.75. The reason for having less AUC is because it is a multi-class dataset, and the chances of predicting 0s with 0s decreases as the number of classes increases.

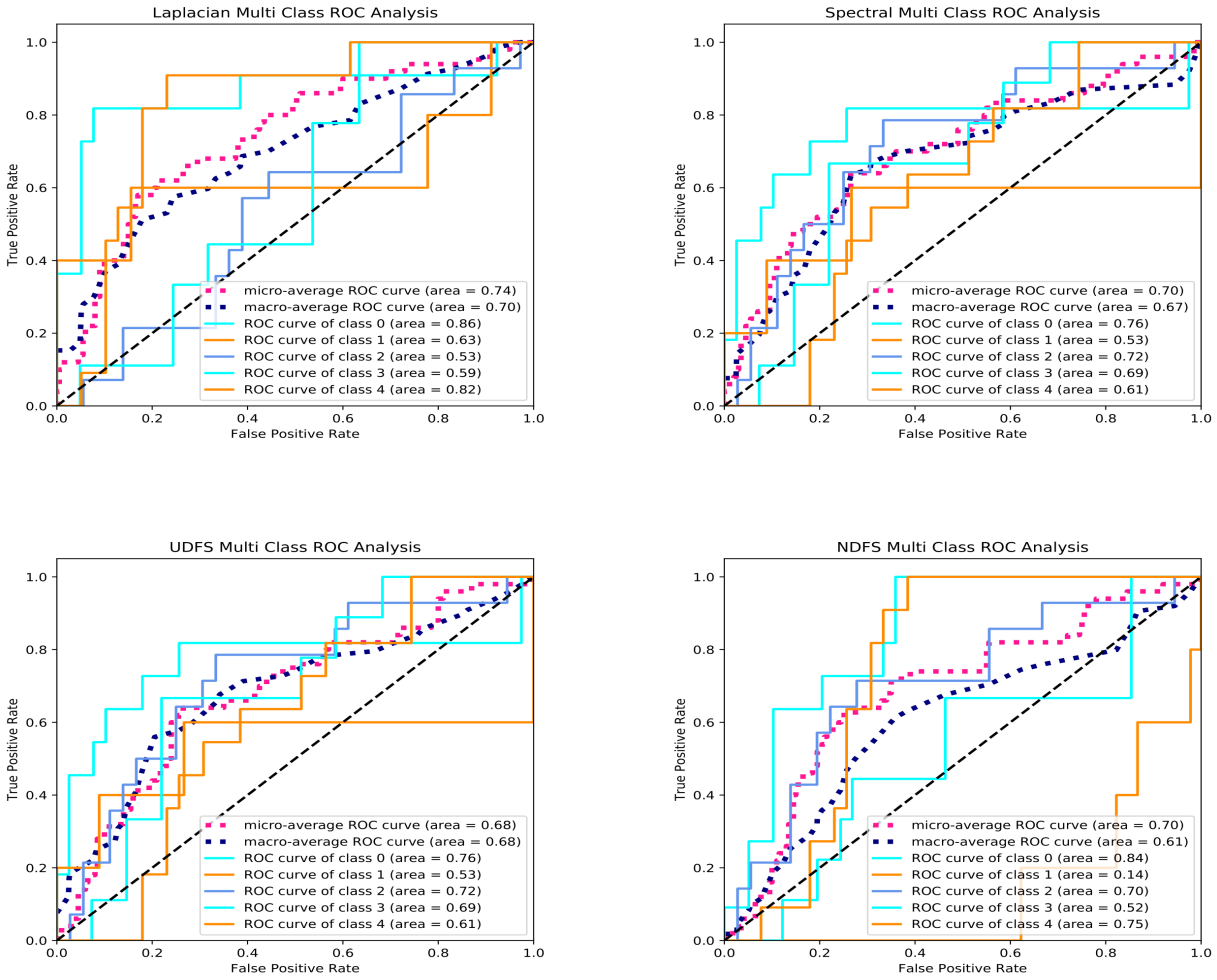


Figure 5.8: ROC Curve using SVM (Robot-Fail Dataset)(1)

	Laplacian	**Spectral**	**UDFS**	**NDFS**	**MCFS**	**Our Approach**
Rep Entropy	0.38221	0.87985	0.54304	0.92084	0.29013	0.44478

Table 5.4: Representation Entropy (Robot-Fail Dataset)

Table 5.4 shows the Representation Entropy obtained from the reduced feature sets of different approaches. The function to determine the Representation Entropy is the same as discussed in the Musk dataset. The entropy obtained from our approach is average. It means that the Representative Feature selected from each cluster in our approach may have a strong correlation with each other. Due to the presence of correlation, the selected features may share the same information as a result of which the entropy is low, and the redundancy is high. In other words, we can say that the redundancy rate of the selected features is average using our approach.

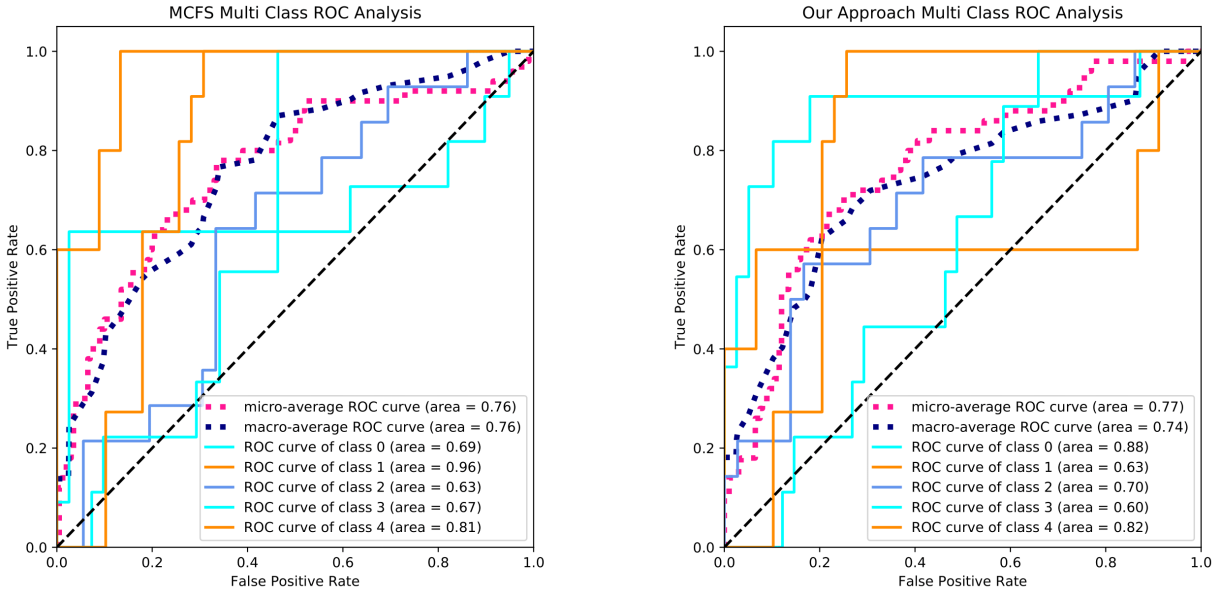


Figure 5.9: ROC Curve using SVM (Robot-Fail Dataset)(2)

5.6.3 Friedman Test

The Friedman test is the non-parametric alternative to the one-way ANOVA with repeated measures. It is used to test for differences between groups across multiple test attempts [16]. The idea is to verify if there is a significant difference between blocks with a fast multiple comparison method. Non-parametric means the test doesn't assume the data comes from the normal distribution.

Dataset	Laplacian	Spectral	UDFS	NDFS	MCFS	Our Approach
Autos	0.8064	0.6935	0.7258	0.8225	0.7419	0.7903
Brst_Cncr	0.9649	0.9005	0.8421	0.9415	0.8362	0.9532
QSAR_bio	0.8138	0.7791	0.8296	0.8454	0.8264	0.8801
Sonar	0.4603	0.5079	0.5873	0.6507	0.4761	0.7619
Emotions	0.8314	0.8314	0.8595	0.7977	0.6853	0.8314
Robot_Fl	0.68	0.54	0.5	0.48	0.64	0.66
Specmtr	0.925	0.93125	0.95	0.9125	0.975	0.95625
Musk	0.8601	0.8531	0.8391	0.8881	0.9930	0.9998

Table 5.5: Voting Classifier Scores

To perform the Friedman Test, we have created a score table (Table 5.5) for each of the feature selection methods corresponding to the selected datasets. The score is obtained by using the ensemble learning method on the reduced feature sets. Ensemble learning uses multiple machine learning models to try to make better predictions on a dataset. An ensemble model works by training different models on a dataset and having each model make predictions individually. The predictions of these models are then combined in the ensemble model to make a final prediction [17]. Since every model has its strengths and weaknesses, therefore, ensemble methods can be beneficial by combining individual models. We have used Voting Classifier in which the ensemble model makes the prediction by majority vote. We have used three different estimators to put into our Voting Classifier: K-Nearest Neighbors, Random Forest, and Logistic Regression. The reduced feature sets are divided into train and test data with stratification so that the proportion of the classes is maintained in the sample. Grid search [18] is used to find the optimal hyperparameters for each estimator. It works by training the estimator multiple times on a range of parameters.

[16] `https://www.statisticshowto.datasciencecentral.com/friedmans-test/`
[17] `https://towardsdatascience.com/ensemble-learning-using-scikit-learn-85c4531ff86a`
[18] `https://scikit-learn.org/stable/modules/grid_search.html`

Dataset	Laplacian	Spectral	UDFS	NDFS	MCFS	Our Approach
Autos	5	1	2	6	3	4
Brst_Cncr	6	3	2	4	1	5
QSAR_bio	2	1	4	5	3	6
Sonar	1	3	4	5	2	6
Emotions	5	5	6	4	3	5
Robot_Fl	6	3	2	1	4	5
Specmtr	2	3	4	1	6	5
Musk	3	2	1	4	5	6
Total	**30**	**21**	**25**	**30**	**27**	**42**

Table 5.6: Friedman Ranks

We have used 5-fold cross-validation to determine the accuracy. The best values of each estimator are ensembled to the Voting Classifier, and the combined score is obtained.

The Friedman test is applied to the score obtained from the Voting Classifier. It is based on the null hypothesis that there is no difference among the results. If the calculated score is greater than the critical value, we can reject the null hypothesis. It starts by ranking each column separately. The smallest score should get a rank of 1, and the largest score should get a rank of 6 (since we have 6 types of feature selection methods). The total of each column is then determined. The Friedman score is calculated using the below formula.

$$FriedmanScore = \frac{12}{N \times k \times (k+1)} \sum R^2 - 3 \times N \times (k+1)$$

where N : the number of datasets, k: the number of feature selection methods, R: The total ranks for each of the six columns.

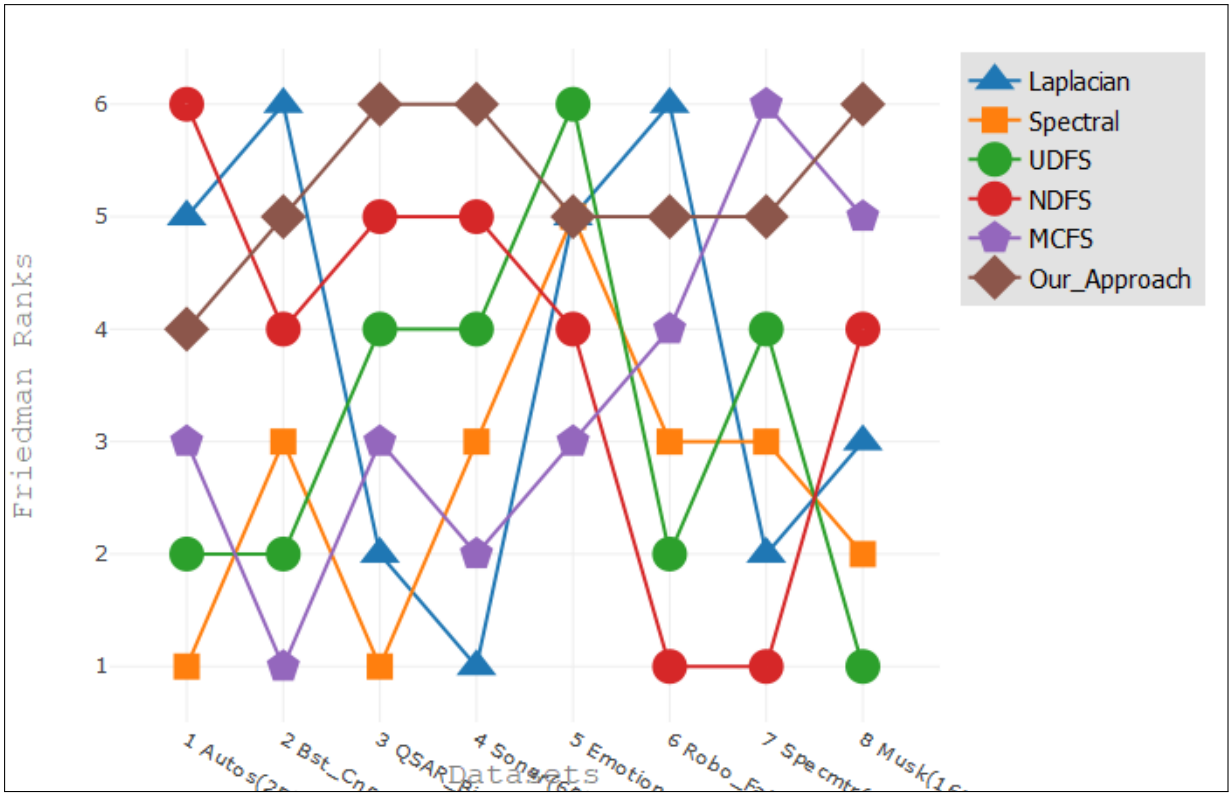


Figure 5.10: Friedman Ranks

Using the scmamp package [19] in R, the Friedman score is computed to 10.851. It has 5 degrees of freedom and a p-value of 0.054. The score is compared from the table of critical values [20]. Using the degree of freedom as 5 an alpha level of 10%, we found that the critical value is 9.236. Since our score is greater than the critical value, we can reject the null hypothesis and conclude that there are statistical differences between the results. To visualize the differences, the average ranks are plotted in the form of multiple line chart, as shown in Figure 5.10.

[19] https://www.rdocumentation.org/packages/scmamp/versions/0.2.55

[20] https://www.statisticshowto.datasciencecentral.com/tables/chi-squared-table-right-tail/

5.6.4 Summary of the Quantitative Evaluation Results

In this section, the quantitative evaluation results on the selected datasets are summarized. The results of 2 datasets; Musk and Robot-fail are discussed in the Section 5.6.1 and 5.6.2. The remaining results are given in Appendix A.2.

- **Datasets having binary class** - The classification and clustering accuracy of the datasets having binary class is relatively good. The ROC curve also has shown good coverage. The redundancy rate is calculated to be below average. The Sonar and the Emotions dataset show these results. The datasets having binary class and a balanced proportion of categorical and numerical feature groups (Musk dataset) have shown even better results.

- **Datasets having multiple classes** - The overall quantitative results are low when dealing with multi-class datasets. The ROC coverage area is also average, and the redundancy rate is high. As the number of classes increases, the overall accuracy decreases. The Auto Univ dataset has 8 classes, and the accuracy percent is low.

- **Datasets having missing values** - The accuracy decreases as the number of missing values in the dataset increases. The Auto dataset has missing values in a couple of features. Although it has a balanced proportion of feature groups, the accuracy is low. We have manually created missing values in the QSAR Biodegradation dataset to compare the effect of missing values with the original dataset. The classification accuracy has dropped with the increase in missing values. The ROC curves have also shown below-average coverage.

- **Datasets with skewed Classes** - The relative accuracy and the ROC coverage is found to be good on the datasets having skewed classes. The Representation Entropy is also average in these cases. The Breast Cancer and the Spectrometer are the two datasets having skewed classes. The Spectrometer datasets have redundant features, as a result of which the classification with full feature set leads to overfitting and show a relative decrease in performance.

5.7 Qualitative Evaluation

In this section, the two selected datasets; Heart and Australian Credit Approval are plotted using Parallel Coordinate Plot. The reduced feature sets from Laplacian method and our approach are also plotted and the observations are described.

5.7.1 Heart Dataset

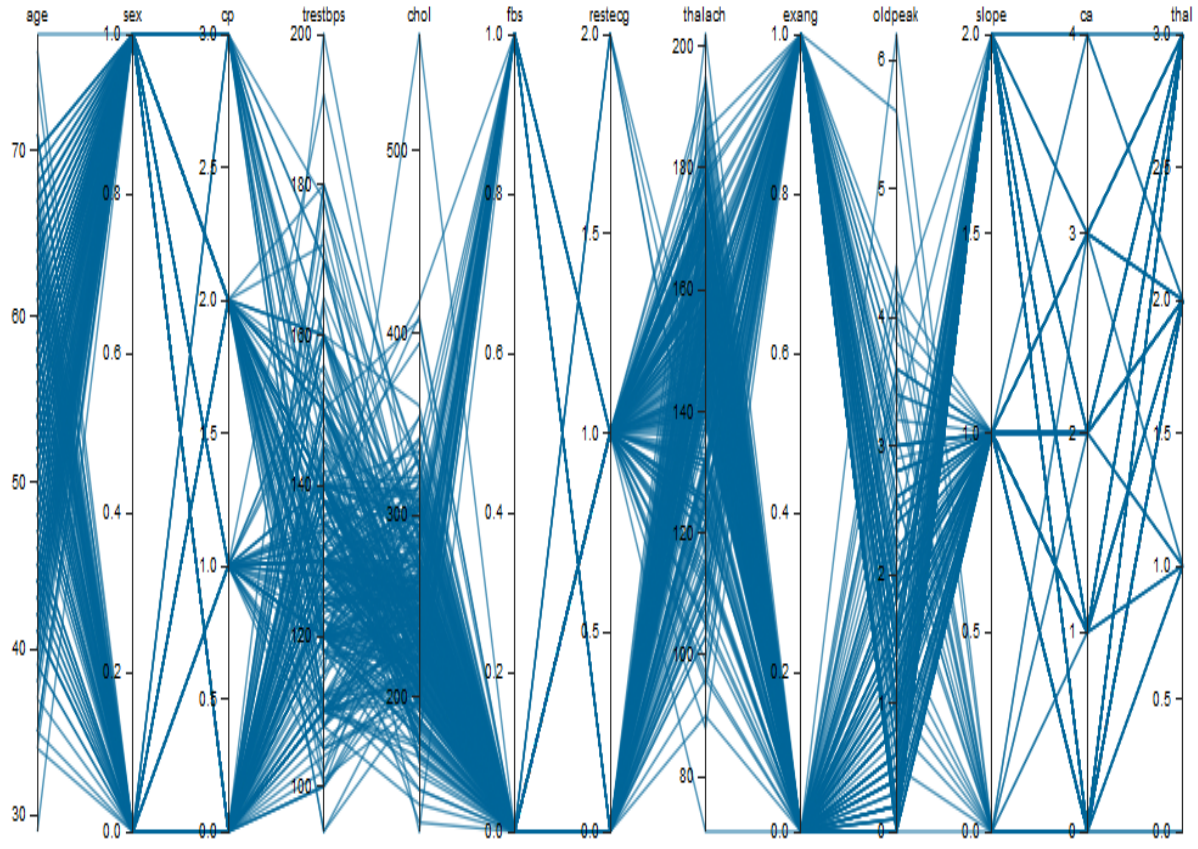


Figure 5.11: Parallel Coordinate Plot with all Features in the Heart Dataset

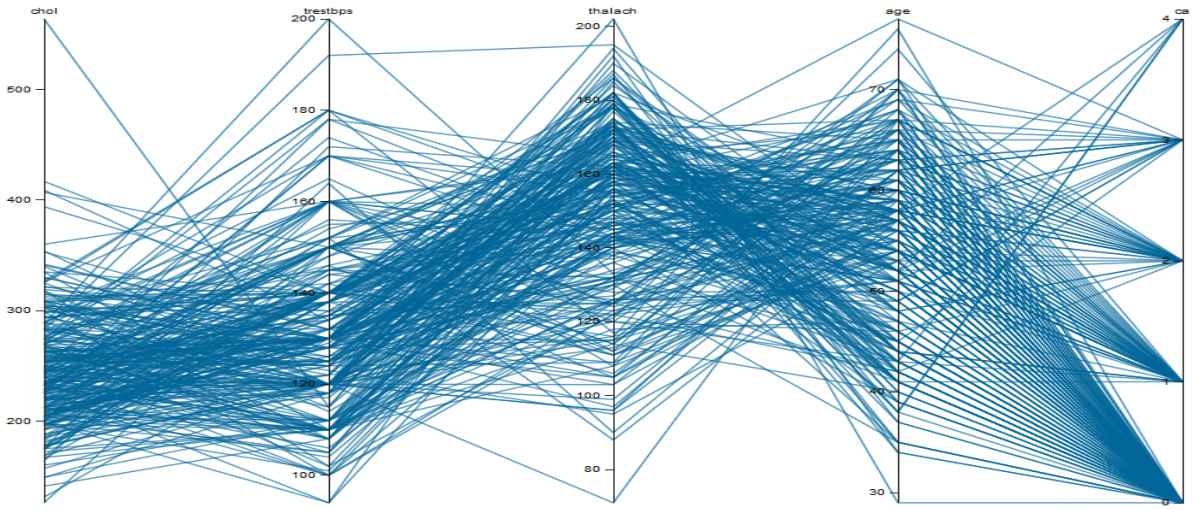


Figure 5.12: Parallel Coordinate Plot with Reduced Feature Set from the Laplacian Method in Heart dataset

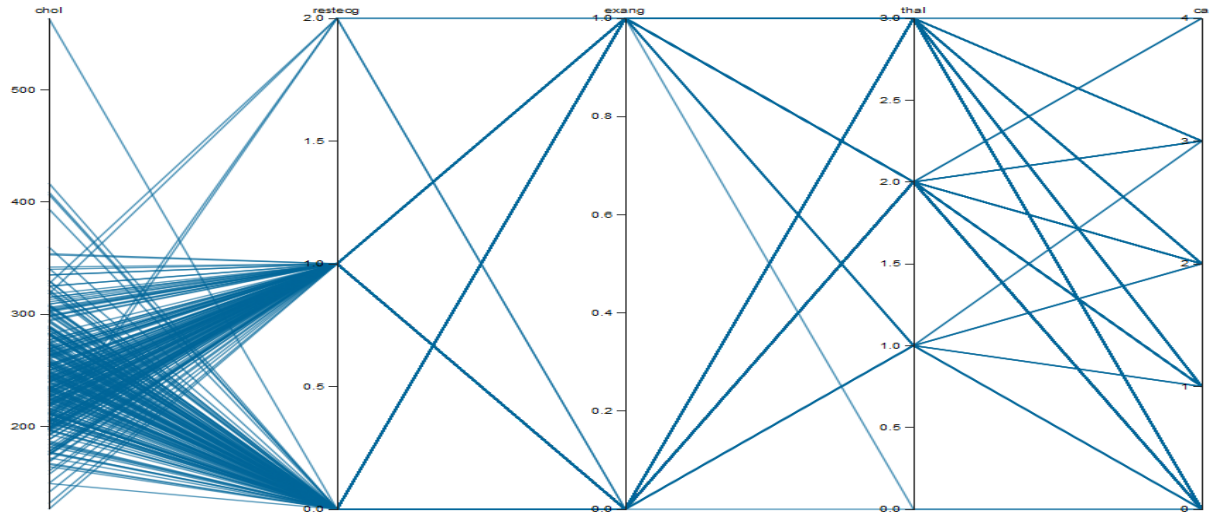


Figure 5.13: Parallel Coordinate Plot with Reduced Feature Set from the Proposed Approach in Heart dataset

5.7.2 Australian Credit Approval Dataset

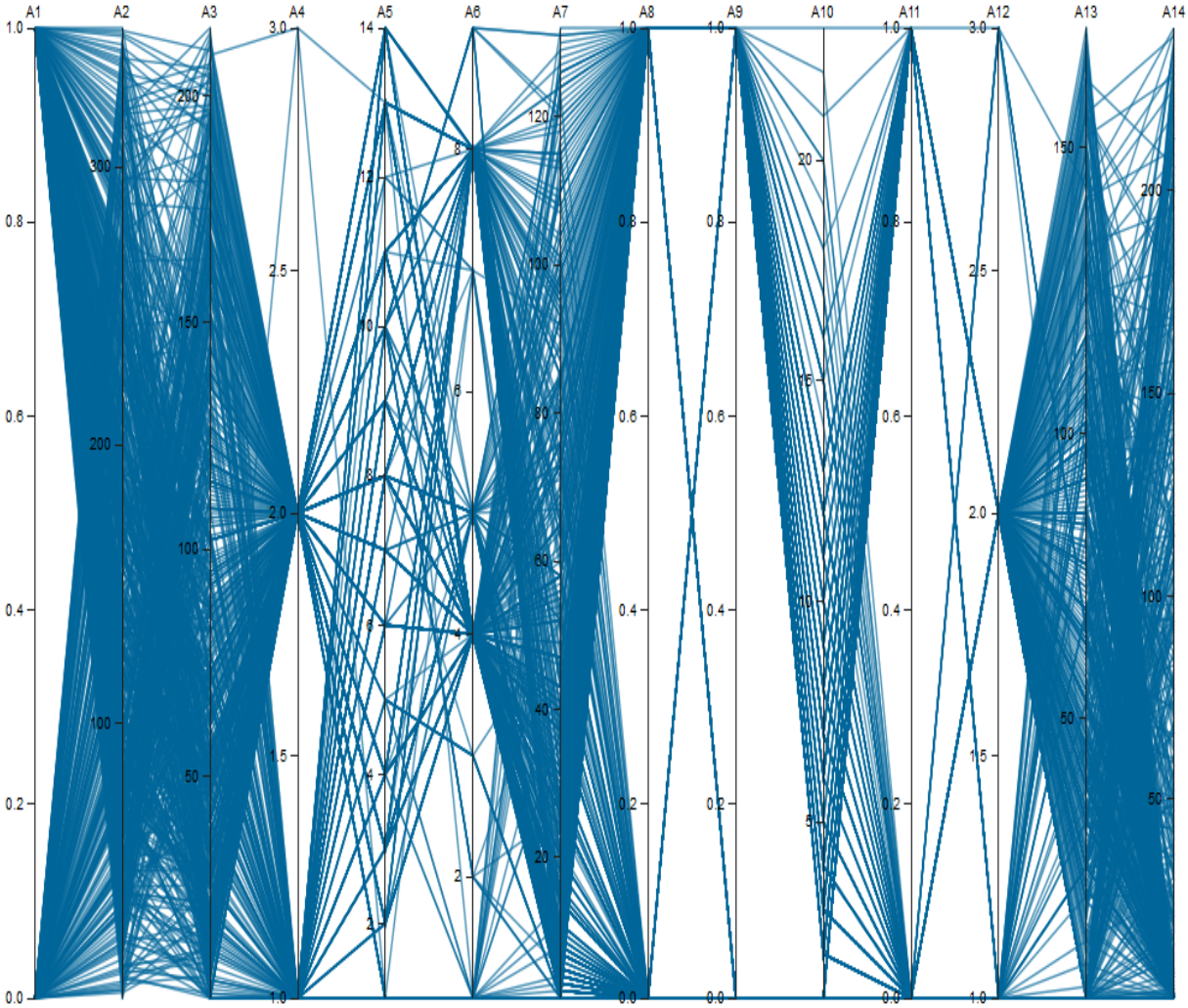


Figure 5.14: Parallel Coordinate Plot with all Features in the Aus. Credit Dataset

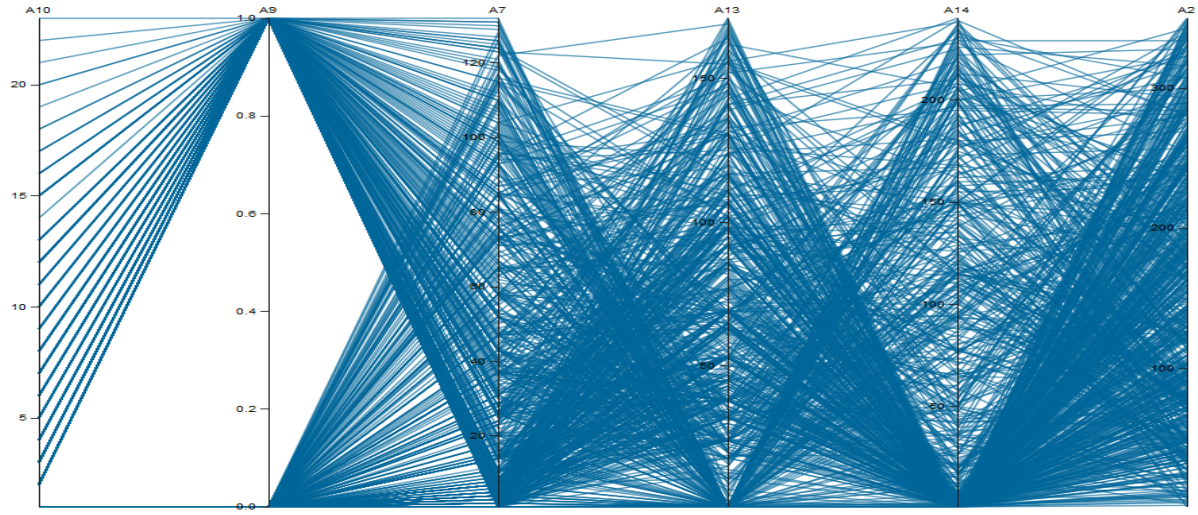


Figure 5.15: Parallel Coordinate Plot with Reduced Feature Set from the Laplacian Method in Aus. Credit dataset

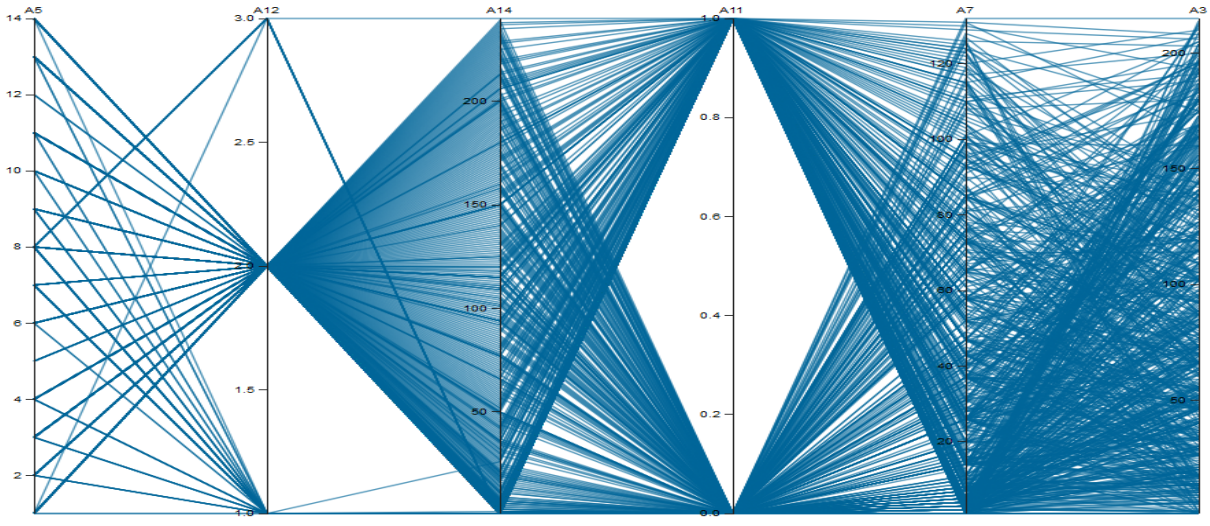


Figure 5.16: Parallel Coordinate Plot with Reduced Feature Set from the Proposed Approach in Aus. Credit dataset

5.7.3 Observations

- The Parallel Coordinate Plot with all the features from the Heart dataset is difficult to interpret because it has a lot of dimensions or axes. The clutter is mainly observed more when the feature has several factors. For example, chol, trestbps, thalach has a lot of clutter because all of them have more than 10 factors. The reduced feature set obtained from the Laplacian method is more clear compared to the full feature set because it has 5 axes. The Parallel Coordinate Plot obtained from our proposed approach also has 5 axes, but it is much clear compared to the previous plots. The reason is that 4 out of the 5 selected features have less number of factors. The proposed approach works by segregating the dataset into categorical and numerical parts based on the factors. Since 9 out of 13 features in the original dataset have a limited number of factors, they are considered as categorical in our approach. The algorithm proceeds by determining the clusters and the Representative Features from each of the categorical and numerical feature groups. As the majority of the features are treated as categorical, the reduced feature set also has more categorical features making the Parallel Coordinate far less cluttered.

- The Parallel Coordinate Plot from all the features in the Australian Credit Approval dataset has a lot of clutter because of many axes and numerical features. The Laplacian method encodes the categorical features to numerical, and the Parallel Coordinate Plot has 6 features. Our proposed approach starts by segregating the dataset. Since the categorical features constitute almost half of the total features, there are 3 categorical and 3 numerical features in the reduced feature set. The clutter observed is a lot less than the previous plots.

Chapter 6

Conclusion

6.1 Summary

In this thesis, an approach to perform unsupervised feature clustering and selection is presented. A novel graph-based clustering algorithm based on Clique Cover Theory is used to perform feature clustering. The number of clusters and the size of each cluster is determined dynamically using the intrinsic properties of the data without any prior estimation from the user. The Representative Features are selected using a graph centrality measure known as Eigenvector Centrality. A User Interface is also designed to explore the feature graphs and to visualize the datapoints of the reduced feature space using standard visualization methods like Parallel Coordinate Plot, Scatterplot Matrix, Group, and Bar charts. The approach is also evaluated on several datasets having a varying number of features and properties. It enables to perform the stress test of the approach at different circumstances. The evaluation is performed using various quantitative and qualitative measures with respect to 5 existing unsupervised methods. The selected features have shown good classification and clustering accuracy in the case of datasets having binary class. The ROC curve has shown good coverage compared to other methods in most of the datasets. The redundancy rate is also observed to be average in most of the datasets. In the qualitative evaluation, the clutter observed is relatively less in Parallel Coordinate Plot using the selected features from our proposed approach. Since the approach segregates the features based on factors, it ensures that there are some categorical features in the reduced feature set. This way, our proposed approach selects features that have good accuracy and also attempts to minimize the clutter. The computational complexity of our proposed algorithm for feature clustering and selection is found to be exponential with respect to the number of features. The overall complexity is measured as $O(1.38^n)$.

6.2 Future Work

Our proposed approach of unsupervised feature clustering is based on the feature graph. The construction of the feature graph depends on the correlations between the features. In

our approach, we have used the correlation measures (MIC and Cramers'V) between every pair of features. An important extension of the correlation is by determining the multiple correlations. It can optimize the process of constructing the feature graph by generating multiple edges simultaneously.

Another aspect where future work can be done is by determining the clusters having a mix of categorical and numerical features. In our approach, we have segregated the datasets, and the clusters are determined in the respective feature groups. We can extend the cluster determination by incorporating a correlation measure to determine the relationship between numerical and categorical features. The feature graph generated this way will have a combination of features, and the resulting cluster can have a blend of both the feature groups.

There is also a possibility to extend the feature selection to deal with the skewed clusters. The reduced feature set can be inadequate and can led to very low accuracy if the clusters are highly skewed. For example, suppose a dataset has 24 features, and in the clustering phase, 20 features become a part of the first cluster, and the remaining 4 features are part of the second cluster. Since there are only 2 clusters, there will be 2 Representative Features from each cluster. The resultant feature set is very small compared to the original dataset, and we can expect it to have low accuracy. It can be mitigated by considering more than one Representative Feature in the case of such skewed clusters. The algorithm should decide itself and select more features from each cluster to increase the accuracy.

Bibliography

[1] L. Parsons, E. Haque, and H. Liu, "Subspace clustering for high dimensional data: a review," *Acm Sigkdd Explorations Newsletter*, vol. 6, no. 1, pp. 90–105, 2004.

[2] S. Liu, D. Maljovec, B. Wang, P.-T. Bremer, and V. Pascucci, "Visualizing high-dimensional data: Advances in the past decade," *IEEE transactions on visualization and computer graphics*, vol. 23, no. 3, pp. 1249–1268, 2016.

[3] D. N. Reshef, Y. A. Reshef, H. K. Finucane, S. R. Grossman, G. McVean, P. J. Turnbaugh, E. S. Lander, M. Mitzenmacher, and P. C. Sabeti, "Detecting novel associations in large data sets," *science*, vol. 334, no. 6062, pp. 1518–1524, 2011.

[4] P. Mitra, C. Murthy, and S. K. Pal, "Unsupervised feature selection using feature similarity," *IEEE transactions on pattern analysis and machine intelligence*, vol. 24, no. 3, pp. 301–312, 2002.

[5] X. He, D. Cai, and P. Niyogi, "Laplacian score for feature selection," in *Advances in neural information processing systems*, 2006, pp. 507–514.

[6] Z. Zhao and H. Liu, "Spectral feature selection for supervised and unsupervised learning," in *Proceedings of the 24th international conference on Machine learning*. ACM, 2007, pp. 1151–1157.

[7] D. Cai, C. Zhang, and X. He, "Unsupervised feature selection for multi-cluster data," in *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2010, pp. 333–342.

[8] M. Dash and H. Liu, "Feature selection for clustering," in *Pacific-Asia Conference on knowledge discovery and data mining*. Springer, 2000, pp. 110–121.

[9] M. Ester, H.-P. Kriegel, and X. Xu, *A database interface for clustering in large spatial databases*. Inst. für Informatik, 1995.

[10] M.-S. Chen, J. Han, and P. S. Yu, "Data mining: an overview from a database perspective," *IEEE Transactions on Knowledge and data Engineering*, vol. 8, no. 6, pp. 866–883, 1996.

[11] C. C. Aggarwal and C. Zhai, *Mining text data*. Springer Science & Business Media, 2012.

[12] S. Alelyani, J. Tang, and H. Liu, "Feature selection for clustering: A review." *Data Clustering: Algorithms and Applications*, vol. 29, pp. 110–121, 2013.

[13] D. Zhou, O. Bousquet, T. N. Lal, J. Weston, and B. Schölkopf, "Learning with local and global consistency," in *Advances in neural information processing systems*, 2004, pp. 321–328.

[14] W. Du, K. Inoue, and K. Urahama, "Dimensionality reduction for semi-supervised face recognition," in *International Conference on Fuzzy Systems and Knowledge Discovery*. Springer, 2005, pp. 1–10.

[15] S. R. Bulò and M. Pelillo, "Dominant-set clustering: A review," *European Journal of Operational Research*, vol. 262, no. 1, pp. 1–13, 2017.

[16] B. H. Margolin and R. J. Light, "An analysis of variance for categorical data, ii: small sample comparisons with chi square and other competitors," *Journal of the American Statistical Association*, vol. 69, no. 347, pp. 755–764, 1974.

[17] I. Guyon and A. Elisseeff, "An introduction to variable and feature selection," *Journal of machine learning research*, vol. 3, no. Mar, pp. 1157–1182, 2003.

[18] S. Solorio-Fernández, J. A. Carrasco-Ochoa, and J. F. Martínez-Trinidad, "A review of unsupervised feature selection methods," *Artificial Intelligence Review*, pp. 1–42, 2019.

[19] N. Meghanathan, "A comprehensive analysis of the correlation between maximal clique size and centrality metrics for complex network graphs," *Egyptian Informatics Journal*, 2016.

[20] R. E. Bellman, *Adaptive control processes: a guided tour*. Princeton university press, 2015, vol. 2045.

[21] C. O. S. Sorzano, J. Vargas, and A. P. Montano, "A survey of dimensionality reduction techniques," *arXiv preprint arXiv:1403.2877*, 2014.

[22] I. Jolliffe, "Principal component analysis," in *International encyclopedia of statistical science*. Springer, 2011, pp. 1094–1096.

[23] J. Friedman, T. Hastie, and R. Tibshirani, *The elements of statistical learning*. Springer series in statistics New York, NY, USA:, 2001, vol. 1, no. 10.

[24] S. T. Roweis and L. K. Saul, "Nonlinear dimensionality reduction by locally linear embedding," *science*, vol. 290, no. 5500, pp. 2323–2326, 2000.

[25] S. Lee, "Drawbacks of principal component analysis," *arXiv preprint arXiv:1005.1770*, 2010.

[26] H. Yan and Y. Dai, "The comparison of five discriminant methods," in *2011 International Conference on Management and Service Science*. IEEE, 2011, pp. 1–4.

[27] J. A. Lee and M. Verleysen, *Nonlinear dimensionality reduction*. Springer Science & Business Media, 2007.

[28] H. Liu and H. Motoda, *Feature selection for knowledge discovery and data mining*. Springer Science & Business Media, 2012, vol. 454.

[29] A. L. Blum and P. Langley, "Selection of relevant features and examples in machine learning," *Artificial intelligence*, vol. 97, no. 1-2, pp. 245–271, 1997.

[30] J. G. Dy and C. E. Brodley, "Feature selection for unsupervised learning," *Journal of machine learning research*, vol. 5, no. Aug, pp. 845–889, 2004.

[31] A. Noruzi and H. Sahebi, "A graph-based feature selection method for improving medical diagnosis," *Advances in Computer Science: an International Journal*, vol. 4, no. 5, pp. 36–40, 2015.

[32] Z. Zhang and E. R. Hancock, "A graph-based approach to feature selection," in *International Workshop on Graph-Based Representations in Pattern Recognition*. Springer, 2011, pp. 205–214.

[33] A. M. Elgammal and M. A. Ismail, "A graph-based segmentation and feature extraction framework for arabic text recognition," in *Document Analysis and Recognition, 2001. Proceedings. Sixth International Conference on*. IEEE, 2001, pp. 622–626.

[34] M. Steinbach, L. Ertöz, and V. Kumar, "The challenges of clustering high dimensional data," in *New directions in statistical physics*. Springer, 2004, pp. 273–309.

[35] R. Agrawal, J. Gehrke, D. Gunopulos, and P. Raghavan, *Automatic subspace clustering of high dimensional data for data mining applications*. ACM, 1998, vol. 27, no. 2.

[36] S. Goil, H. Nagesh, and A. Choudhary, "Mafia: Efficient and scalable subspace clustering for very large data sets," in *Proceedings of the 5th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 1999, pp. 443–452.

[37] C. C. Aggarwal, J. L. Wolf, P. S. Yu, C. Procopiuc, and J. S. Park, "Fast algorithms for projected clustering," in *ACM SIGMoD Record*, vol. 28, no. 2. ACM, 1999, pp. 61–72.

[38] J. H. Friedman and J. J. Meulman, "Clustering objects on subsets of attributes (with discussion)," *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, vol. 66, no. 4, pp. 815–849, 2004.

[39] J. Tang, S. Alelyani, and H. Liu, "Feature selection for classification: A review," *Data classification: Algorithms and applications*, p. 37, 2014.

[40] M. Robnik-Šikonja and I. Kononenko, "Theoretical and empirical analysis of relieff and rrelieff," *Machine learning*, vol. 53, no. 1-2, pp. 23–69, 2003.

[41] S. Das, "Filters, wrappers and a boosting-based hybrid for feature selection," in *Icml*, vol. 1, 2001, pp. 74–81.

[42] H.-H. Hsu, C.-W. Hsieh *et al.*, "Feature selection via correlation coefficient clustering." *JSW*, vol. 5, no. 12, pp. 1371–1377, 2010.

[43] J. Benesty, J. Chen, Y. Huang, and I. Cohen, "Pearson correlation coefficient," in *Noise reduction in speech processing*. Springer, 2009, pp. 1–4.

[44] A. Tsanas, M. A. Little, and P. E. McSharry, "A simple filter benchmark for feature selection," *Journal of Machine Learning Research*, no. 1-24, 2010.

[45] L. Myers and M. J. Sirois, "Spearman correlation coefficients, differences between," *Encyclopedia of statistical sciences*, vol. 12, 2004.

[46] H. Peng, F. Long, and C. Ding, "Feature selection based on mutual information criteria of max-dependency, max-relevance, and min-redundancy," *IEEE Transactions on pattern analysis and machine intelligence*, vol. 27, no. 8, pp. 1226–1238, 2005.

[47] R. M. Gray, *Entropy and information theory*. Springer Science & Business Media, 2011.

[48] C. Yin, L. Feng, L. Ma, Z. Yin, and J. Wang, "A feature selection algorithm based on hoeffding inequality and mutual information," *Int J Signal Process Image Process Pattern Recognit*, vol. 8, no. 11, pp. 433–444, 2015.

[49] J. Duchi, "Cs229 supplemental lecture notes hoeffdingâs inequality."

[50] D. J. Sheskin, "Parametric and nonparametric statistical procedures," *Chapman & Hall/CRC: Boca Raton, FL*, 2000.

[51] D. L. Clason and T. J. Dormody, "Analyzing data measured by individual likert-type items," *Journal of agricultural education*, vol. 35, p. 4, 1994.

[52] S. Yu and H. Zhao, "Rough sets and laplacian score based cost-sensitive feature selection," *PloS one*, vol. 13, no. 6, p. e0197564, 2018.

[53] M. Belkin and P. Niyogi, "Laplacian eigenmaps and spectral techniques for embedding and clustering," in *Advances in neural information processing systems*, 2002, pp. 585–591.

[54] X. He and P. Niyogi, "Locality preserving projections," in *Advances in neural information processing systems*, 2004, pp. 153–160.

[55] T. Hesterberg, N. H. Choi, L. Meier, C. Fraley *et al.*, "Least angle and â1 penalized regression: A review," *Statistics Surveys*, vol. 2, pp. 61–93, 2008.

[56] A. Inselberg and B. Dimsdale, "Parallel coordinates for visualizing multi-dimensional geometry," in *Computer Graphics 1987*. Springer, 1987, pp. 25–44.

[57] E. Bertini, A. Tatu, and D. Keim, "Quality metrics in high-dimensional data visualization: An overview and systematization," *IEEE Transactions on Visualization and Computer Graphics*, vol. 17, no. 12, pp. 2203–2212, 2011.

[58] D. H. Jeong, C. Ziemkiewicz, B. Fisher, W. Ribarsky, and R. Chang, "ipca: An interactive system for pca-based visual analytics," in *Computer Graphics Forum*, vol. 28, no. 3. Wiley Online Library, 2009, pp. 767–774.

[59] C. Turkay, P. Filzmoser, and H. Hauser, "Brushing dimensions-a dual visual analysis model for high-dimensional data," *IEEE transactions on visualization and computer graphics*, vol. 17, no. 12, pp. 2591–2599, 2011.

[60] J. H. Friedman and J. W. Tukey, "A projection pursuit algorithm for exploratory data analysis," *IEEE Transactions on computers*, vol. 100, no. 9, pp. 881–890, 1974.

[61] T. N. Dang, A. Anand, and L. Wilkinson, "Timeseer: Scagnostics for high-dimensional time series," *IEEE Transactions on Visualization and Computer Graphics*, vol. 19, no. 3, pp. 470–483, 2012.

[62] D. Guo, "Coordinating computational and visual approaches for interactive feature selection and multivariate clustering," *Information Visualization*, vol. 2, no. 4, pp. 232–246, 2003.

[63] A. Inselberg and B. Dimsdale, "Parallel coordinates: a tool for visualizing multi-dimensional geometry," in *Proceedings of the 1st conference on Visualization'90*. IEEE Computer Society Press, 1990, pp. 361–378.

[64] W. Peng, M. O. Ward, and E. A. Rundensteiner, "Clutter reduction in multi-dimensional data visualization using dimension reordering," in *IEEE Symposium on Information Visualization*. IEEE, 2004, pp. 89–96.

[65] P. Hoffman, G. Grinstein, K. Marx, I. Grosse, and E. Stanley, "Dna visual and analytic data mining," in *Proceedings. Visualization'97 (Cat. No. 97CB36155)*. IEEE, 1997, pp. 437–441.

[66] H. Chernoff, "The use of faces to represent points in k-dimensional space graphically," *Journal of the American Statistical Association*, vol. 68, no. 342, pp. 361–368, 1973.

[67] D. A. Keim, M. C. Hao, J. Ladisch, M. Hsu, and U. Dayal, "Pixel bar charts: A new technique for visualizing large multi-attribute data sets without aggregation," in *IEEE Symposium on Information Visualization: INFOVIS 2001*, 2001, pp. 113–120.

[68] M. N. Wright and A. Ziegler, "ranger: A fast implementation of random forests for high dimensional data in c++ and r," *arXiv preprint arXiv:1508.04409*, 2015.

[69] D. J. Stekhoven and P. Bühlmann, "Missforestânon-parametric missing value imputation for mixed-type data," *Bioinformatics*, vol. 28, no. 1, pp. 112–118, 2011.

[70] S. v. Buuren and K. Groothuis-Oudshoorn, "mice: Multivariate imputation by chained equations in r," *Journal of statistical software*, pp. 1–68, 2010.

[71] T. Clark and L. Foster, "Chi-square: introducing the âgoodness of fitâtest and the âtest of associationâ," 2014.

[72] T. Speed, "A correlation for the 21st century," *Science*, vol. 334, no. 6062, pp. 1502–1503, 2011.

[73] R. Paredes and E. Chávez, "Using the k-nearest neighbor graph for proximity searching in metric spaces," in *International Symposium on String Processing and Information Retrieval*. Springer, 2005, pp. 127–138.

[74] M. Lucińska and S. T. Wierzchoń, "Spectral clustering based on k-nearest neighbor graph," in *IFIP International Conference on Computer Information Systems and Industrial Management*. Springer, 2012, pp. 254–265.

[75] W. Apriani, N. S. Sihombing, Z. Habsyah, and S. Suwilo, "Finding maximal clique in a graph."

[76] H. Elgazzar and A. Elmaghraby, "Evolutionary centrality and maximal cliques in mobile social networks," *International Journal of Computer Science & Information Technology (IJCSIT) Vol*, vol. 10, 2018.

[77] S. Segarra and A. Ribeiro, "Stability and continuity of centrality measures in weighted graphs," *IEEE Transactions on Signal Processing*, vol. 64, no. 3, pp. 543–555, 2015.

[78] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten, "The weka data mining software: an update," *ACM SIGKDD explorations newsletter*, vol. 11, no. 1, pp. 10–18, 2009.

[79] Y. Yang, H. T. Shen, Z. Ma, Z. Huang, and X. Zhou, "L2, 1-norm regularized discriminative feature selection for unsupervised," in *Twenty-Second International Joint Conference on Artificial Intelligence*, 2011.

[80] Z. Li, Y. Yang, J. Liu, X. Zhou, and H. Lu, "Unsupervised feature selection using nonnegative spectral analysis," in *Twenty-Sixth AAAI Conference on Artificial Intelligence*, 2012.

[81] P. R. Kersten, J.-S. Lee, and T. L. Ainsworth, "Unsupervised classification of polarimetric synthetic aperture radar images using fuzzy clustering and em clustering," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 43, no. 3, pp. 519–527, 2005.

[82] R. E. Tarjan and A. E. Trojanowski, "Finding a maximum independent set," *SIAM Journal on Computing*, vol. 6, no. 3, pp. 537–546, 1977.

Appendix A

Appendix

A.1 Existing Feature Selection Methods

Algorithm 6: Unspervised Feature Selection using Feature Similarity [4]

Procedure : $featureSelection$;
Assumptions : $O = F_i, i = 1...D$ be the original number of features D.
(F_i, F_j) denotes the dissimilarity measure between F_i and F_j computed using λ_2.
r_i^k represent the dissimilarity between F_i and k-th nearest neighbor in R.
Step 1: Choose an initial value of $k \leq$ D-1. Initialize reduced feature set to original feature set,i.e., R $\leftarrow O$.
Step 2: For each feature $F_i \in R$, compute r_i^k.
Step 3: Find feature $F_{i'}$ for which $r_{i'}^k$ is minimum. Retain this feature in R and discard k nearest features of $F_{i'}$. Let $\epsilon = r_{i'}^k$.
Step 4: If k > cardinality(R)-1: k = cardinality(R)-1.
Step 5: If k=1, Goto **Step 8**.
Step 6: **while** $r_{i'}^k > \epsilon$ **do**
  a) $k = k - 1$.
    $r_{i'}^k = inf_{F_i \in R} r_i^k$.
  /* k is decremented by 1 until the k-th nearest neighbor of atleast one of the features in R is less than ϵ- dissimilar features. */
  b) If k=1, Goto **Step 8**.
  /* If no feature in R has less than ϵ- dissimilar nearest neighbor, select all the features in R */
end
Step 7: Goto **Step 2**.
Step 8: Return the feature set R as the reduced feature set.

Algorithm 7: Laplacian Score for Feature Selection [5]

Procedure : $featureSelection$;
Assumptions : Let L_r denote the he Laplacian Score of the r-th feature.
Let f_{ri} denote the i-th sample of the r-th feature, $i = 1, ..., m$.
Step 1: A nearest neighbor graph G with m nodes is constructed. An edge is put between nodes i and j if x_i and x_j are "close", i.e. x_i is among k nearest neighbors of x_j or vice-versa.
Step 2: **if** *nodes i and j are connected* **then**
  $S_{ij} \leftarrow e^{-\frac{|x_i - x_j|^2}{t}}$,where t is a suitable constant.
end
else
  $S_{ij} \leftarrow 0$
end
Step 3: For each r-th feature.
  $f_r = [f_{r1}, f_{r2}, f_{r3} ... f_{rm}]^T$,
  $D = diag(S\mathbf{1})$,
  $\mathbf{1} = [1...1]^T$,
  $L = D - S$
where D is the diagonal matrix, S is the similarity matrix and L is the Laplacian matrix. Let $\tilde{f}_r$ be given as
  $\tilde{f}_r = f_r - \frac{f_r^T D1}{1^T D1}\mathbf{1}$
Step 4: The Laplacian score of the r-th feature.
  $L_r = \frac{\tilde{f_r}^T L \tilde{f_r}}{\tilde{f_r}^T D \tilde{f_r}}$

Algorithm 8: Spectral Feature Selection [6]

Procedure : $featureSelection$;
Input: X, k, $\varphi \in (\varphi_1, \varphi_2, \varphi_3)$
Output: SF_{SPEC} - the ranked feature list.
Step 1: Construct S, the similarity set from X and Y.
Step 2: Construct graph G from S.
Step 3: Build W, D and L from G.
Step 4: For each feature vector f_i do
  $SF_{SPEC}(i) \leftarrow \varphi(F_i)$
Step 5: Ranking SF_{SPEC} in ascending order for φ_1 and φ_2 or descending order for φ_3.
Step 6: Return SF_{SPEC}.

Algorithm 9: Unsupervised Feature Selection for Multi-Cluster Data [7]

Procedure : $featureSelection$;
Input N data points with M features;
The number of clusters K;
The number of selected features d;
The number of nearest neighbors p.
Output d selected features.
Step 1: Construct a p nearest neighbor graph.
Step 2: : Solve the generalized eigen-problem in Let Y = $[y_1...y_K]$ contain the top K eigenvectors with respect to the smallest eigenvalues.
Step 3: Solve K L1-regularized regression problem using LARs algorithm with the cardinality constraint set to d.
Step 4: Compute the MCFS score for each feature
Step 5: Return the top d features according to their MCFS scores.

A.2 Quantitative Evaluation Results

A.2.1 Automobile Dataset

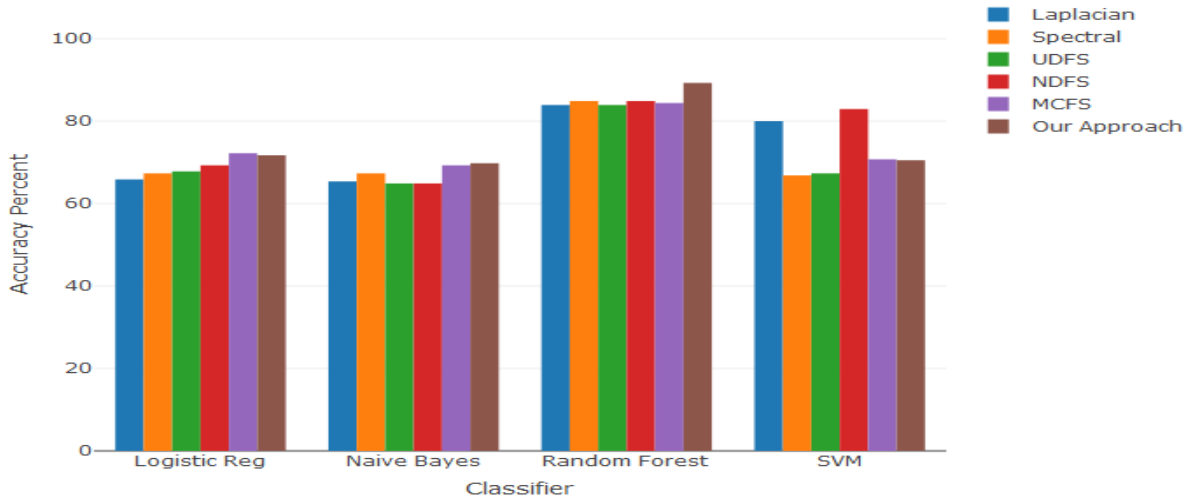


Figure A.1: Classification Accuracy (Automobile Dataset)

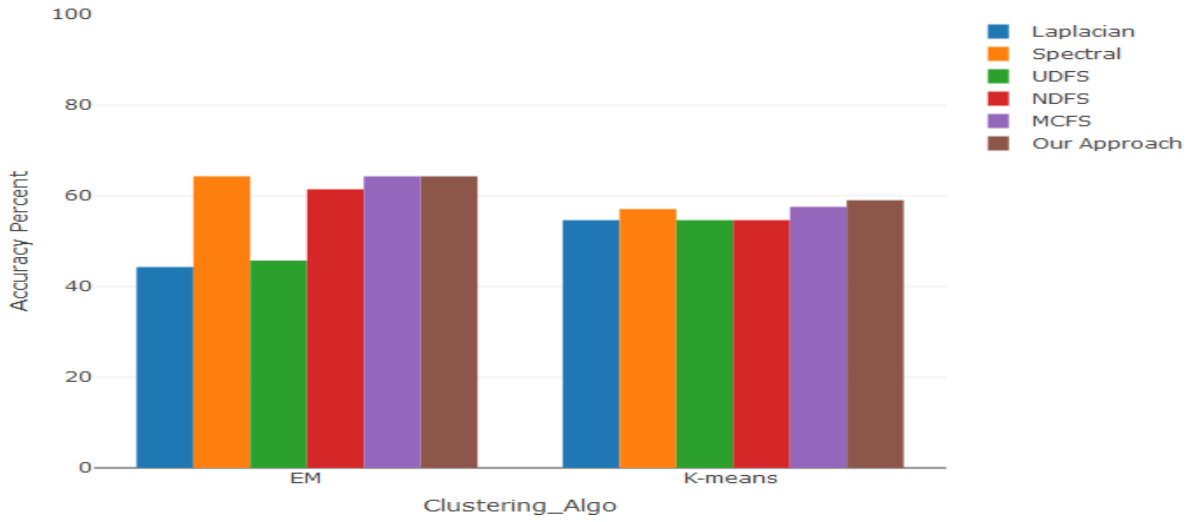


Figure A.2: Clustering Accuracy (Automobile Dataset)

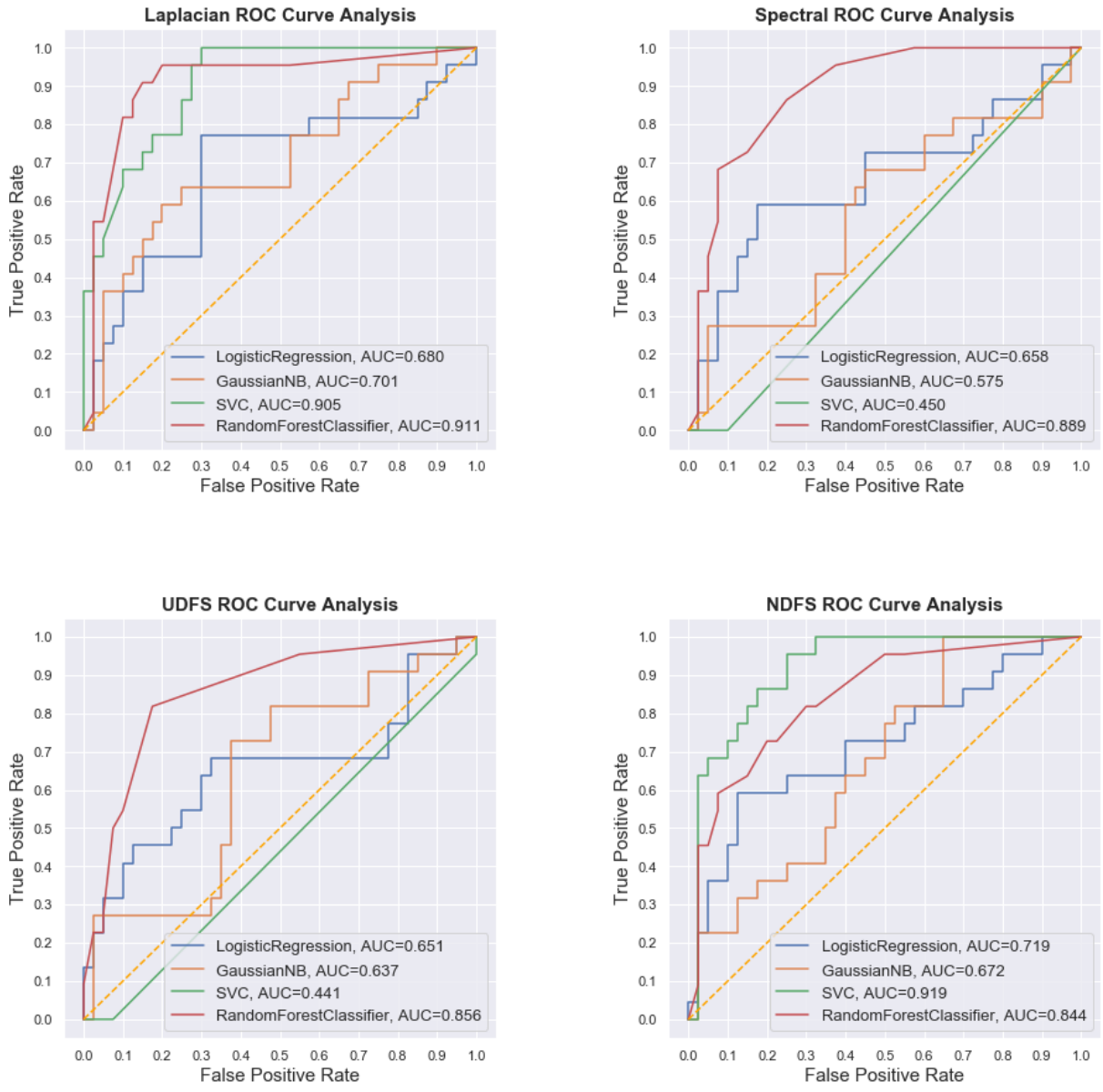


Figure A.3: ROC Curves (Automobile Dataset) 1

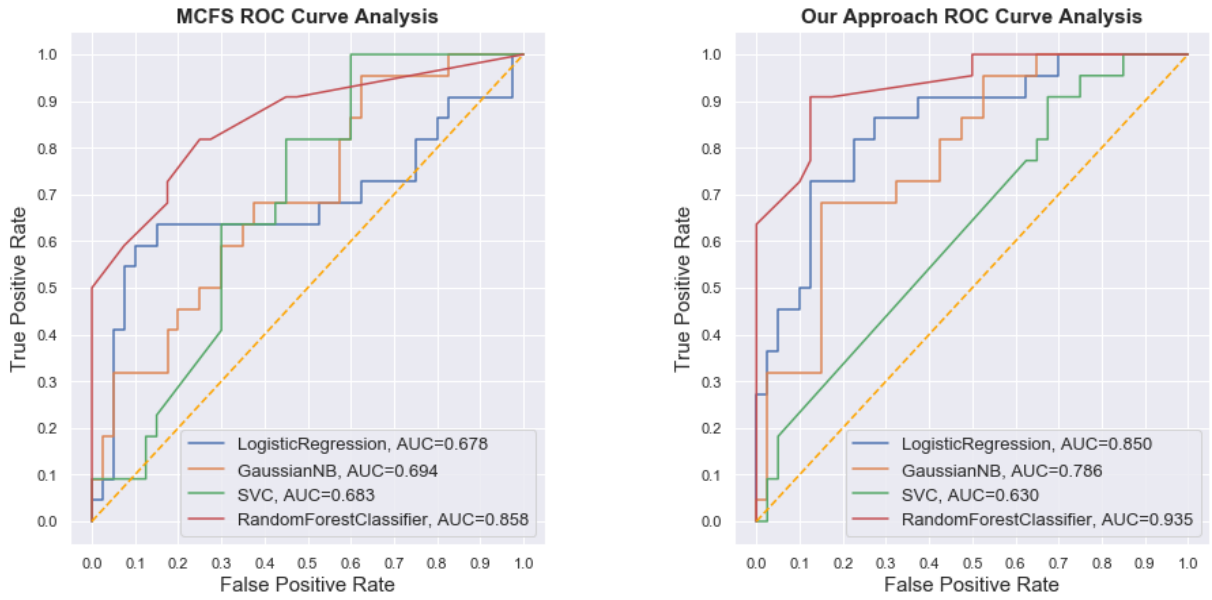


Figure A.4: ROC Curves (Automobile Dataset) 2

	Laplacian	Spectral	UDFS	NDFS	MCFS	Our Approach
Rep Entropy	0.3462	0.00186	0.0150	0.216	0.00181	0.0162

Table A.1: Representation Entropy (Automobile Dataset)

A.2.2 Breast Cancer Dataset

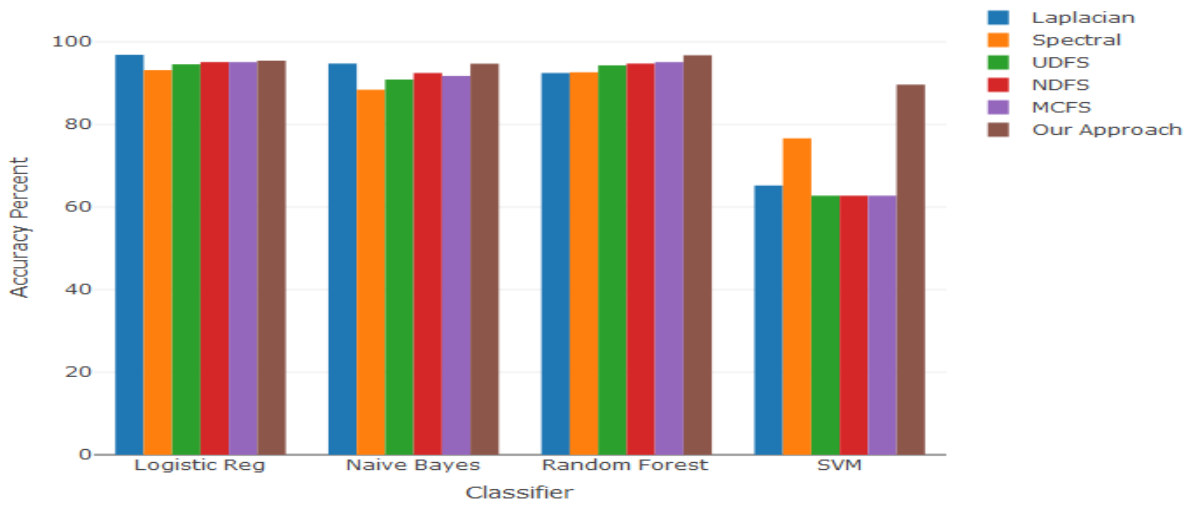


Figure A.5: Classification Accuracy (Breast Cancer Dataset)

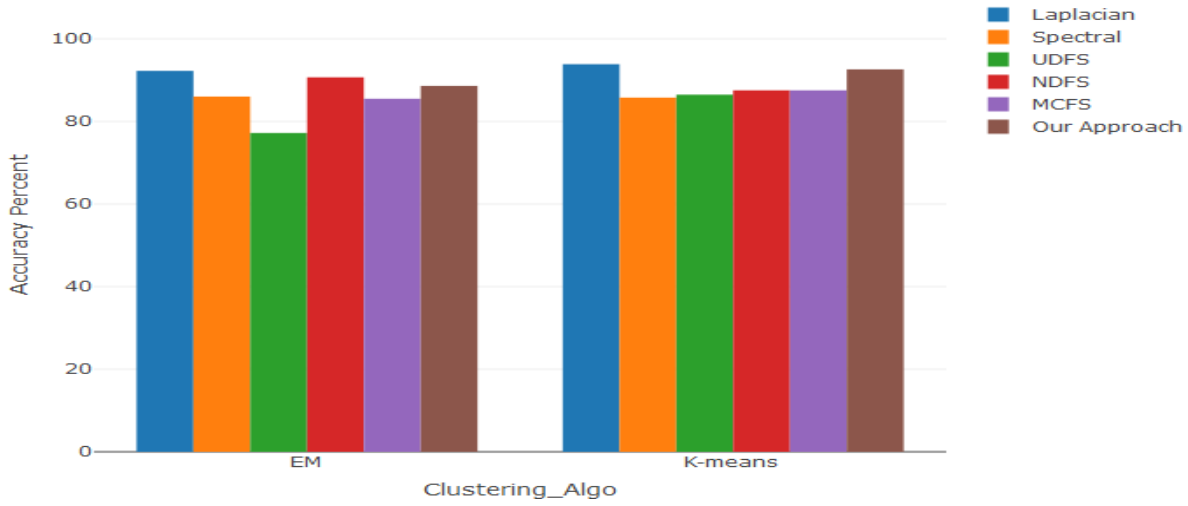


Figure A.6: Clustering Accuracy (Breast Cancer Dataset)

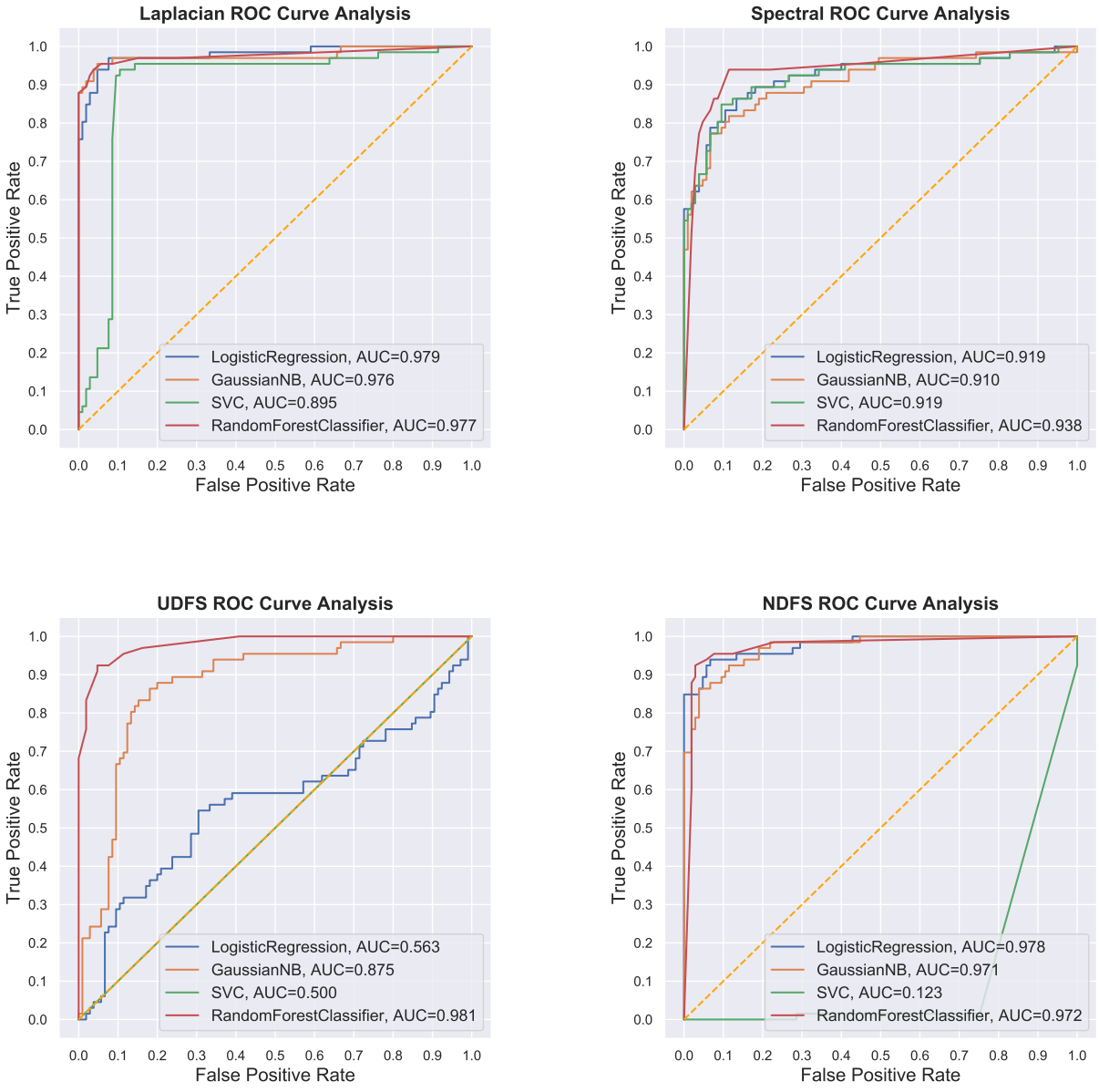


Figure A.7: ROC Curves (Breast Cancer Dataset) 1

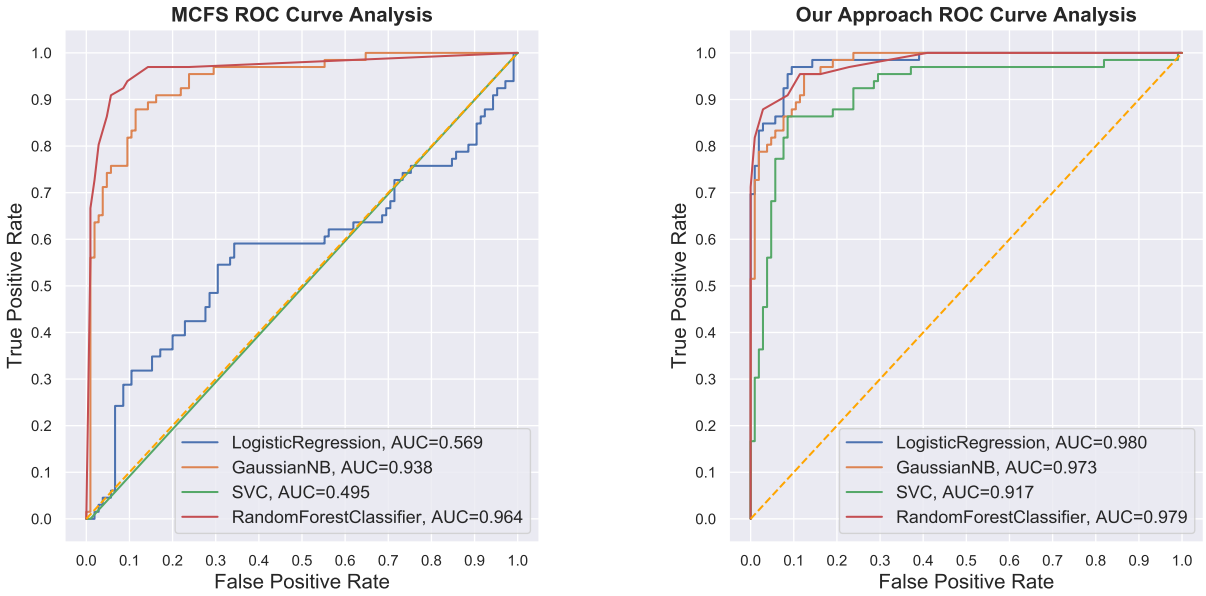


Figure A.8: ROC Curves (Breast Cancer Dataset) 2

	Laplacian	**Spectral**	**UDFS**	**NDFS**	**MCFS**	**Our Approach**
Rep Entropy	0.0012	0.359	0.00091	0.04194	-	0.00920

Table A.2: Representation Entropy (Breast Cancer Dataset)

A.2.3 Auto Univ Dataset

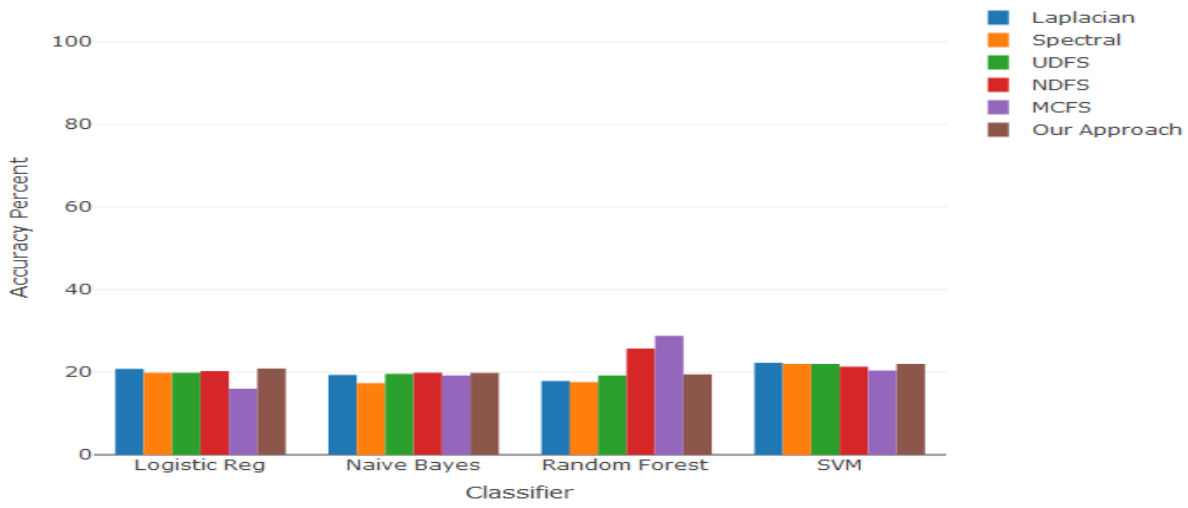


Figure A.9: Classification Accuracy (Auto Univ Dataset)

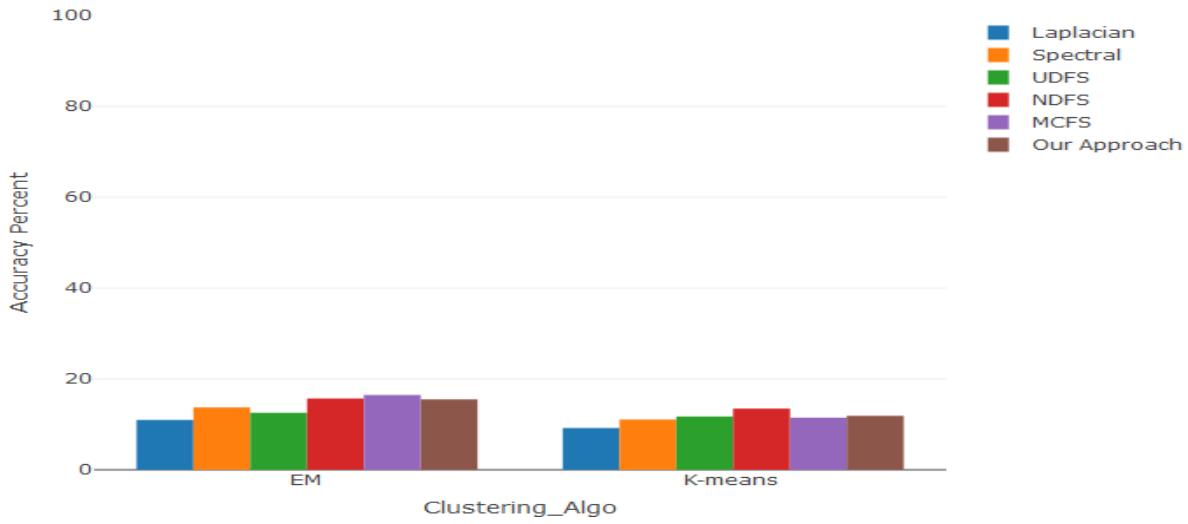


Figure A.10: Clustering Accuracy (Auto Univ Dataset)

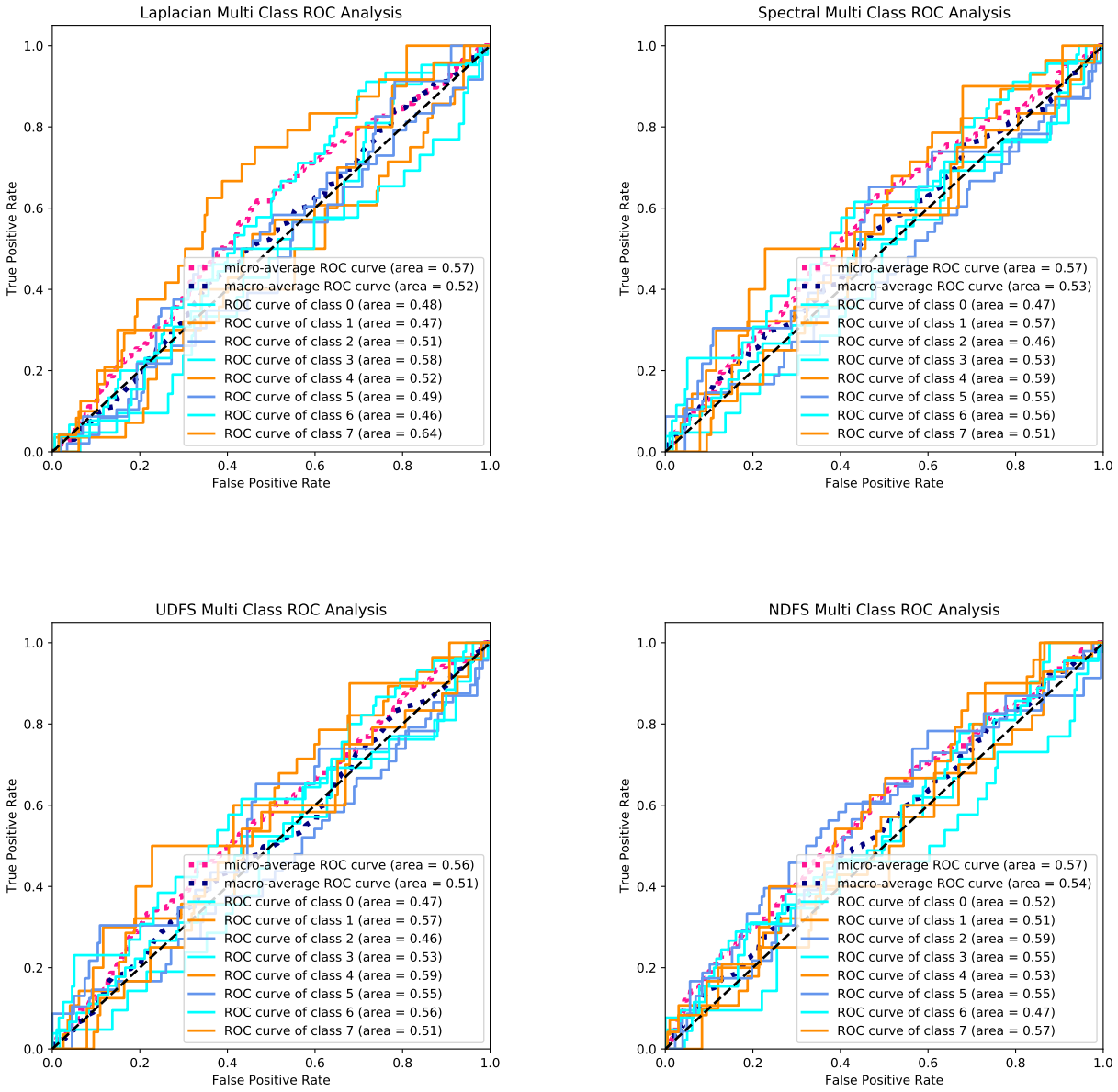


Figure A.11: ROC Curves (Auto Univ Dataset) 1

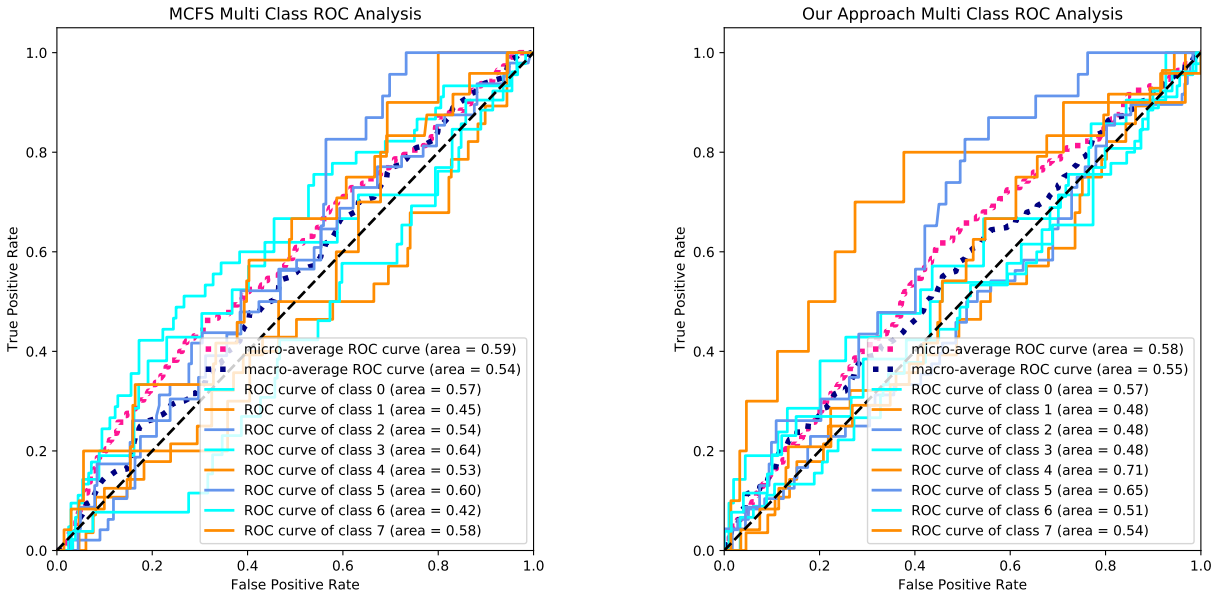


Figure A.12: ROC Curves (Auto Univ Dataset) 2

	Laplacian	**Spectral**	**UDFS**	**NDFS**	**MCFS**	**Our Approach**
Rep Entropy	0.05386	0.003875	0.05718	0.90212	0.77475	0.00391

Table A.3: Representation Entropy (Auto Univ Dataset)

A.2.4 QSAR Biodegradation Dataset (No Missing Values)

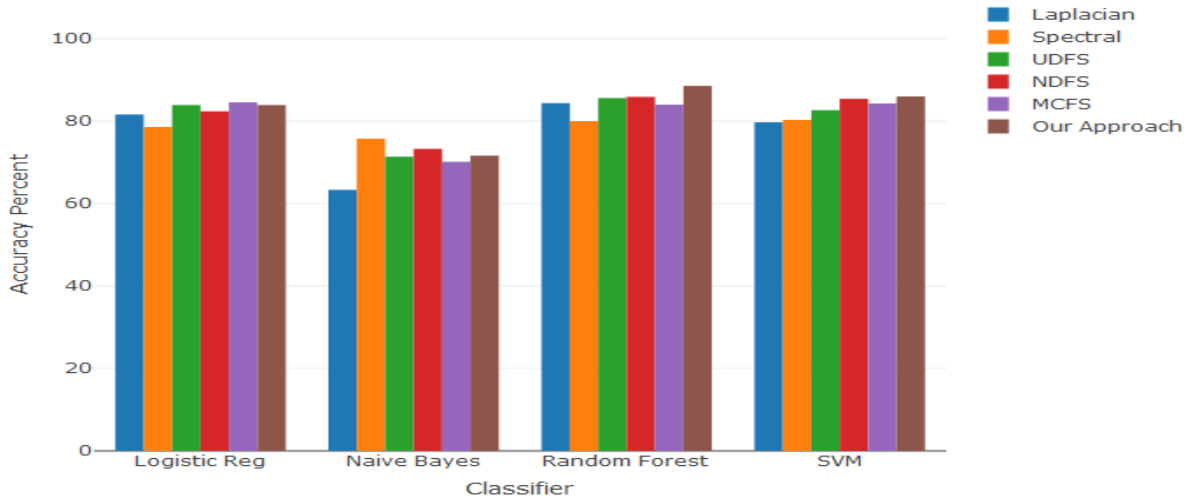


Figure A.13: Classification Accuracy (QSAR Bio Dataset)

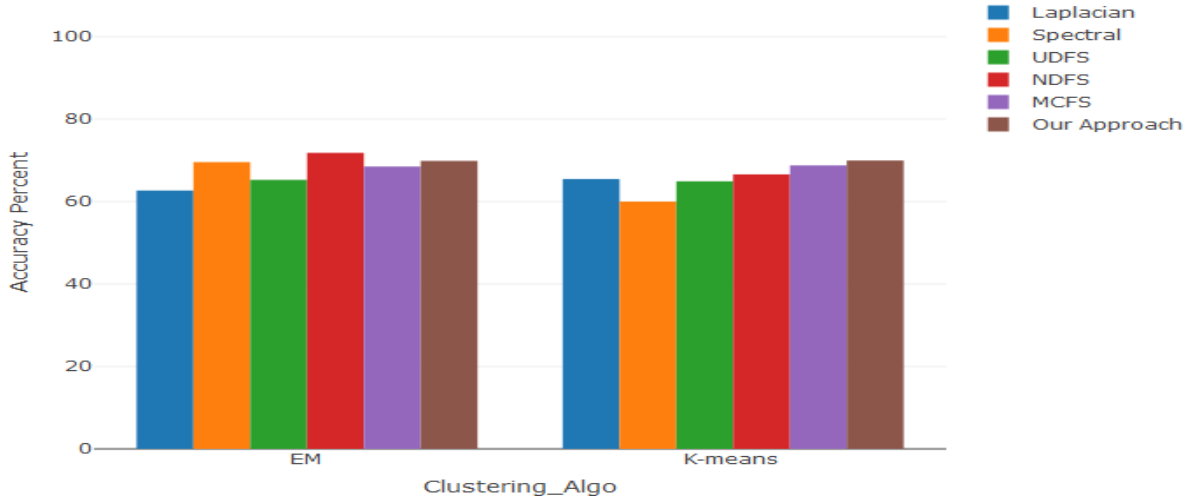


Figure A.14: Clustering Accuracy (QSAR Bio Dataset)

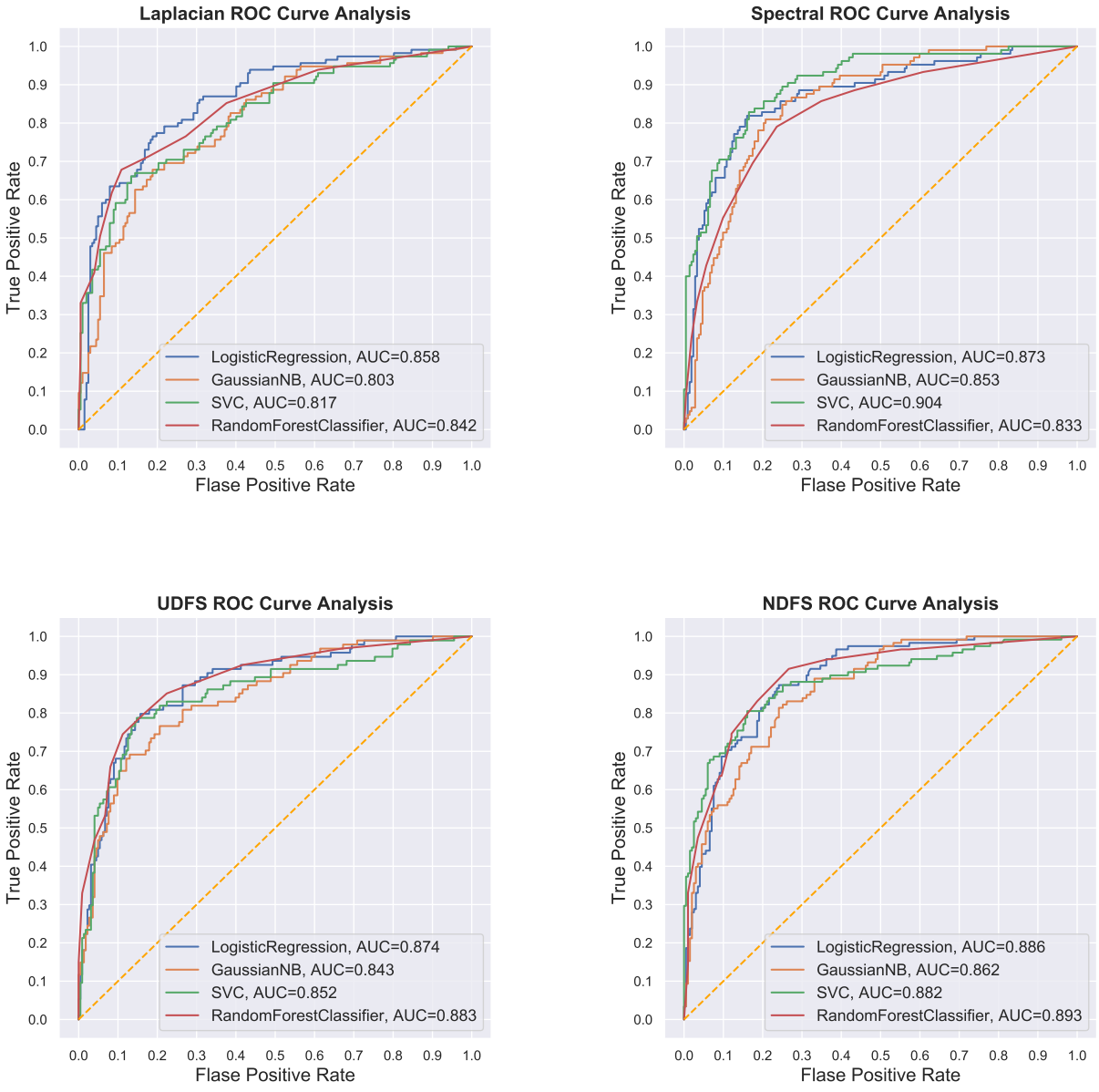


Figure A.15: ROC Curves (QSAR Bio Dataset) 1

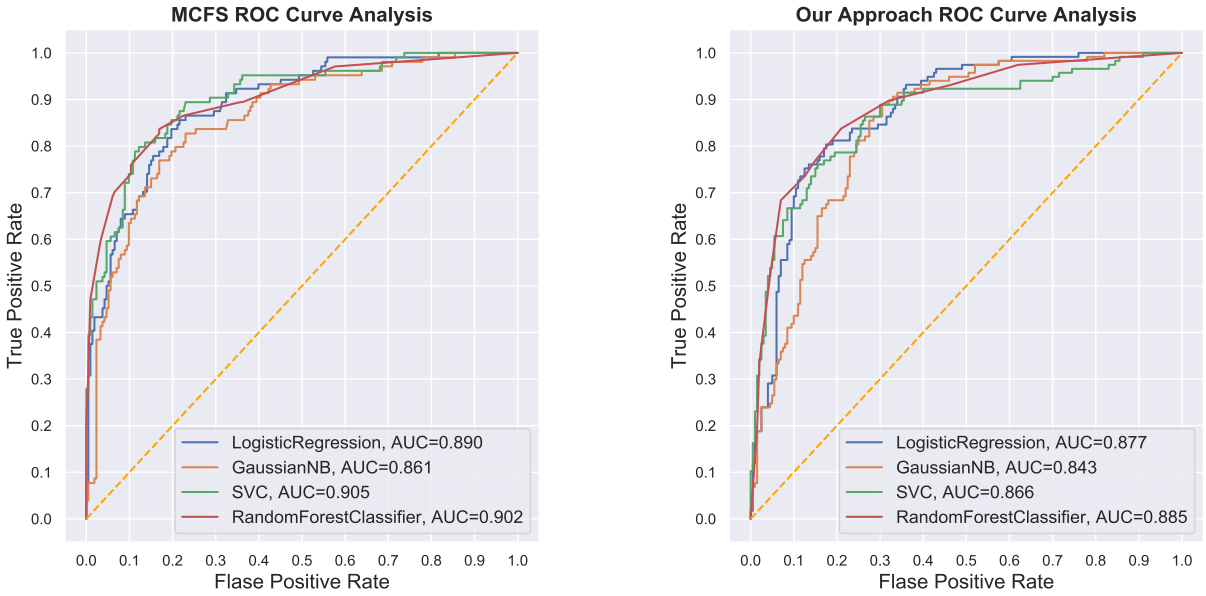


Figure A.16: ROC Curves (QSAR Bio Dataset) 2

	Laplacian	**Spectral**	**UDFS**	**NDFS**	**MCFS**	**Our Approach**
Rep Entropy	0.47426	0.70142	0.5449	0.33878	0.6993	0.65399

Table A.4: Representation Entropy (QSAR Bio Dataset)

A.2.5 QSAR Biodegradation Dataset (With Missing Values)

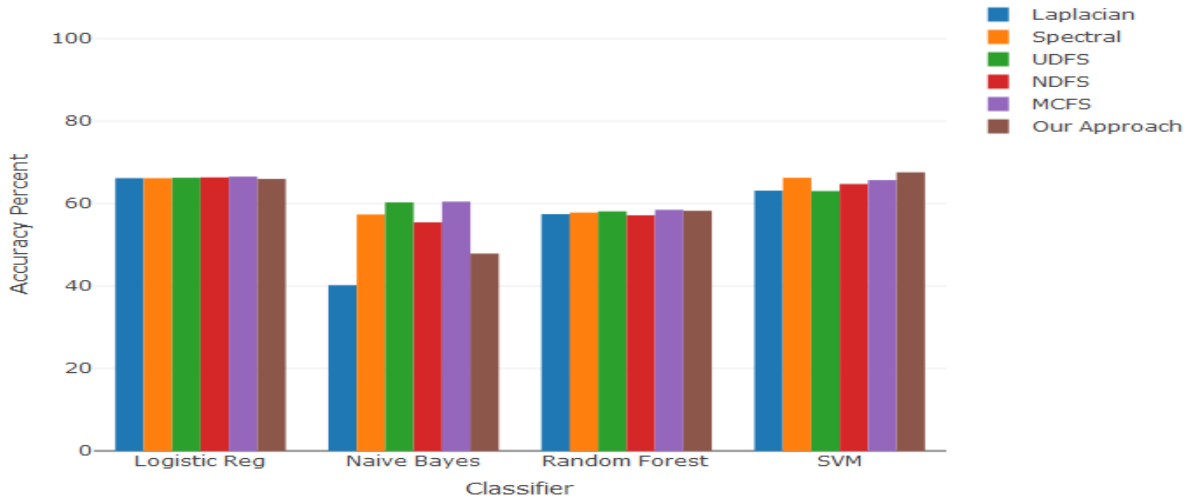


Figure A.17: Classification Accuracy (QSAR Bio Dataset)

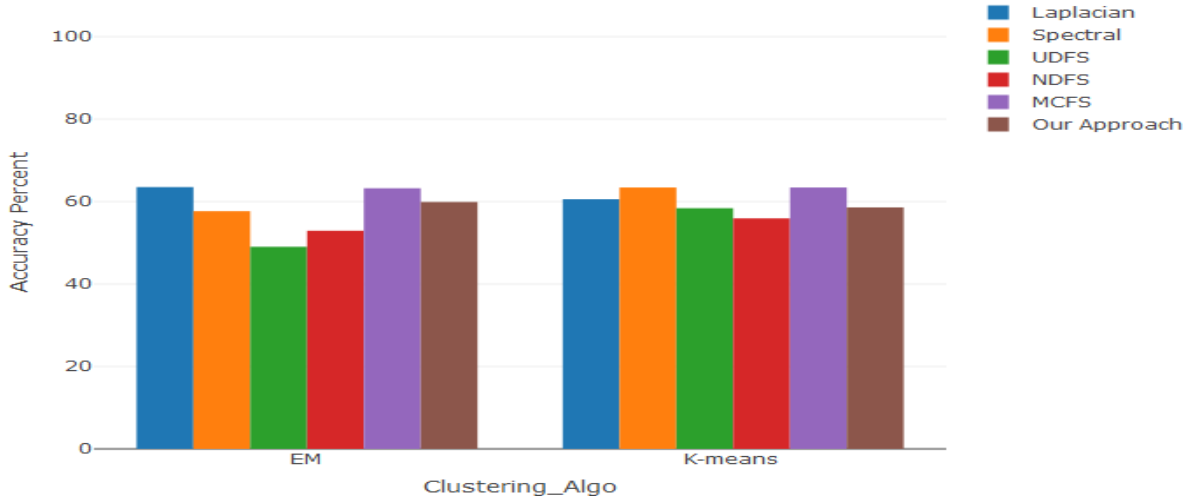


Figure A.18: Clustering Accuracy (QSAR Bio Dataset)

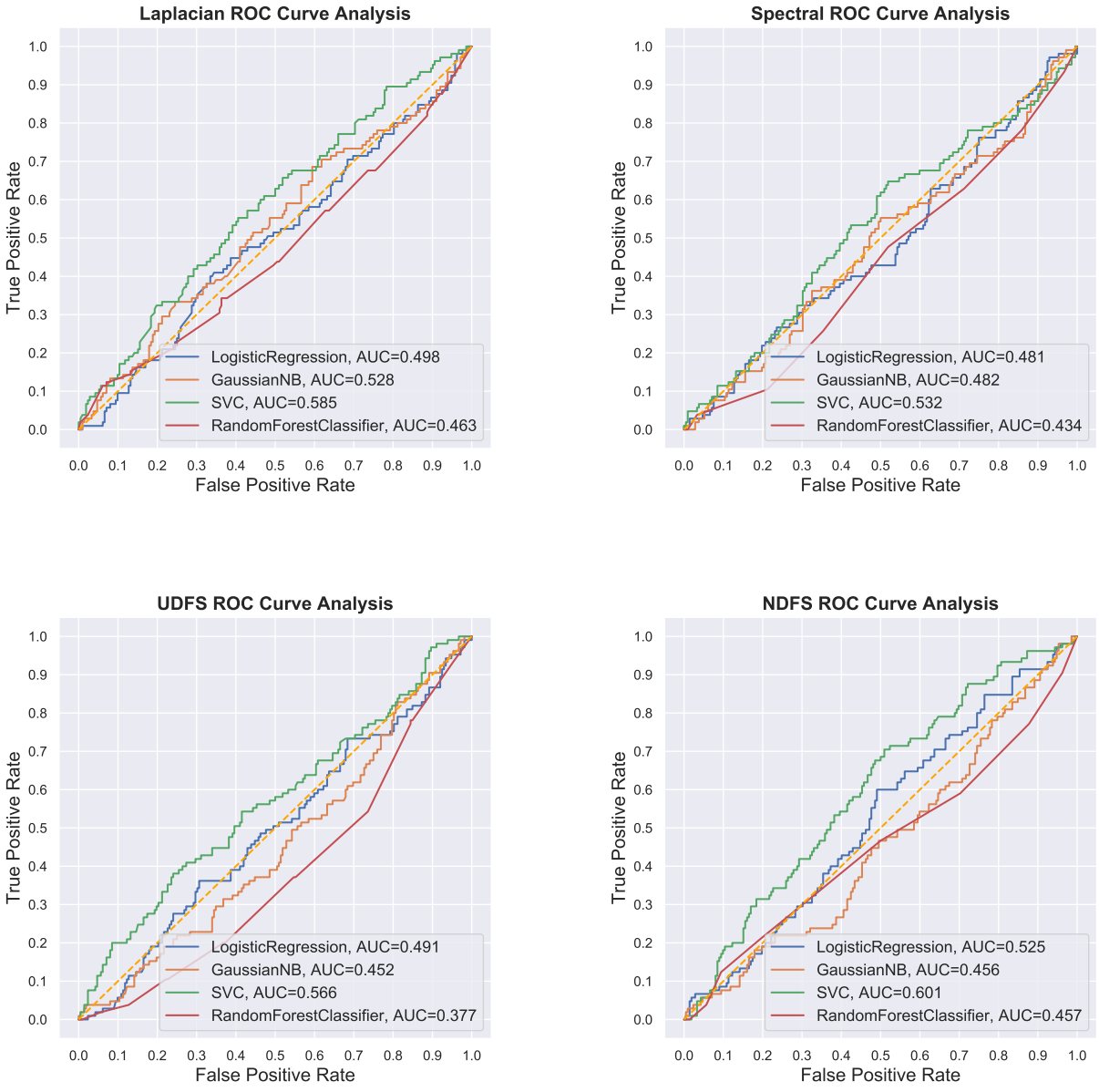


Figure A.19: ROC Curves (QSAR Bio Dataset) 1

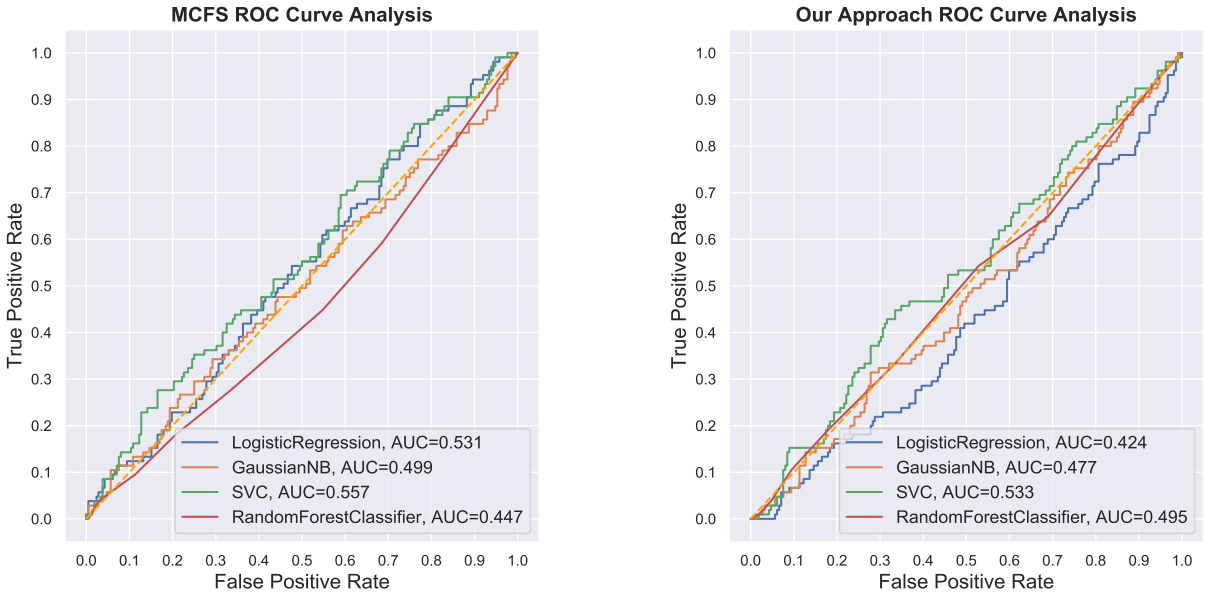


Figure A.20: ROC Curves (QSAR Bio Dataset) 2

	Laplacian	Spectral	UDFS	NDFS	MCFS	Our Approach
Rep Entropy	0.57004	0.68327	0.54075	0.42013	0.69525	0.62873

Table A.5: Representation Entropy (QSAR Bio Dataset)

A.2.6 Sonar Dataset

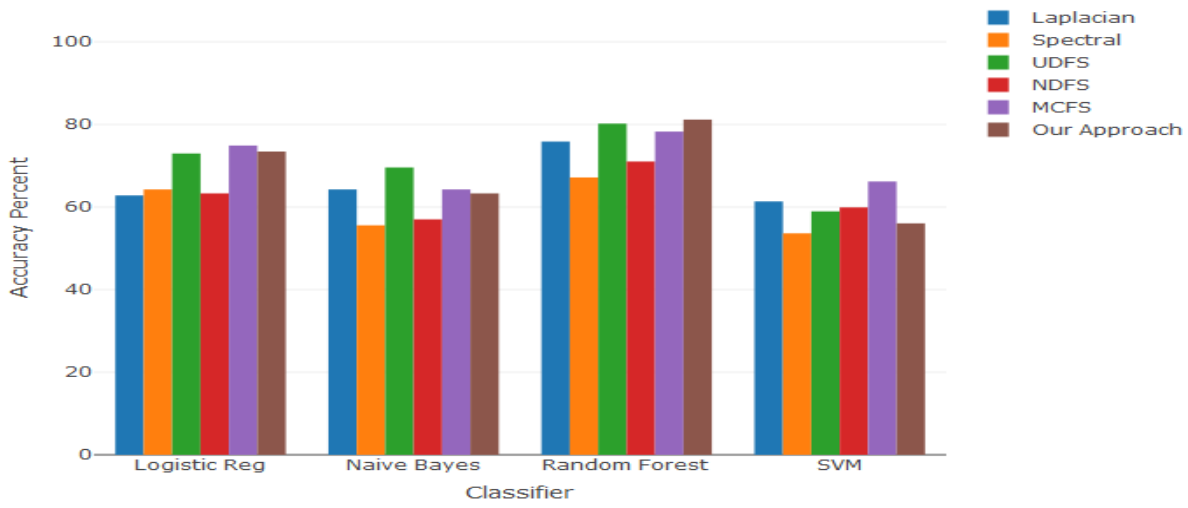


Figure A.21: Classification Accuracy (Sonar Dataset)

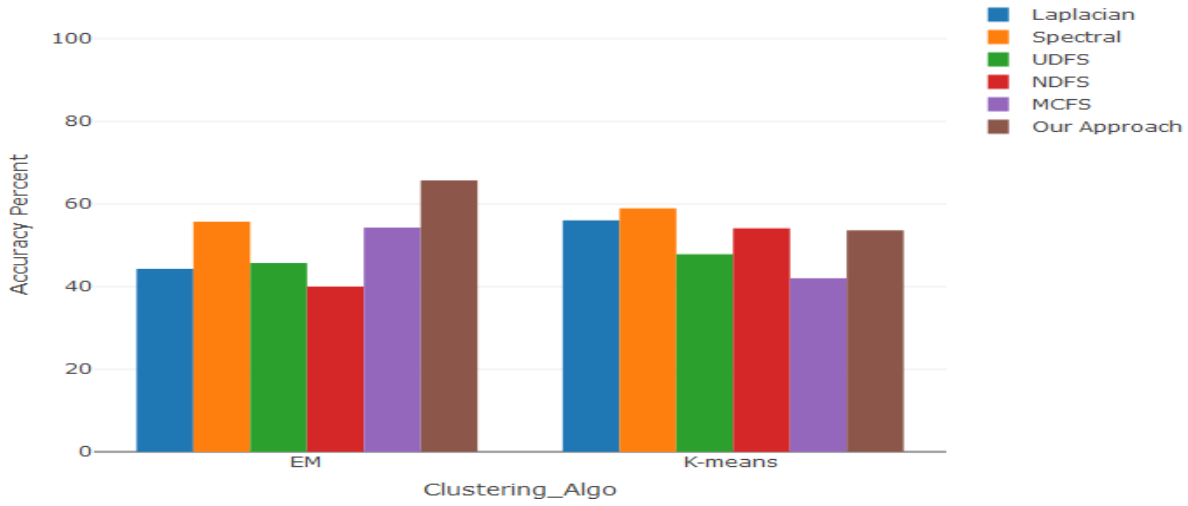


Figure A.22: Clustering Accuracy (Sonar Dataset)

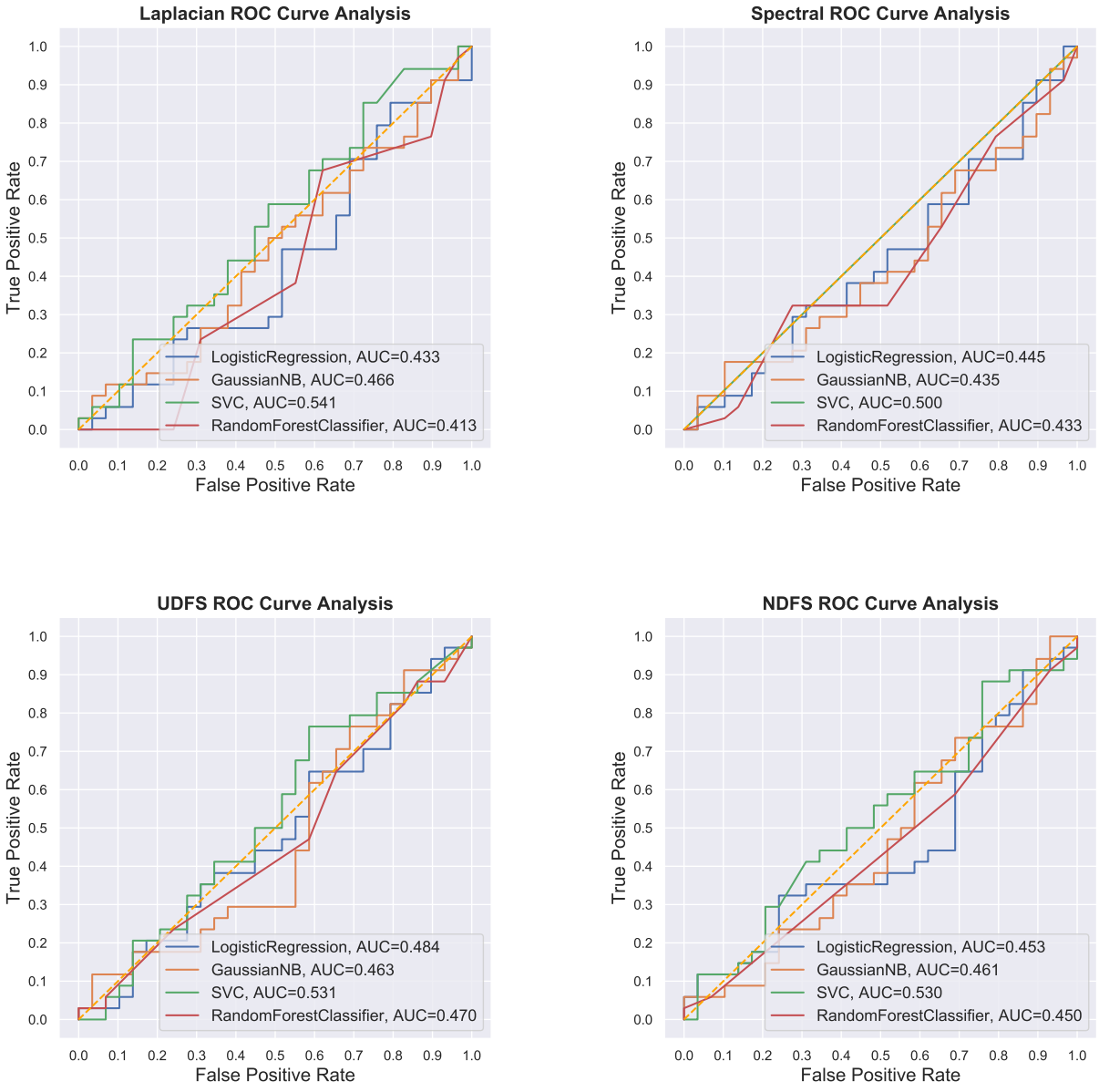


Figure A.23: ROC Curves (Sonar Dataset) 1

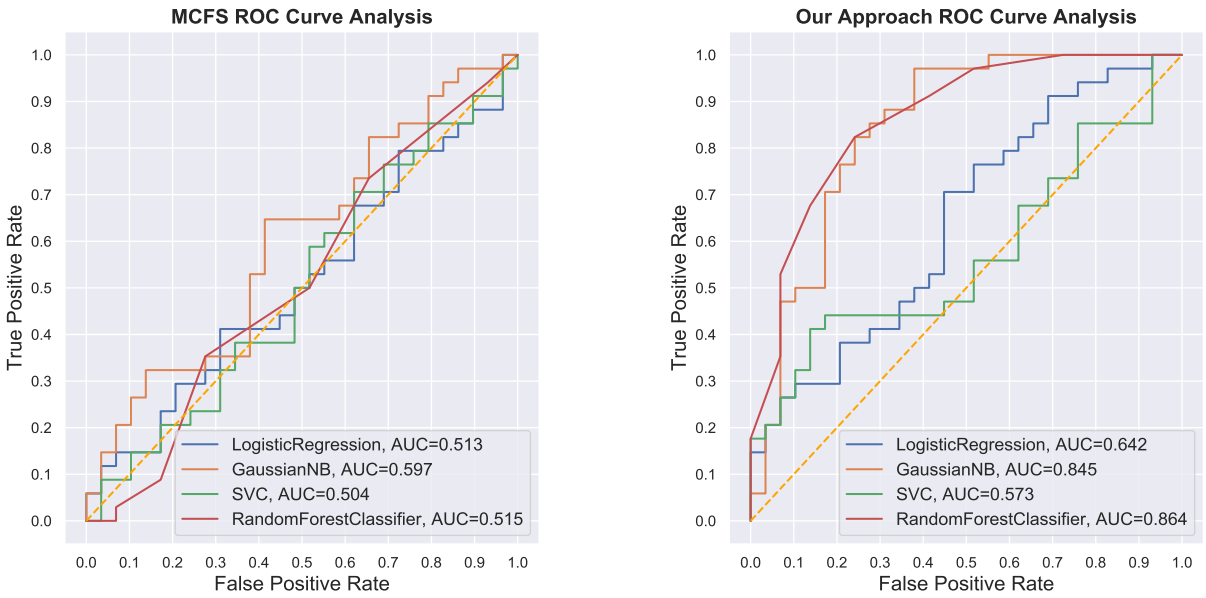


Figure A.24: ROC Curves (Sonar Dataset) 2

	Laplacian	Spectral	UDFS	NDFS	MCFS	Our Approach
Rep Entropy	0.72008	0.527482	0.796367	0.76677	0.79827	0.76724

Table A.6: Representation Entropy (Sonar Dataset)

A.2.7 Emotions Dataset

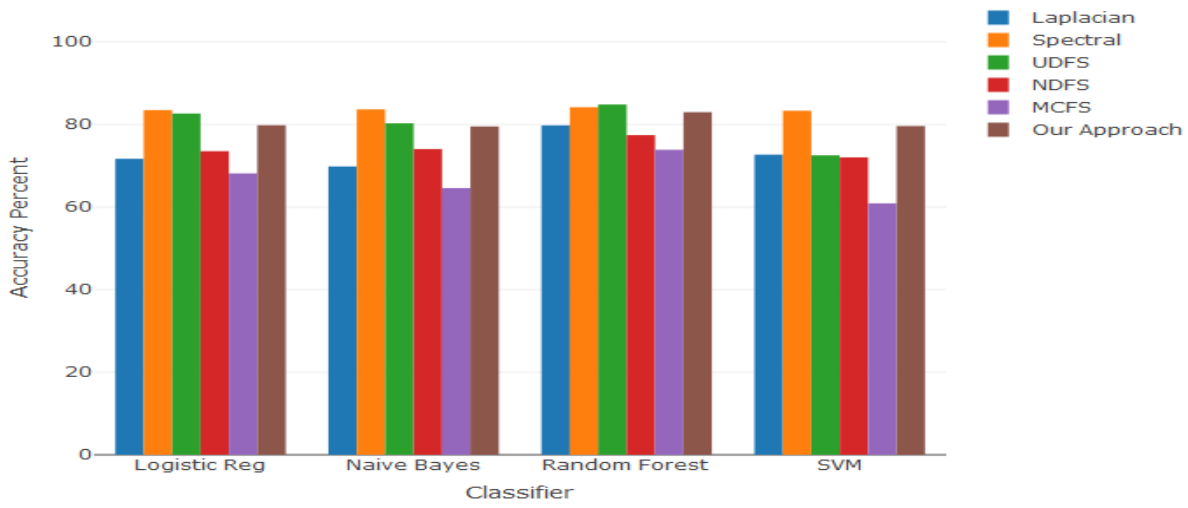


Figure A.25: Classification Accuracy (Emotions Dataset)

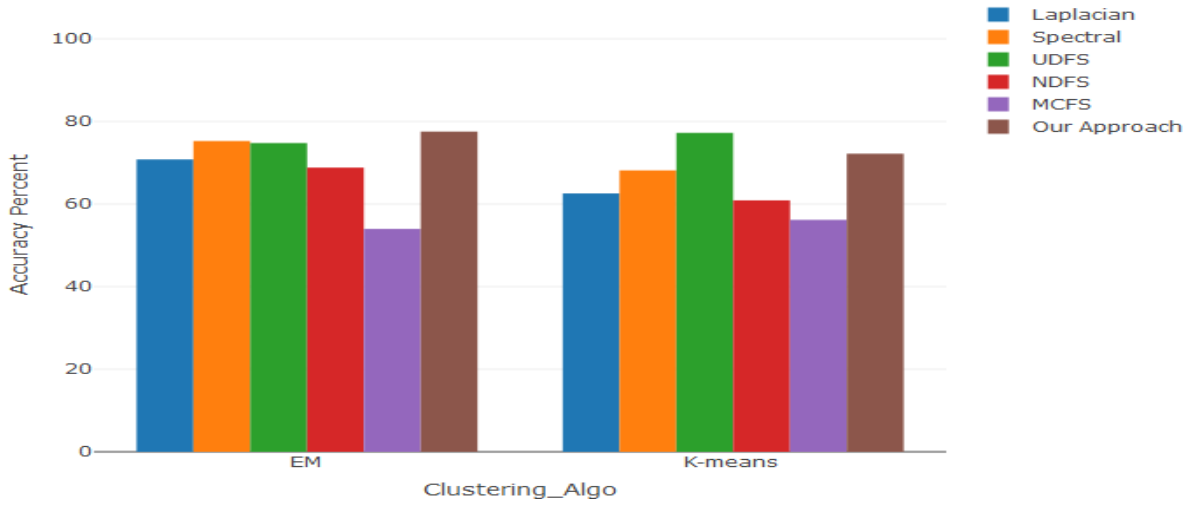


Figure A.26: Clustering Accuracy (Emotions Dataset)

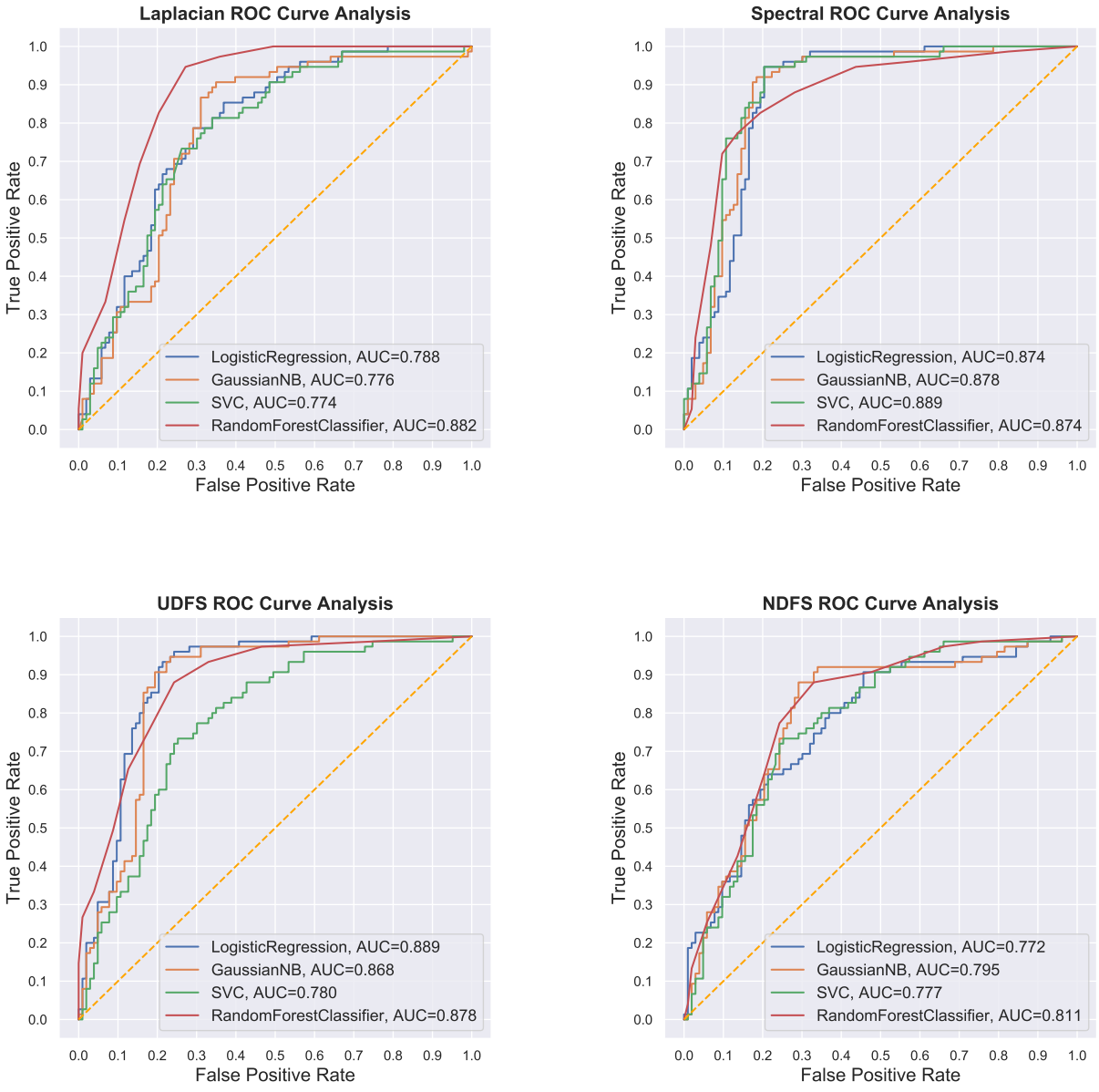


Figure A.27: ROC Curves (Emotions Dataset) 1

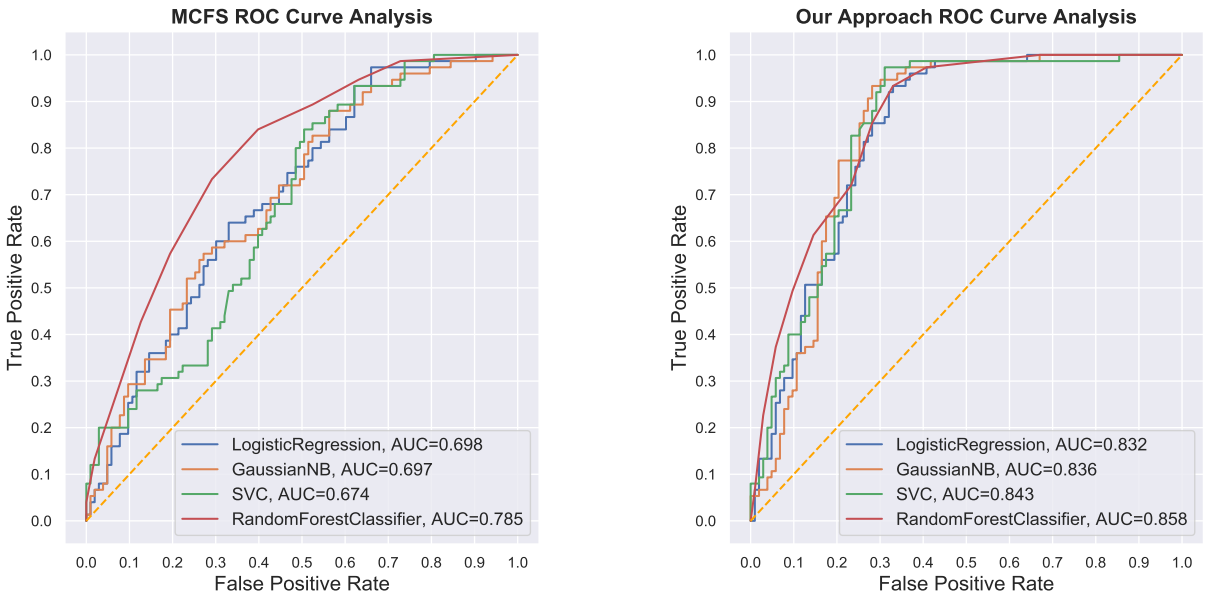


Figure A.28: ROC Curves (Emotions Dataset) 2

	Laplacian	Spectral	UDFS	NDFS	MCFS	Our Approach
Rep Entropy	0.20329	0.91409	0.20628	0.20617	0.01400	0.46283

Table A.7: Representation Entropy (Emotions Dataset)

A.2.8 Spectrometer Dataset

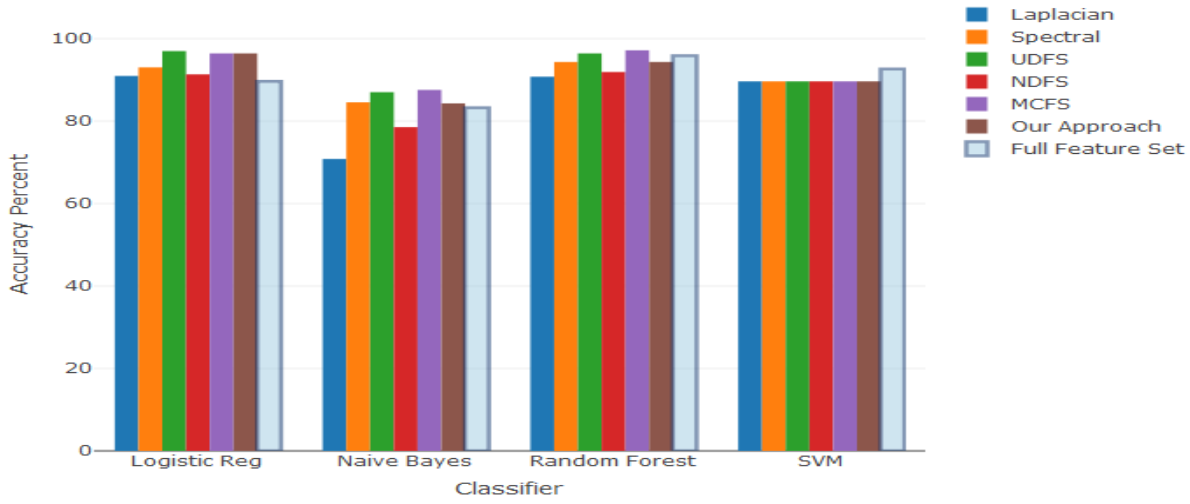


Figure A.29: Classification Accuracy (Spectrometer Dataset)

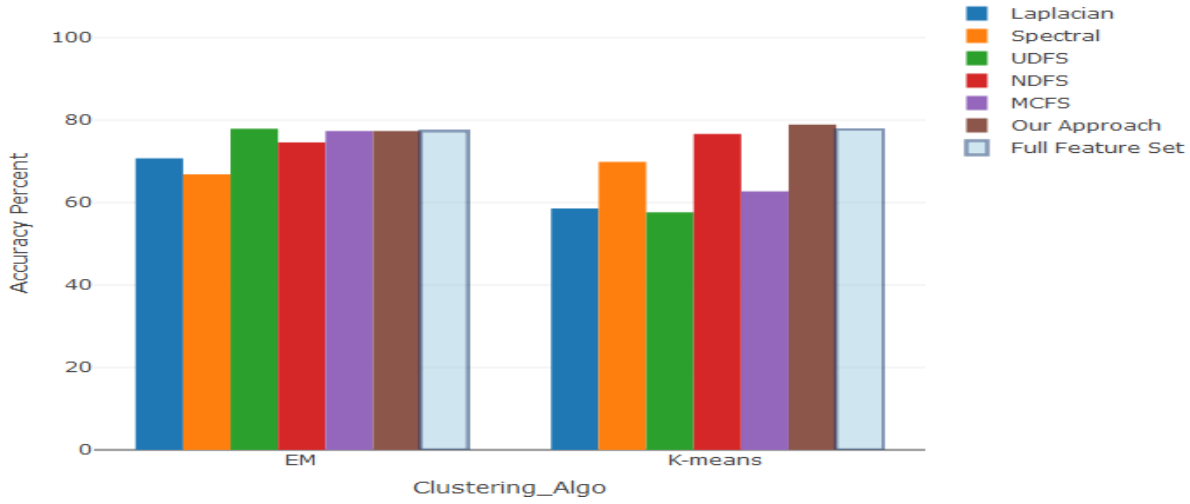


Figure A.30: Clustering Accuracy (Spectrometer Dataset)

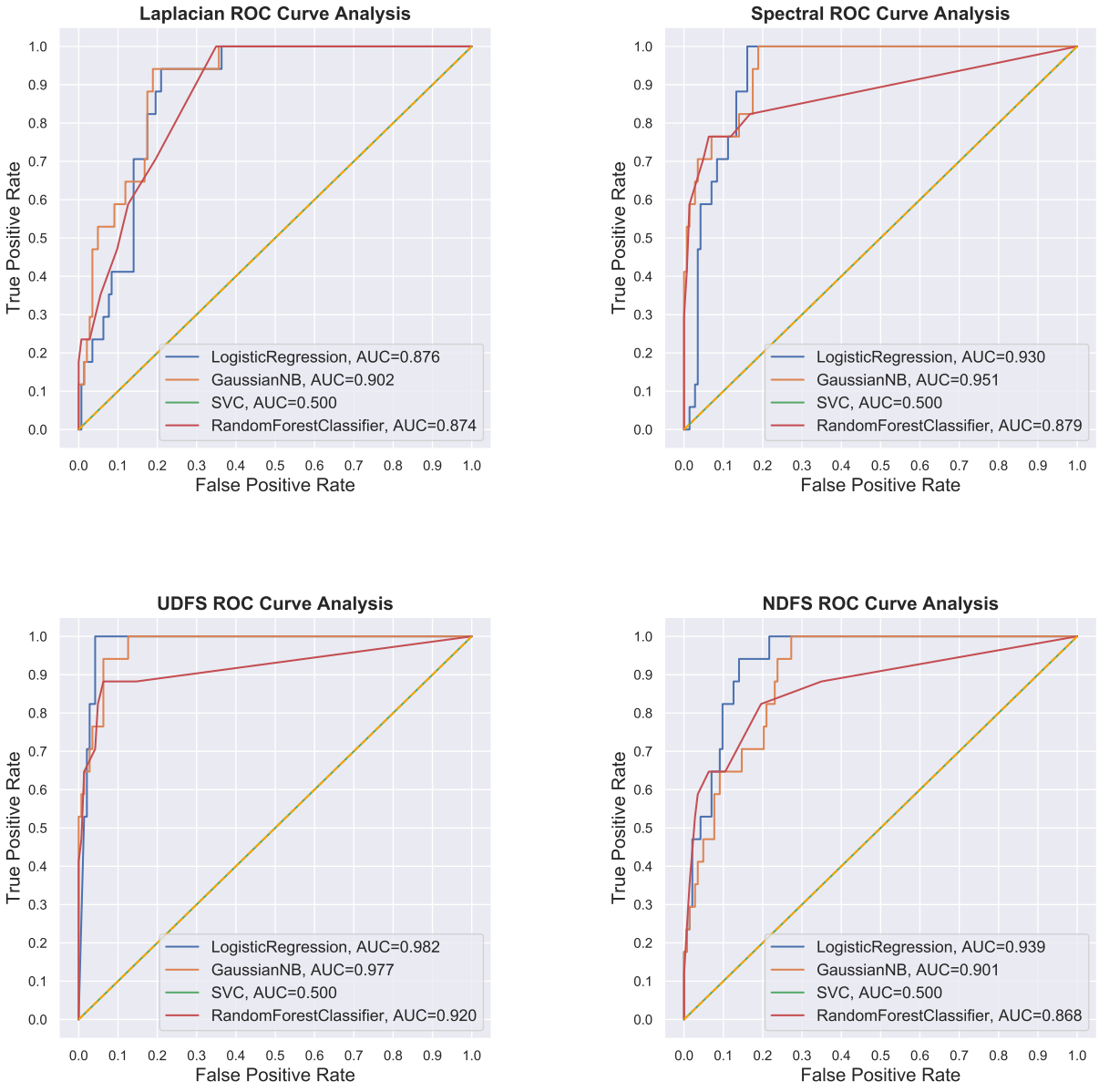


Figure A.31: ROC Curves (Spectrometer Dataset) 1

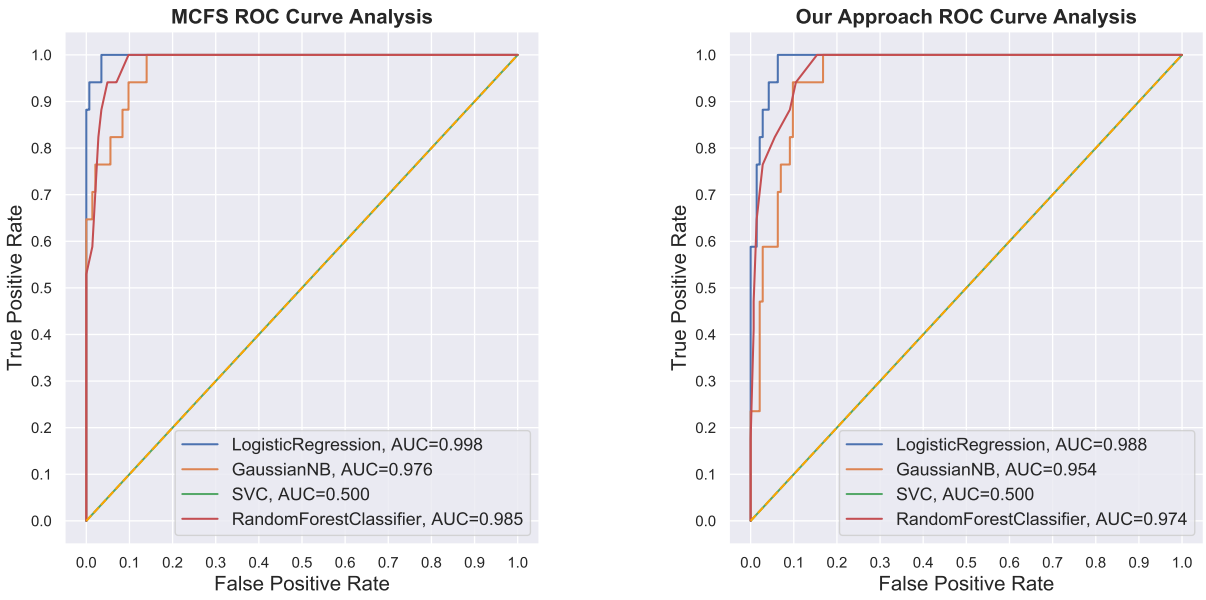


Figure A.32: ROC Curves (Spectrometer Dataset) 2

	Laplacian	Spectral	UDFS	NDFS	MCFS	Our Approach
Rep Entropy	0.28753	-	0.37981	0.44524	0.43421	0.4492

Table A.8: Representation Entropy (Spectrometer Dataset)

Appendix B

Abbreviations

PCA Principal Component Analysis

LDA Linear Discriminant Analysis

LLE Locally Linear Embedding

EM Expectation-Maximization

MLE Maximum Likelihood Estimator

MST Minimum Spanning Tree

KNN K-Nearest Neighbour

MIC Maximal Information Coefficient

CLIQUE Clustering In QUEst

MAFIA Merging Adaptive Finite Intervals Algorithm

PROCLUS PROjected CLUStering

COSA Clustering On Subsets of Attributes

FSSEM Feature Subset Selection using Expectation Maximization

BBHFS Boosting Based Hybrid Feature Selection

MCFS Multi-Cluster Feature Selection

LAR Least Angel Regression

MINE Maximal Information-based Non-parametric Exploration

MICE Multivariate Imputation by Chained Equation

MAS Maximum Asymmetry Score

MEV Maximum Edge Value

IDE Integrated Development Environment

ACC Clustering Accuracy

AUC Area Under The Curve

ROC Receiver Operating Characteristics