ÇUKUROVA UNIVERSITY
INSTITUTE OF NATURAL AND APPLIED SCIENCES

MSc THESIS

Madaa ALHAJI

A STUDY ON SOME DIMENSIONALITY REDUCTION ALGORITHMS IN MACHINE LEARNING

DEPARTMENT OF STATISTICS

ADANA, 2022

ABSTRACT

MSc THESIS

A STUDY ON SOME DIMENSIONALITY REDUCTION ALGORITHMS IN MACHINE LEARNING

Madaa ALHAJI

ÇUKUROVA UNIVERSITY
INSTITUTE OF NATURAL AND APPLIED SCIENCES
DEPARTMENT OF STATISTICS

Supervisor : Prof. Dr. Sadullah SAKALLIOĞLU
Year: 2022, Pages: 77
Jury : Prof. Dr. Sadullah SAKALLIOĞLU
: Prof. Dr. Güzin YÜKSEL
: Prof. Dr. Turgay İBRİKÇİ

In recent decades, there has been an explosion in the amount of data in most fields of life and science, especially in the practical sciences like physics, chemistry, biology, and astronomy. The data coming from experiments and laboratory devices are becoming increasingly complicated, and reports hundreds or thousands of measurements for a single experiment, and that makes statistical approaches facing difficulty in dealing with such high-dimensional data. This not only does make processing extremely slow, but it can also make it much harder to find a good solution. However, much of the data is redundant, and it is possible to reduce the number of variables to a manageable level without losing too much information. Dimensionality reduction techniques are mathematical procedures that enable this reduction; they have been widely developed in fields such as statistics and machine learning, and are now a hot research topic. In this thesis, we discussed dimensionality reduction, the common approaches used for dimensionality reduction, and then went through four of the most popular dimensionality reduction techniques: PCA, Multidimensional scaling, LLE, and Isomap. We applied these algorithms to real and artificial data sets and discussed the results for a better understanding of these algorithms.

Key Words: Dimensionality reduction, Machine learning, PCA, Multidimensional Scaling, Isomap, LLE.

ÖZ

YÜKSEK LİSANS TEZİ

MAKİNE ÖĞRENMESİNDE BAZI BOYUT İNDİRGEME ALGORİTMALARININ İNCELENMESİ

Madaa ALHAJI

ÇUKUROVA ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ
İSTATİSTİK ANABİLİM DALI

Danışman : Prof. Dr. Sadullah SAKALLIOĞLU
Yıl: 2022, Sayfa: 77
Jüri : Prof. Dr. Sadullah SAKALLIOĞLU
: Prof. Dr. Güzin YÜKSEL
: Prof. Dr. Turgay İBRİKÇİ

Son yıllarda, özellikle fizik, kimya, biyoloji ve astronomi gibi uygulamalı bilimlerde, yaşamın ve bilimin birçok alanında veri miktarında bir patlama olmuştur. Deneylerden ve laboratuvar cihazlarından gelen veriler giderek daha karmaşık hale geliyor ve tek bir deney için yüzlerce, binlerce ölçüm rapor ediliyor ve bu da istatistiksel yaklaşımların bu tür yüksek boyutlu verilerle uğraşmasını zorlaştırıyor. Bu yalnızca işlemeyi aşırı derecede yavaşlatmakla kalmaz, aynı zamanda iyi bir çözüm bulmayı da çok daha zor hale getirebilir. Bununla birlikte, verilerin çoğu gereksizdir ve çok fazla bilgi kaybetmeden değişkenlerin sayısını yönetilebilir bir düzeye indirmek mümkündür. Boyut indirgeme teknikleri, bunu sağlayan matematiksel işlemlerdir; istatistik ve makine öğrenimi gibi alanlarda geniş çapta geliştirilmiş ve şu anda sıcak bir araştırma konusu olmuştur. Bu tezde boyut indirgeme konusunu ele aldık ve boyut indirgeme için yaygın olarak kullanılan yaklaşımları tartıştık ve ardından en popüler dört boyut indirgeme algoritmalarından dördünü inceledik: PCA, Çok boyutlu ölçekleme, LLE ve Isomap. Bu algoritmaları gerçek ve yapay veri kümelerine uyguladık ve bu algoritmaların daha iyi anlaşılması için sonuçları tartıştık.

.

Anahtar kelimeler : Boyut indirgeme, Makine öğrenme, PCA, Çok boyutlu ölçekleme, Isomap, LLE.

EXPANDED ABSTRACT

This study examines some methods, approaches, and algorithms of dimensionality reduction that are used in machine learning. There is a huge amount of data that is produced daily in various areas of life and experimental science fields such as physics, chemistry, natural sciences, and astronomy. In general, classical statistical analysis methods are used in order to process and analyze small data, but when dealing with large data, it may become difficult to process and analyze this data with classical statistical analysis methods. And here appears the importance of machine learning algorithms and data mining techniques which can be used to deal with such data and process it more easily.

Before discussing the issue of dimensionality reduction, some basic definitions related to machine learning, and data mining, were given. In addition to that learning algorithms which include supervised learning, unsupervised learning, semi-supervised learning, and reinforcement learning were explained. The importance of these algorithms and their uses for regression, classification, clustering, and other purposes have been discussed in the first chapter of the thesis. However, some machine learning problems may involve hundreds or even thousands of features for every training set. This does make training slow, and also makes determining a solid solution much more hard. This thing is referred to as "the curse of dimensionality" or the "Hughes effect". Bellman(1957) was the first one to use this expression when thinking about difficulties that he was facing while working on problems in the field of dynamic programming.

Dimensionality reduction may be defined as the transfer of data from a high-dimensional space to a low-dimensional space in such a way that the low-dimensional representation preserves some essential features of the original data, typically similar to its fundamental dimension (Van Der Maaten et al., 2009). Two basic methods are used for reducing dimensionality which are feature extraction and feature selection. In feature selection, the goal is to find the k of the d

dimensions(where $k < d$) that have most of the information in the data while ignoring the remaining (d-k) dimensions. But in feature extractions the goal is to find a new set of k dimensions that are the outcome of the original d dimensions being combined, i.e., creating new more valuable ones. The choice between feature selection or feature extraction methods is influenced by the unique type of data for the application field. There are two other approaches related to dimensionality reduction algorithms which are projection and manifold learning approaches. The methods and approaches to dimensionality reduction have been presented in the third chapter of the thesis.

The importance of dimensionality reduction comes from that it helps to eliminate data that is redundant, irrelevant, or noisy. It also reduces the complexity of most learning algorithms which is determined by the number of features, d, and the size of the data set, n. Thus, we are concerned about lowering the problem's dimensionality to get less storage space and faster processing. Lowering d during the testing stage will make the inference process easier too. In addition to that simple models are much more effective on small data sets. That is the variance will be much less on simple models, which means that these models will change less depending on the amount of noise, outliers,...etc found in the sample. Reducing the dimensionality of data may help us to more clearly represent and analyze the structures. patterns and outliers because visualizing data in greater dimensions is difficult.

In the fourth chapter of the thesis, detailed information, and discussion of the most common algorithms used for dimensionality reduction, which are principal components analysis(PCA), multidimensional scaling(MDS), Isometric mapping(Isomap), and locally linear embedding(LLE), have been presented. The advantages and limitations of the algorithms and the proposed methods for avoiding these limitations have been explained too.

In the last chapter, the above-mentioned algorithms have been applied on two real data sets, which are the famous breast cancer data set and driving distances between 15 Turkish cities, and two artificial data sets, which are the data sphere and S-curve data sets.

The performance and execution time of each algorithm when applied on different datasets have been examined and compared. As a result, it can be said that the different algorithms gave similar results on some data sets and different results on other data sets based on the type of data sets. It can also be said that it is useful to apply more than one-dimensionality reduction algorithm on the given high-dimensional data set and compare the results of these algorithms in order to find the one that gives the best results for the data.

GENİŞLETİLMİŞ ÖZET

Bu çalışma makine öğrenmesinde kullanılan bazı boyut indirgeme yöntemlerini, yaklaşımlarını ve algoritmalarını incelemektedir. Fizik, kimya, biyoloji, astronomi gibi çeşitli deneysel bilim alanlarında ve farklı yaşamsal alanlarda günlük olarak üretilen çok büyük miktarda veriler bulunmaktadır. Genel olarak küçük verileri işlemek ve analiz etmek için klasik istatistiksel analiz yöntemleri kullanılır ancak büyük veriler söz konusu olduğunda bu verileri klasik istatistiksel analiz yöntemleri ile işlemek ve analiz etmek zorlaşabilir. Burada da bu tür verilerle başa çıkmak ve onu daha kolay bir şekilde işlemek için kullanılabilecek makine öğrenme algoritmalarının ve veri madenciliği tekniklerinin önemi ortaya çıkmaktadır.

Boyut indirgeme konusunu ele almadan önce, makine öğrenme ve veri madenciliğiyle ilgili temel tanımlar verilmiştir. Bununla birlikte denetimli öğrenme, denetimsiz öğrenme, yarı denetimli öğrenme ve pekiştirmeli öğrenme olarak ayrılan makine öğrenmesi algoritmaları da ele alınmıştır. Bu algoritmaların önemi ve regresyon, sınıflandırma, kümeleme ve diğer amaçlar için kullanımları tezin ilk bölümünde tartışılmıştır. Ancak bazı makine öğrenme problemleri her eğitim seti için yüzlerce hatta binlerce değişken içerebilir. Bu eğitimi yavaşlatır ve aynı zamanda sağlam bir çözüm bulmayı çok daha zor hale getirir. Buna "boyutsallığın laneti" veya "Hughes etkisi" denir. Bellman (1957), dinamik programlama alanındaki problemler üzerinde yaptığı çalışmalarda zorlukları düşünürken bu ifadeyi kullanan ilk kişi olmuştur.

Boyut indirgeme, yüksek boyutlu bir uzaydan düşük boyutlu bir uzaya, düşük boyutlu temsilin orijinal verinin bazı temel özelliklerini tipik olarak temel boyutuna benzer şekilde koruyacak şekilde veri transferi olarak tanımlanabilir (Van Der Maaten et al., 2009). Boyut indirgeme için değişken seçme ve değişken çıkarma olmak üzere iki temel yöntem kullanılmaktadır. Değişken seçiminde amaç, kalan (*d-k*) boyutları yok sayarak, verilerdeki bilgilerin çoğuna sahip olan d

boyutlarının (burada $k < d$) k tanesini bulmaktır. Ancak değişken çıkarımında amaç orijinal d değişkenlerin birleştirilmesinin sonucu olan yeni bir k boyutlu değişken kümesi bulmak, yani daha değerli yeni değişkenler üretmektir. Değişken seçme veya değişken çıkarma yöntemleri arasındaki seçim, uygulama alanındaki verinin özgün türünden etkilenir. Boyut indirgeme algoritmaları ile ilgili projeksiyon ve manifold öğrenme olarak bilinen iki yaklaşım daha vardır. Boyut indirgeme yöntemleri ve yaklaşımları ise tezin üçüncü bölümünde sunulmuştur.

Boyut indirgemesinin önemi, gereksiz, alakasız veya gürültülü verilerin ortadan kaldırılmasına yardımcı olmasından gelir. Ayrıca boyut indirgeme değişkenlerin sayısı d ve veri setinin boyutu n ile belirlenen çoğu öğrenme algoritmasının karmaşıklığını da azaltır. Bu nedenle, problemin boyutluluğunu azaltmak, daha az depolama alanı ve daha hızlı işlem elde etmek için önemlidir. Test aşamasında d 'yi düşürmek, çıkarım sürecini de kolaylaştıracaktır. Bununla birilikte basit modeller küçük veri setlerinde çok daha etkilidir. Yani basit modellerde varyans çok daha az olacaktır, bu da bu modellerin örneklemde bulunan gürültü, aykırı değer vb. miktarına bağlı olarak da daha az değişeceği anlamına gelir. Verilerin boyutluluğu azaltmak, yapıları, biçimleri ve aykırı değerleri daha net bir şekilde temsil etmemize ve analiz etmemize yardımcı olabilir çünkü verileri daha büyük boyutlarda görselleştirmek zordur.

Tezin dördüncü bölümünde, temel bileşenler analizi (PCA), çok boyutlu ölçekleme (MDS), izometrik gömülme (Isomap) ve yerel lineer gömülme (LLE) olan boyut indirgemesi için kullanılan en yaygın algoritmaların ayrıntılı bilgileri ve tartışması yer almaktadır. Algoritmaların avantajları ve sınırlamaları ile bu sınırlamalardan kaçınmak için önerilen yöntemler de açıklanmıştır.

Son bölümde, yukarıda belirtilen algoritmalar, ünlü meme kanseri veri seti ve Türkiye'nin 15 şehri arasındaki sürme mesafeleri olan iki gerçek veri seti ile küre veri seti ve S-eğrisi veri seti olan iki yapay veri seti üzerinde uygulanmıştır. Her bir algoritmanın farklı veri setleri üzerinde uygulandığında performans ve yürütme süresine göre incelenmiş ve karşılaştırılmıştır. Sonuç olarak farklı

algoritmaların veri setinin türüne göre benzer ve farklı sonuçlar verdiği söylenebilir. Ayrıca verilen yüksek boyutlu veri seti üzerinde birden fazla boyut indirgeme algoritmasının uygulanmasının ve bu algoritmaların sonuçlarını karşılaştırmanın verilere en iyi sonucu veren algoritmayı bulmak için faydalı olduğu da söylenebilir.

ACKNOWLEDGEMENTS

First and foremost I would like to thank my supervisor, Prof. Sadullah Sakallıoğlu for his invaluable advice, continuous support, and guidance during my master's study.

Secondly, I would like to express my sincere gratitude to all the teaching staff in the Department of Statistics, whom I benefited a lot from their abundant knowledge and experience.

Last but not least, I would like to thank my dear family, wife and children who always gave me unlimited support and encouraged me to continue my higher studies.

CONTENTS **PAGES**

LIST OF FIGURES **PAGES**

LIST OF TABLES **PAGES**

LIST OF ABBREVIATIONS

Isomap	: Isometric mapping
LLE	: Locally linear embedding
MDS	: Multidimensional scaling
ML	: Machine learning
PCA	: Principal components analysis

1. INTRODUCTION

There is a huge amount of data that is produced daily in various areas of life. This data may be numbers, letters, words, images, sounds, etc. There are many methods and algorithms used in statistics, machine learning, and data mining for analyzing these data and extracting useful information from them. But when the data is in high dimensional space, or when the number of variables is large and there is a huge amount of data, then processing this data may become exhausting, complicated, or meaningless. And this large number of dimensions then turns into a curse called the curse of dimensionality. Therefore, dimensionality reduction methods are used as an initial step in reducing the dimensionality of the data before starting to process it. When we can characterize data with fewer features, we have a better understanding of the process that generates the data and can extract knowledge. These fewer characteristics could be seen as hidden or implicit features when selected or combined produce the wanted characteristics of the data. Dimensionality reduction methods take a high-dimensional dataset and turn it into a low-dimensional dataset while preserving the original characteristics as much as possible.

In this thesis, we are going to deal with the topic of dimensionality reduction and try to explain the methods and approaches used in it and we are going to explain four of the most widely used algorithms for this purpose. The algorithms that we are going to discuss are principal components analysis, multidimensional scaling, Isomap, and locally linear embedding.

These algorithms are very widely used in machine learning. Some of these algorithms are supervised and the other some are unsupervised. For a better understanding of dimensionality reduction, we have to give some basic definitions related to machine learning and basic learning algorithms which are supervised learning, unsupervised learning, semisupervised, and reinforcement learning.

2. BASIC DEFINITIONS

2.1. Machine Learning

Machine learning, especially the powerful machine learning (ML) techniques we know today, was not invented by a single person. It was developed with the help of many brilliant people. Alan Turing (1950) in his published paper "Computing Machinery and Intelligence" had asked the question "Can machines think?" The "Imitation Game," as described in the paper, comprises three players: human acting as a judge, a second human, and a computer attempting to persuade the judge that it is human. To "speak" to the other two participants, the judge would input into a terminal software. After both the human and the computer answered, the judge would determine which answer came from the machine. If the judge could not consistently discriminate between human and computer answers, the machine won the game. When we talk about machine learning we should also mention Arthur Samuel, an IBM computer scientist and pioneer in artificial intelligence and computer gaming, who created the term "machine learning." That's when he came up with the idea of a computer program for playing checkers in 1959. Owing to a minimax algorithm for evaluating movements to come up with winning strategies, the more the program played the game, the more it gained from its experience. According to Arthur Samuel (1959) earning is the branch of science that studies how computers can learn without being explicitly programmed.

Another definition for machine learning was given by Tom M. Mitchell " A computer program is said to learn from experience E with respect to some class of tasks *T* and performance measure *P*, if its performance at tasks in *T*, as measured by *P*, improves with the experience *E*. " (Mitchell, 1997). As a result, we can say that Machine Learning is the science and art of programming computers so they can learn from data. (Géron, 2017)

Machine learning is among the most rapidly growing fields of computer science. It is not only the data that is increasing massively but so is the

methodology for handling it and turning it into meaningful knowledge. Not just in numerous domains of science, but also in ordinary life, as modern technology gradually invades our daily existence and our digital imprint deepens, so much data is produced and gathered. Data that stays latent passively, whether it is scientific or private, serves no useful purpose, and smart people have been developing ever new techniques to utilize that data and transform it into beneficial items or services. Machine learning is becoming increasingly important in this change. Every year, datasets grow in size. İt is not just the number of data points that have increased, but so does the number of observed features. Data now contains more structures and patterns than just numbers or letters strings: photos, videos, sounds, documents, web pages, charts, diagrams, and so on. The data is moving away from the statistical assumptions that were supposed to be achieved, such as normality. (Alpaydın, 2014)

We create data every time we purchase something, hire a movie, visit a website, write a blog, or share on social media, and even when we simply walk or drive around. Let's Think of a supermarket chain that offers thousands of goods to millions of people through hundreds of physical centers across the country, or an online store on the internet. The information of every transaction is stored. This information may include the date, customer Identity, goods bought and their prices, total money being spent, etc. This generates a tremendous amount of data every day. To maximize sales and profits, the supermarket chain needs to be able to predict which customers are most inclined to buy certain goods. Accordingly, each customer seeks to find the product that best fulfills his or her needs. Data mining refers to the use of machine learning techniques for large datasets. This phase is known as mining since it resembles what occurs in mines when enormous quantities of soil and raw material are collected to yield a limited amount of valuable materials. In data mining, a large number of data is analyzed in order to produce a basic practical model.

For solving a problem on a computer, we need to have an algorithm, which is a collection of instructions that must be executed to transform input into output. For example, one could build an algorithm for sorting. In this algorithm, The input is a set of numbers, and the output is an ordered set of those numbers. There could be different algorithms for one task, and we may be interested in selecting the most effective one, one that requires the least instructions or storage space, or both. However, we lack an algorithm for some jobs, such as forecasting customer behavior or discriminating between spam and legitimate emails, i.e., we would like to make a classification. Machine learning, on the other hand, is a part of artificial intelligence as well as a database problem. To be intelligent, a system in a changing environment must be able to learn. In case the system can learn and adjust to new situations, then the system developer does not have to predict and respond to all possible circumstances.

According to Géron (2017), Machine Learning is ideal for :

- Problems that demand a lot of human tuning or long lists of rules, where one Machine Learning technique may frequently make code more simple and perform better.
- Using the latest machine learning approaches, for solving complex problems that can not be solved using traditional approaches.
- In changing settings, a machine learning model may adjust to new data.
- Obtaining insights about difficult issues using big amounts of data.

2.2. Machine Learning Algorithms

There are numerous algorithms available for use in machine learning. But here are four types of learning which are supervised learning, unsupervised learning, semi-supervised learning, and reinforcement learning. We will give some basic information on each of them.

2.2.1. Supervised Machine Learning

The existence of an output variable to assist the learning process gives it the name "supervised ". This learning approach uses a training dataset to train models to produce the desired output. This training dataset comprises both true and false outputs, which allows the model to develop over time. The loss function is used to assess the accuracy of the algorithm and is changed until the error is sufficiently reduced. When it regards data mining, there are two types of supervised learning which are regression and classification. (Alpaydın, 2014). Regression is a way to analyze the relationship between input and output variables in order to predict a desired numeric value, like car price, given a set of features called predictors (for example age, miles, model, and so on). In order to train the model, we must provide it with numerous examples of cars, giving their predictors and labels (i.e., the price of the cars). Regression is commonly used for prediction, such as a company's sales revenue. Linear regression, polynomial regression, and logistic regression are all popular regression techniques. We can use supervised learning techniques to develop and improve a variety of business applications, such as image and object recognition, spam filtering customer sentiment analysis, and so on. The algorithm of classification is based on allocating test data to certain groups. It identifies specific elements in the dataset and attempts to draw insights about how they're being labeled or identified. Some of the most popular classification algorithms are, Nave Bayes, k-nearest neighbor, decision trees, and vector machines. A good illustration of this is the spam filter: it

is trained given a large number of example emails and their classifications, and then it must be able to classify the new emails like it is illustrated in Figure 2.1.

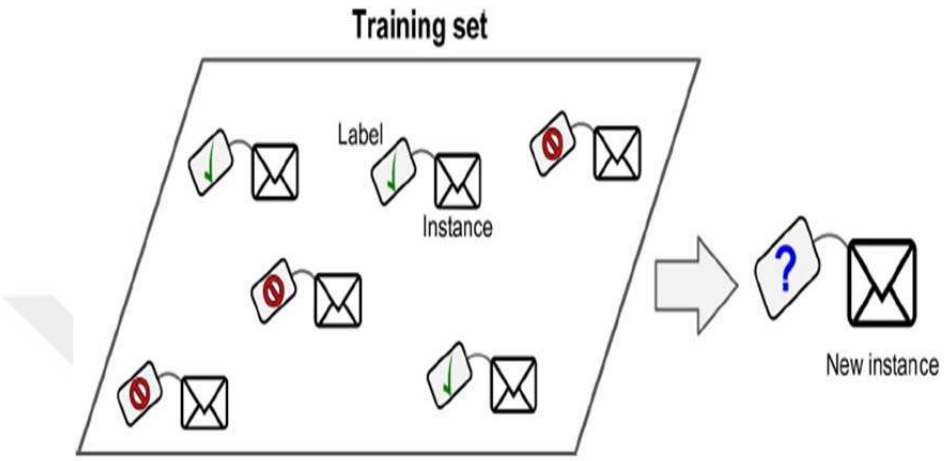

Figure 2.1. A labeled training set for supervised learning (Géron, 2017)

2.2.2. Unsupervised Machine Learning

Unsupervised learning, in contrast to supervised learning, uses unlabeled data. It extracts data patterns and employs them for solving dimensionality reduction, clustering, visualization, and association problems. This is especially helpful when we are unsure of common characteristics within a data set. Unsupervised machine learning employs machine learning methods to analyze and cluster unlabeled datasets. These algorithms discover hidden structures or data groupings without any need for human interference. Because of its ability to detect differences and similarities in information, it is ideal for exploratory data analysis, cross-selling techniques, customer segmentation, and image recognition.

Some of the most essential unsupervised learning methods for the purpose of clustering are k-Means, hierarchical cluster analysis, and expectation-maximization methods. For visualization and dimensionality reduction principal component analysis, kernel PCA, and LLE are the most widely used methods. In the fourth chapter, the PCA and LLE algorithms will be thoroughly examined.

Running a clustering algorithm to attempt to find groups of similar visitors among a large amount of data about your blog's visitors is a typical example of clustering. You never inform the algorithm which category a visitor falls in; it figures it out itself. For example, it may be noted that 40% of your visitors are males who like comic books and read your site in the evenings, while 20% are young sci-fi fans who visit it on weekends, and so on. Each group can be divided into smaller groups when employing a hierarchical clustering technique. This may help you tailor your posts to each group (Géron, 2017).

Dimensionality reduction is related to unsupervised learning where the goal is to simplify the data without losing too much information. One of the ways for doing this is through merging many correlated features. For example, if a car's mileage and age are highly correlated, then the dimensionality reduction method will combine them into a single feature that describes the car's wear and tear. This is known as feature extraction. It will be covered in Chapter 3. Visualization techniques are also ideal examples of unsupervised learning algorithms since they take a large amount of unlabeled, complex data and provide a simple two-dimensional or three-dimensional representation of it. These methods attempt to preserve as much structure as possible. So that you can see how the data is organized and potentially notice patterns and structures that you didn't expect.

2.2.3. Semi-Supervised Learning

Semi-supervised learning is a hybrid of supervised and unsupervised learning. The algorithm is given some supervision information in addition to unlabeled input — but not always for all examples. Frequently, the targets linked with some of the examples will be the information standard setting. The data of semi-supervised learning set $X = (x_i)$, i =*1,2, ...,n*, may be separated into two portions in this case. the points $X_l = (x_1, x_2 \ldots . x_l)$ for which labels $Y_l = (y_1, y_2 \ldots . y_l)$ are provided, and the points $X_u = (x_{l+1}, x_2 \ldots . x_{l+u})$ for which labels are unknown,i.e. during the training stage, a mix of labeled and unlabeled datasets

are used (Chapelle et al., 2009). Semi-supervised learning involves training an initial model on a few labeled samples and then applying it iteratively to a larger set of unlabeled data.. Google Photos and other photo-hosting services are wonderful instances of this. When you send all of your family photographs to the service, it will detect that person A is in images 1, 5, and 11, whereas person B is in photos 2, 5, and 7. This is the unsupervised part of the algorithm (clustering). All that is left for you to do now is to identify these persons. It just to give one label to each person to identify everyone in each photograph, which is wonderful for searching. (Géron, 2017)

2.2.4. Reinforcement Learning

The system output in certain applications is a succession of actions. In this scenario, a single action is unimportant; what matters is the policy, which is the series of correct activities that leads to the desired outcome. In any intermediate state, There is no optimal action; an action is excellent in case it is part of a successful strategy. To develop a strategy, in this case, the machine learning algorithm should be able to assess the quality of the strategy and learn from prior excellent action sequences. A common example of reinforcement learning is playing games when a single correct move is not as essential as the succession of correct moves. If a move is part of a good game-playing policy, then it is a good move.

2.3. Methods for Evaluating Machine Learning Models

2.3.1. Hold-out Method

In this method, the dataset is divided into training and test sets for model evaluation. First, we train the model using the training set and after that, we test it using the test set to attain the most optimal model. This method is generally applied when the data collection is tiny and there is insufficient data to be divided into training, validation, and training sets. This method has the advantage of being easy to execute, but it is dependent on how the data is divided into two sets. If the

division is not random, the result may be biased. When using the hold-out method, a common division is to use 70 % of the data for training and the remaining 30 % for testing. Overall, the hold-out model evaluation method is a decent way to start when developing machine learning models, but it should be used with caution.

2.3.1. Cross-Validation Method

Cross-validation is a statistical method that is used for estimating the performance or accuracy of a machine learning model and evaluating a machine learning model's ability to predict new data. It's used to prevent a prediction model from overfitting, especially when the amount of data supplied is limited. It can also be used to detect problems like selection bias, as well as provide information on how the model can be generalized to a different dataset. In the cross-validation method we make a set number of folds (or partitions) of the data, execute the analysis on each fold, and then average the overall error estimate in cross-validation. There is a few types of cross-validation method in machine learning but the most commonly used method is *K*-fold cross-validation. This strategy ensures that our model's score is independent of how we chose the training and testing sets. In K-fold cross-validation, we randomly divided the dataset *X* into *K* equal pieces. To create each pair, we choose one of the K parts as the validation set and combine the remaining *K* -1 parts to form the training set. By repeating this method *K* times, we have *K* pairs. Each time, one of the *K* parts is left out, and the holdout method is applied to each subset *K* times. There are two problems with this strategy. The first one is that we allow for very small validation sets in order to maintain a large training set. The second one is that the training sets are significantly overlapping, with *K*-2 parts shared by any two training sets. *K* generally ranges from 10 to 30. As *K* increases, the percentage of training examples grows, and we get more robust estimators, while the validation set decreases.

Furthermore, there is the expense of training the classifier K times, which grows as K grows. As the sample size N grows, K can be reduced; if N is small, K should be big to provide enough training sets. (Alpaydın,2014).

3. DIMENSIONALITY REDUCTION

3.1. The Curse of Dimensionality

For every training set, some machine learning problems may include hundreds or maybe thousands of variables. This does make training slow, and also makes determining a solid solution much more hard. This thing is known as the curse of dimensionality or the Hughes effect. Bellman (1957) was the first one to use this expression when thinking about difficulties that he was facing while working on problems in the field of dynamic programming.

Luckily, in actual problems, it is frequently possible to lower the number of features, converting an unsolvable problem into a solvable one. Reducing dimensionality cause the loss of some information. Therefore while it is expected to allow faster training, this may also lead the system to work worse. It also complicates plans, making maintenance more complex. When training takes a too long time, we should first start training our model with the original data before trying dimensionality reduction. But, in some situations, reducing the dimensionality of the training data may cause some noise and unneeded features, resulting in improved performance. But in general, it will not; it will simply accelerate training. (Géron, 2017)

Dimensionality reduction may be defined as the transfer of data from a high-dimensional space to a low-dimensional space in such a way that the low-dimensional representation preserves some essential features of the original data, typically similar to its fundamental dimension (Van Der Maaten et al., 2009). The reduction of dimensionality is very important in data visualization. When the number of dimensions is lowered to two or three, then it will be possible to display a high-dimensional training dataset on a graph and often get crucial insights by visually discovering structures and patterns in data, like clusters.

3.2.The Advantages of Dimensionality Reduction

Applying dimensionality reduction techniques to the data as a preprocessing step can provide several advantages such that :

- Data that is redundant, irrelevant, or noisy can be eliminated.
- We save the expense of extracting an input when it is determined to be unnecessary.
- The complexity of most learning algorithms is determined by the number of input variables, d, and the size of the data set, n. Thus, we are concerned about lowering the problem's dimensionality for less storage space and faster processing. Lowering d during the testing stage will make the inference process easier.
- Simple models are much more effective on small datasets. That is the variance will be much less on simple models, which means that these models will change less depending on the amount of noise, outliers, etc., found in the sample.
- Lowering the dimension may help us to more clearly represent and analyze the structures. patterns and outliers because visualizing data in greater dimensions are difficult.

3.3. Methods of Dimensionality Reduction

We can say that there are two basic methods that are used for reducing dimensionality which are feature extraction and feature selection. The choice between feature extraction or feature selection methods is influenced by the unique type of data for the application field.

3.3.1. Feature Selection

The goal of feature selection (also known as subset selection) is to find the k of the d dimensions(where $k < d$) that have most of the information in the data while ignoring the remaining (d-k) dimension, — in other words, choosing the most important features within existing features to train on. There are various potential benefits of variable and feature selection, including better visualization and understanding of the data, lowered measurement and storage requirements, shorter training times, and overcoming the curse of dimensionality to increase prediction performance. Due to the large amount of noisy, unnecessary, or deceptive features in several actual problems, there is a great necessity for using feature selection.

Feature selection is generally referred to as a searching problem based on certain performance standards. According to Ladha and Deepa (2011), feature selection algorithms can be characterized by

I. Search organization: There are three kinds of search that are sequential, exponential, and random.
II. Successor generation: There are five different ways that can be used to construct a successor. These are backward selection, forward selection, weighted selection, compound selection, and random selection.
III. Measures for evaluation: As illustrated in Figure 3.1, divergence, the probability of errors, dependence, Inter - class distance, information or uncertainty, accuracy, and consistency evaluation can be used to evaluate the goodness of successors.

Gene selection from microarray data and text classification are two of the most common uses of feature selection.

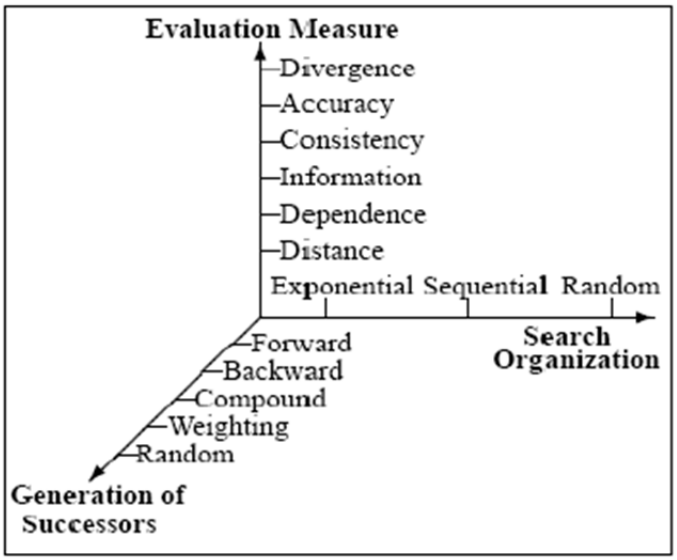

Figure 3.1. Characterization of feature selection algorithms (Ladha & Deepa, 2011)

In literature, feature selection methods are basically divided into three groups that are filter, wrapper, and embedded methods (Li et al., 2017).

1. Filter Method

It is possible to employ the filter approach before further processing of the data. No matter what technique is used to develop and fine-tune the model (such as a predictive model), the variables used as inputs have no effect on this. In general, filters are faster to compute, but they are vulnerable to overfitting. The relationship between the model's outputs and inputs is used to identify a collection of key variables. Statistical tests are used to classify each input based on its importance to the output. To guarantee that the model runs fast, the filter technique has a minimal computational cost. Using the filter technique, however, has the disadvantage of being unable to optimize the model picked in the learning machine because it is independent of the method used to adjust or develop the model given the variables selected as inputs. Figure 3.2 shows a general filter method scheme.

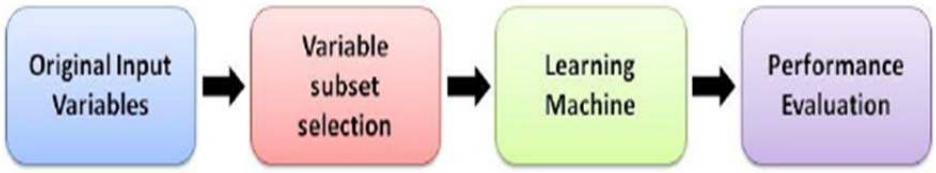

Figure 3.2. Generic scheme of filter methods (Cateni et al., 2012)

2. Wrapper Method

By the year 1997, Kohavi and John came up with the idea for this method. An important component of the method is to utilize a learning machine's prediction performance (or classification accuracy) to determine how beneficial a particular group of characteristics are to the overall system. Wrapper techniques approach machine learning as a black box when picking subsets of variables during the preprocessing phase. When compared to the filter approach, the wrapper method is more time-consuming in terms of computing. On the Opposite to this, wrapper techniques are simple and general when viewed in the context of the learning machine's black box. An exhaustive search becomes too expensive when the number of variables becomes too high. When a dataset includes k variables, for instance, then there are 2^K possible subsets to evaluate, which means 2^K the learning procedure must be executed. Figure 3.3 is a generic diagram of the wrapper method diagram.

According to Rangarajan (2010), wrapper techniques tend to perform better than filter methods since the feature selection method is tuned for the classification algorithm to be applied. And since every feature set has to be assessed using the classification algorithm, then if the number of features is large, wrapper methods are typically too costly to employ compared to filter approaches. Filter methods have less computation costs, and more faster but less efficient, which makes them better suited for high-dimensional datasets.

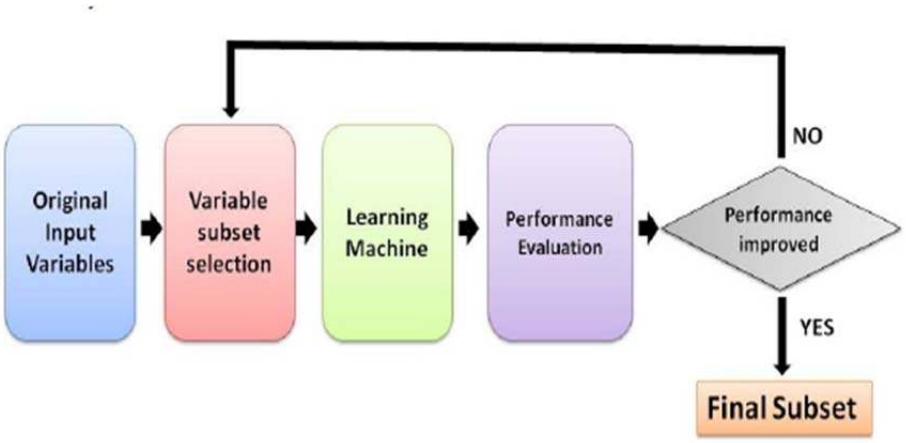

Figure 3.3. Generic diagram of wrapper method (Cateni et al., 2012)

3. Embedded Method

Unlike filter and wrapper methods, the embedded approach uses a machine learning model to choose variables. Variables are chosen during the training phase, decreasing computational costs and increasing efficiency during the variable selection stage. The distinction between embedded and wrapper approaches is not always evident, but the fundamental distinction is that embedded method involves iterative updates and then rates the relevance of each feature based on how much it contributed to the ML model training (eg. LASSO Regularization). The performance of the model under consideration determines how the model's parameters evolve. Wrapper techniques, on the other hand, only analyze the model's performance based on the variables selected. Figuratively, the embedded technique is shown in Figure 3.4.

Figure 3.4. Generic scheme of embedded method(Cateni et al., 2012)

3.3.2. Feature Extraction

The goal behind using this method is to find a new set of k dimensions that are the outcome of the original d dimensions being combined, i.e., creating new more valuable ones. According to Brian Ripley (1996), "feature extraction" can be described as the process of creating linear combinations $\alpha^T x$ of continuous features that have high discriminatory strength between different classes. One of the drawbacks of feature extraction is that information regarding the relative importance of each original feature can be lost during feature extraction because of the lack of interpretability in the linear combination of the original features. (Ladha & Deepa, 2011). A famous and commonly used feature extraction method is Karl Pearson's principal component analysis method. There are two other unsupervised linear methods that are similar to principal component analysis, which are factor analysis and multidimensional scaling.

3.4. Approaches for Dimensionality Reduction

Projections and manifold learning are two of the most often used approaches for reducing dimensionality, that is why it is important to be familiar with these methods before moving on to algorithms like locally linear embedding, multi-dimensional scaling (MDS), and isometric feature mapping that we will cover in the next chapter

3.4.1. Projection

Training examples are not regularly distributed across all dimensions in most real-world problems. Many features are nearly constant, while others are closely correlated. As a result, all training examples are located (or near to being contained) in a significantly lower-dimensional subspace of the higher-dimensional space. Let's look at an example to make this clearer. A 3D dataset is represented by the circles in Figure 3.5. This is a two-dimensional subspace of a three-dimensional (3D) space, as we can see from all the examples used in the training. To create the new 2D dataset shown in Figure 3.6, we simply project every instance of our training data in the plane perpendicularly onto this subspace.

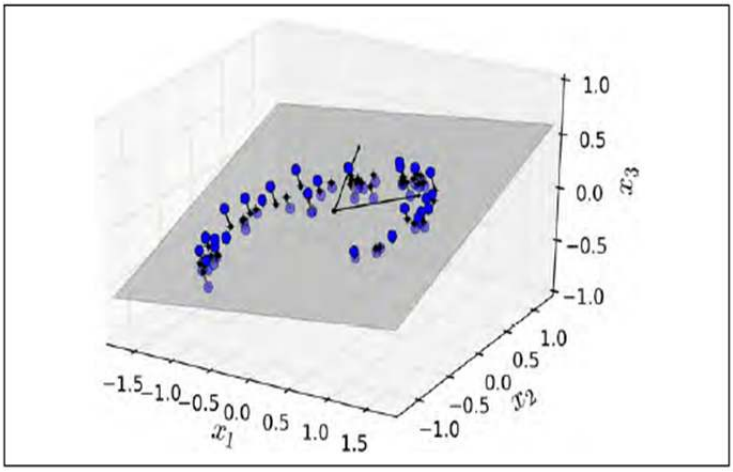

Figure 3.5. A (3D) dataset near (2D) subspace (Géron, 2017)

The dataset's dimensionality has been reduced from 3D to 2D as a result of this procedure. The new features are represented by the axes Z_1 and Z_2 - the coordinates of the plane projections. Projection, on the other hand, may not necessarily be the most effective approach for minimizing the number of dimensions. Figure 3.7 shows how the famous Swiss roll toy dataset causes the subspace to twist in unexpected directions. The left side of Figure 3.8. shows how the Swiss roll layers are compressed together by simply projecting onto a plane. When unrolling the Swiss roll, however, the 2D dataset shown on the right side of Figure 3.8. can be obtained (Géron, 2017). This means that if data are linear or nearly linear in lower-dimensional space, the projection can be an extremely valuable tool for decreasing the dimensionality of a dataset.

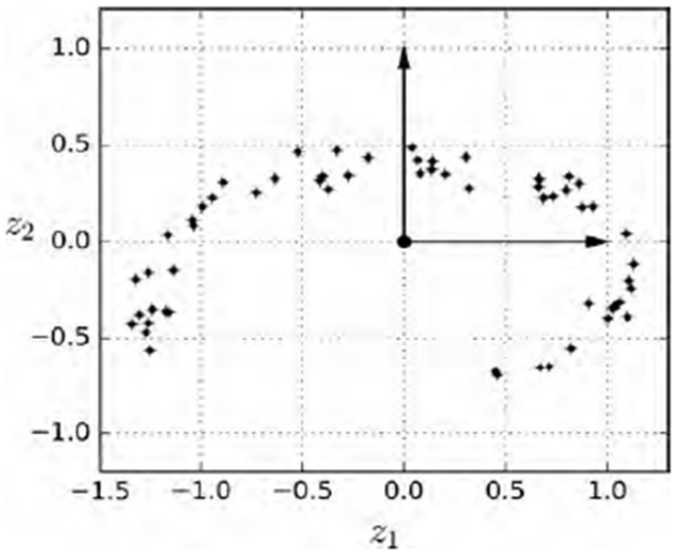

Figure 3.6. The new two-dimensional dataset after projection (Géron, 2017)

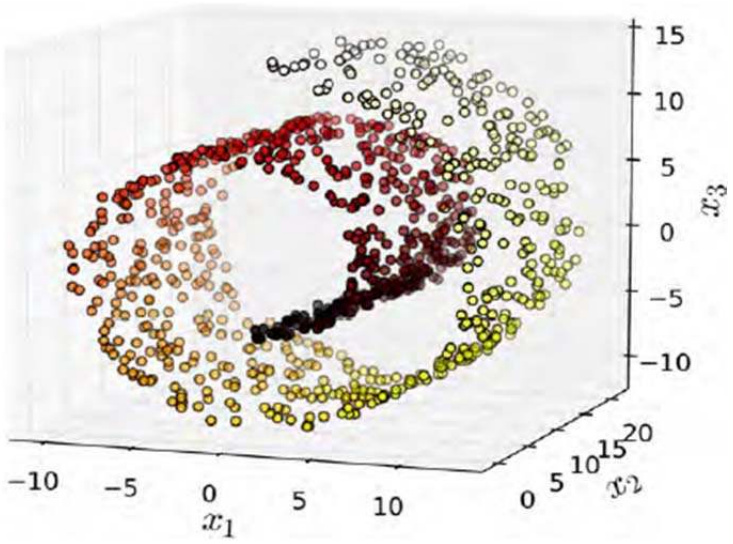

Figure 3.7. Swiss roll dataset (Géron, 2017)

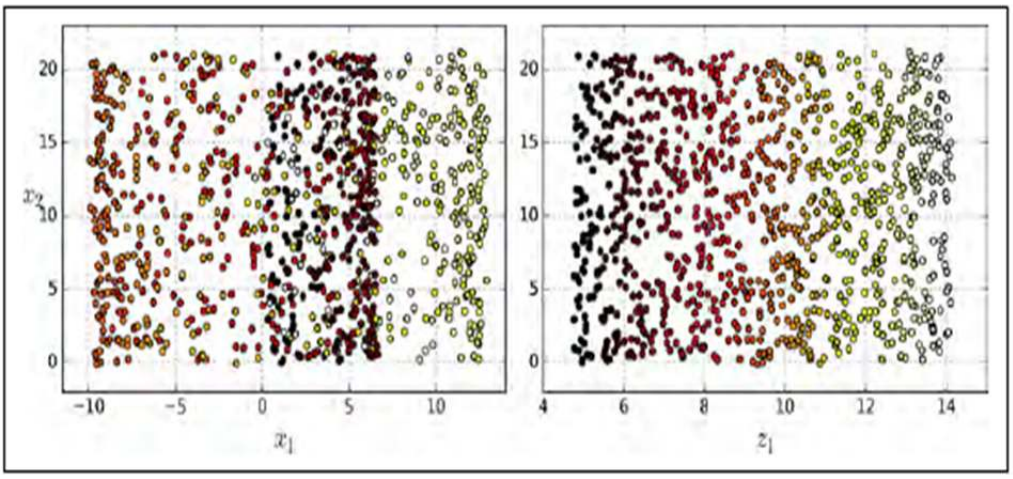

Figure 3.8. Projecting onto a plane vs. unrolling a Swiss roll (Géron, 2017)

3.4.2. Manifold Learning

Geometry, computer science, and statistical learning all come together in the manifold learning field of research. It's difficult to give a short definition of "manifold" because of the complex mathematical ideas that are included in it. In fact, even the brilliant French mathematician 'Elie Cartan' claimed that "The general concept of truth is extremely difficult to describe accurately ". According to the work of Georg Riemann in 1854, a manifold can be used to do differential and integral calculus on it. If a topological manifold is continuously differentiable to any order, it is referred to as M smooth (or differentiable) (i.e., $M \in C^{\infty}$). (Izenmen, 2008)

Manifolds extend the two- and three-dimensional concepts of curves and surfaces to higher dimensions. An ant at a picnic with everything from cups to doughnuts is a good example. Because of its small size, the ant views everything as flat and characterless from a very small viewpoint as it walks all over the picnic items. The curvature of the Earth would be invisible to a human looking only at his immediate surroundings. Same as a manifold, which is also known as a topological manifold, is a topological space with a flat and characterless local appearance and behaves like Euclidean space, which means that every point has a surrounding neighborhood that is topologically identical to the open unit ball in R^d. There is no idea of distance in a topological space. The easiest way to describe manifold learning is to say that is a set of algorithms for extracting low-dimensional manifolds from high-dimensional environments. (Ma & Fu, 2012)

It is necessary to make the assumption that most high-dimensional datasets in the real world occur near a considerably lower-dimensional manifold. This theory is backed up by practical applications. Manifold learning is typically employed to visualize data, interpret it, and classify it. Face recognition and character detection, form analysis, and target classification are all examples of

dimension reduction and manifold learning applications. (Brucher et al., 2007) . There are two main kinds of manifolds which are linear and nonlinear.

Linear Manifold Learning :

We can represent a linear manifold as a line, plane or hyperplane depending on the number of dimensions included. Many people believe that linear manifolds are optimal for describing relationships among variables in a high-dimensional space. This problem of linear manifold embedding in higher-dimensional space is closely connected to the classical statistics problem of linear dimensionality reduction. Reducing the amount of linear transformations of the input variables is the most commonly recommended approach. By looking to linear transformations as projection methods, then the problem is to create a series of low-dimensional data projections.

Nonlinear Manifold Learning:

The structure of nonlinear or curved manifolds cannot always be discovered using linear methods. There are nonlinear methods of manifold learning that attempt to retain the local or global structure of the manifold. Spectral embedding methods include LLE, Isomap, Laplacian eigenmaps, etc. There are three stages to these algorithms: in the first stage, a search for nearest neighbors in a high-dimensional input space is executed. A distance calculation based on the neighborhood graph formed in the stage prior to that is done in the second stage. In the third and last stage an eigenproblem for embedding the points in a low-dimensional space. (Ma & Fu, 2012)

4. ALGORITHMS FOR DIMENSIONALITY REDUCTION

Many algorithms are used for dimensionality reduction. In this section are going to discuss four of the most common algorithms.

4.1. Principle Component Analysis

Principal component analysis (PCA) is the most commonly used multivariate statistical method, with applications in every scientific field. It's an unsupervised linear projection technique for dimensionality reduction where the variables are combined in a certain manner, and then the less significant ones are dropped, while still preserving the most distinct parts of all the variables. Pattern recognition and computer vision applications like face-recognizing and picture compressing use PCA as feature extraction and data representation technique.

According to Preisendorfer et al. (1988), PCA is based on the singular value decomposition (SVD) proposed by both Beltrami (1873) and Jordan (1874). In reality, PCA can be traced back to Pearson (1901). Hotelling (1933), was the first one who used the phrase "principal component" and formalized it in its current form. When a data set has a number of dependent variables that are usually intercorrelated, PCA is used to extract the significant information from the data set and express it as a set of new orthogonal variables known as principal components. The principal components are independent if the data are normally distributed. It also shows the similarity between variables and observations by displaying them as points on a graph (Dikko et al., 2013).

Briefly, we can say that PCA is an approach of extracting only the most significant information from the dataset, compressing the data set by only retaining this information, simplifying data set explanation, and examination of observation and variable structure. (Abdi & Williams, 2010). PCA aims to create a linear d-dimensional subspace that includes the majority of the data points which are given

in n-dimensional space (where $d \prec n$). It is possible, but not guaranteed, that the data will be located exactly in a lower-dimensional subspace. Our goal is to find a subspace that captures the majority of the data's variation. As much variance in the source data as possible should be preserved in the new coordinate systems after applying PCA.

4.1.1.The Derivation of Principal Components

According to Hotelling, the most general definition of PCA states that for a given collection of data vectors X_i ; i=1,2,……, n, the orthonormal axes on which the projection retains the highest variance are known as the *P* principal axes. In order to obtain as much variation as possible, we'll look for the first principal component which we will call it U_1 , to capture the maximum variance. We may therefore assume that all centered observations are organized into rows and columns in a $d \times n$ matrix X , There are n observations, and each column represents a d-dimensional observation. $X = [X_1 \ldots\ldots\ldots X_n]_{d\times n}$, $X_i \in R^d$

The projection of n, d-dimensional observations on the first principal component $U_1$is $U_1^T X$. No, we want to find $U_1^T X$ such that its variance is as large as possible. For that, we are going to maximize the variance of $U_1^T X$. Now, we have an optimization problem $Max_{U1}\, var(\, U_1^T X)$
and since $var(\, U_1^T X) = U_1^T S U_1$, where S is a $d \times d$ sample covariance matrix of X. Then our problem become $Max_{U1} U_1^T S U_1$

In order to be able to solve this problem, we are going to put a constraint on the length of U_1 to be equal to 1, i.e, $U_1^T U_1 = 1$, and the optimization problem that we have to solve is:

$$Max_{U1} U_1^T S U_1 \qquad (4.1)$$

Subject to $U_1^T U_1 = 1$

we will use the lagrangian method to solve this problem:

$L(U_1, \lambda_1) = U_1^T S U_1 - \lambda_1 (U_1^T U_1 - 1)$

$\frac{\partial}{\partial U_1} L(U_1, \lambda_1) = 2SU_1 - 2\lambda_1 U_1 = 0$

$\Rightarrow 2SU_1 = 2\lambda_1 U_1$

$\Rightarrow SU_1 = \lambda_1 U_1$

Since U_1 and λ_1 are respectively an eigenvector and an eigenvalue of S.Then we can write $U_1^T S U_1 = U_1^T \lambda_1 U_1$

$= \lambda_1 U_1^T U_1$

$= \lambda_1$ since $U_1^T U_1 = 1$

S has at most d eigenvalues and d eigenvectors . Let us order the eigenvalues as follows

$$\lambda_1 \succ \lambda_2 \succ \lambda_3 \succ \cdots \ldots \ldots \succ \lambda_d$$

Then the eigenvectors corresponding to these eigenvalues are $U_1, U_2, U_3 \ldots \ldots \ldots, U_d$.

The first principal component is the eigenvector that corresponds to the biggest eigenvalue (because it maximizes the objective function). The eigenvector associated with the second-largest eigenvalue is the second principal component, and so on.

Let us now prove the principal components are orthogonal. Let U_1 be the eigenvector of S corresponding to λ_1 then we can write :

$SU_1 = \lambda_1 U_1$ (4.2)

And Let U_2 be the eigenvector of S corresponding to λ_2 where $\lambda_2 \neq \lambda_1$ then we can write

$SU_2 = \lambda_2 U_2$ (4.3)

And since S is a symmetric covariance matrix and from (4.2) and (4.3) we can write:

$U_1^T S U_2 = U_1^T S^T U_2 = (SU_1)^T U_2 = \lambda_1 U_1^T U_2$ (4.4)

$U_1^T S U_2 = U_1^T (\lambda_2 U_2) = \lambda_2 U_1^T U_2$ (4.5)

From (4.4) and (4.5) we see that : $\lambda_1 U_1^T U_2 = \lambda_2 U_1^T U_2$

$$\Rightarrow (\lambda_1 - \lambda_2) U_1^T U_2 = 0$$

But since $\lambda_1 \neq \lambda_2$ then $U_1^T U_2 = 0$ which means that U_1 and U_2 are orthogonal.

In order to approximate the space spanned by the original data points $X = [X_1 \ldots\ldots\ldots X_n]_{d \times n}$, we can choose p (where $p \leq d$) based on what percentage of the variance of the original data we would like to maintain.

4.1.2. Advantages and Limitations of PCA

The adavantages of PCA are :

- Improving visualization: PCA transforms data from their original high-dimensional space to a lower-dimensional subspace. making it more visual.
- Reduction of overfitting: With too many variables in the dataset, overfitting is probable. By lowering the number of variables, PCA assists in solving the overfitting problem.
- Removing the correlation between variables: All of the principal components will be independent of each other after executing PCA on the dataset. There will not be any correlation among them.
- Improvement of the performance of the algorithm by eliminating related variables that have no effect on decision-making.

Where the limitations of PCA are :

- It supposes that all relationships among variables are linear.
- PCA is sensitive to scaling therefore numerical data must be scaled/ standardized before running PCA, otherwise, PCA will be unable to discover the optimal principal components.
- Independent variables become more difficult to interpret.**:** After applying PCA to the dataset, original features will be transformed into principal components, that are linear combinations of original

features. Original features are easier to understand and read than principal components.

- It doesn't have the structure of a probabilistic model that is required for Bayesian selection and mixture modeling.
- Loss of information**:** Even though principal components attempt to cover as much variance as possible among the dataset's features. That is why the selection of the number of principal components without care may result in loss of data.

4.1.3. Some Suggested ways for the limitations of PCA

In 1941 Guttman suggested nonlinear principal components analysis as an alternative method to overcome the first second and third limitations. However, this method is more appropriate for variables that have a wide range of measurement points (nominal, ordinal, or numeric) than PCA, which has the same objective function. All variables are viewed as categorical in nonlinear PCA, and each variable's unique value refers to a category. (Linting et al., 2007)

Nominal variables can't be studied with PCA because of its limitations. Categorical variables and ordinal variables are both terms used to describe the same type of variable. A Likert scale rating is an example of a variable that is composed of a set of ordered categories. The most significant difference is that in linear PCA, the measured variables are instantly examined, however, the values of the measured variables are maintained throughout the analysis in the nonlinear PCA (Khalid et al., 2014).

Schölkopf et al.(1997) suggested a method known as Kernel principal component analysis (KPCA), which gets over the first limitation by making use of a kernel technique. The kernel technique has been used for the reformulation of linear approaches in recent years, which has led to the creation of effective techniques such as kernel ridge regression and support vector machines. The

KPCA technique was developed with the primary goal of avoiding the direct assessment of the needed dot product in high-dimensional feature space by utilizing the kernel function. As a consequence of this, an explicit nonlinear function to project the data from the original space to the feature space is not required. Instead, the process may be done implicitly. Another kind of PCA is something called probabilistic principal component analysis(PPCA), and it was initially proposed by Tipping and Bishop (1998). The fourth limitation is eliminated by PPCA, which allows the noise component to have an isotropic structure. In this model, the PCA is implicitly incorporated in a parameter learning step by applying the maximum likelihood estimation approach. In addition to this, an effective expectation/maximization approach for iteratively learning the parameter has been developed. To get over PCA's limitations, Tipping and Bishop presented probabilistic kernel principle component analysis in the same year. This method analyzes kernel principal components in a probabilistic manner while combining PPCA and KPCA (Zhou, 2003).

Incremental principal components analysis (IPCA) was recently proposed by Artac et al. (2002). IPCA is often used as an alternative to PCA for subdividing the set of data when PCA cannot be performed because the dataset is too huge to hold in memory. For improved memory control, IPCA splits the training set into mini-batches and runs the algorithm on each mini-batch separately.

4.2. Multidimensional Scaling

When it comes to computer science, data analysis, or multivariate analysis, multidimensional scaling (MDS) is one of the most commonly used approaches. The Thomson Reuters Web of Science returned 5,186 papers cited in 68,429 other papers when searching for "multidimensional scaling". (per January 2013). This shows that MDS is a widely used multivariate analysis tool (Groenen & Borg, 2013).

Some people use MDS to refer to a pretty specific collection of procedures while others use it in a more broad context, making it difficult to define it clearly. For this reason, distinguishing between MDS in its narrowest sense broadest sense, and MDS in its broadest sense seems to be advantageous. If we take MDS in its narrowest sense, it depicts dissimilarity data in a low-dimensional space, but in its broadest sense, it covers numerous cluster analysis and linear multivariate analysis methods (de Leeuw & Heiser, 1982). When comparing sets of things, multidimensional scaling provides a visual depiction of how far apart they are. Colors, faces, coordinates on a map, political ideology, and other physical or mental inputs can all be considered objects (Kruskal & Wish, 1978). Another comprehensive definition of MDS was given by Groenen ve Borg (2005), who stated that Multidimensional scaling (MDS) is a technique that makes use of distances between points in a low-dimensional multidimensional space to express measures of similarity (or dissimilarity) between pairs of objects. For example, data may be correlations between intelligence tests, then the MDS representation is a plane that represents the tests as points that are close to each other when the tests have a positive correlation and vice versa. Visual representation of correlations given by MDS allows the data researcher to look well at data and examine their structure. MDS is a method that uses distances between objects in a low-dimensional multidimensional space to describe measurements of similarity (or dissimilarity) between pairs of items. As a result, instead of relying on the typical

instances by variables, MDS relies on closeness measurements between pairs of objects. It is possible to assess correlations between test item correlations, the similarity or dissimilarity of politicians, or the dissimilarity of mobile phones. Low-dimensional (usually 2D) space is used to represent those objects so that researchers can see how similar or different they are from each and this also makes the data structure becomes much simpler for them than a data matrix with several digits (Groenen & Borg, 2013). According to Mugavin (2008), MDS is an exploratory data analysis technique, that condenses huge amounts of data into a relatively simple map that effectively reflects essential correlations. Data can be nominal or ordinal and no multivariate normality is required to build a model that depicts nonlinear relationships among variables in MDS. As a result, MDS can be used instead of approaches like factor analysis and smallest space analysis in data exploration to obtain representative information. (Johnston, 1995; Steyvers et al., 2002). As a result, Visual representation of dissimilarities or similarities between objects, situations, or observations, in general, is provided by MDS. By transforming a series of dissimilarity measures into distances that are given to specific locations in a spatial configuration, the technique aims to identify patterns in data.

According to de Leeuw & Heiser (1982), surveyors and geographers have used MDS-related methodologies since Kruskal established a primitive MDS approach in systematic zoology used by Boyden around 1930. In a research by Fisher (1922), algorithms for mapping genomes based on crossing-over frequencies were provided. However, the systematic development of MDS has basically taken place in psychometrics. The Thurstonian school is responsible for many of the method's developments. Classical psychophysical data gathering methods were employed to pairs of stimuli by Richardson (1938) and Klingberg (1941). Table 4.1. provides a subjective overview of MDS's development from earlier stages to more recent ones.

Table 4.1. Subjective overview of milestones in MDS (Groenen & Borg, 2013)

Years	Main author(s)	Topic
Past		
1958, 1966	Torgerson,Gower	Classical MDS
1962	Shepard	First MDS heuristic
1964	Kruskal	Least-squares MDS through Stress with transformations
1964	Guttman	Facet theory and regional interpretations in MDS
1969, 1970	Horan, Carroll	Three-way MDS models (INDSCAL, IDIOSCAL)
1977-	De Leeuw and others	The majorization algorithm for MDS
Present		
1986-1998	Meulman	Distance-based MVA through MDS
1994	Buja	Constant dissimilarities
1978, 1995-	Various	Local minimum problem
1998	Buja	Smart use of weights in MDS
Future		
1999-,	Heiser, Meulman, Busing	Modern MDS software: Proxscal in SPSS (PASW)
2000	Tenenbaum, et al.	Large scale MDS ISOMAP heuristic
2002	Buja, Swayne, Cook	Dynamic MDS in GGvis (part of GGobi)
2003	Groenen	Dynamic MDS visualization through iMDS
2005-	Groenen, Trosset, Kagie	Large scale MDS through Stress
2002	Denœux, Masson, Groenen, Winsberg, Diday	Symbolic MDS of interval dissimilarities
2006	Groenen, Winsberg	Symbolic MDS of histograms
2009	De Leeuw, Mair	SMACOF package in R

4.2.1. MDS Algorithm

Classical MDS works under the assumption that the coordinates matrix X may be obtained from the scalar product matrix $B = XX^T$ by using eigenvalue decomposition. We can derive B from the proximity matrix $D = [d_{ij}]$, i,j=1,2.....n, by multiplying it with the dual centering matrix J_n . The following is a concise description of the algorithm:

1. We begin by creating a squared proximity matrix $D^2 = [d_{ij}^2]$, where d_{ij} is the Euclidian distance between i-th and j-th objects, where i,j=1,2.....n, and n is the number of objects.
2. We perform a double-centering operation $B = -\frac{1}{2}CD^2C$ by the use of the centering matrix $C = I - \frac{1}{n}J_n$, where n is the number of objects, $I$is an $n \times n$ identity matrix and J_n is a matrix of all ones.
3. We find the m greatest eigenvalues $\lambda_1, \lambda_2, \dots, \lambda_m$ and corresponding eigenvectors $e_1, e_2, \dots, e_m$ of B where m is the number of dimensions, we want the output to be.
4. Then, $X = E_m \Lambda_m{}^{\frac{1}{2}}$ where E_m is a matrix of m eigenvectors and $\Lambda_m^{\frac{1}{2}}$ is a diagonal matrix of m eigenvalues of B.

The Euclidean distance is used in classical MDS, but in a broader sense, it can be any metric or arbitrary distance function.

4.2.2. Types of MDS

According to Cox and Cox (2001), MDS can be divided into two types based on the manner in which dissimilarities δ_{rs} are transformed into distances d_{rs}. These two types are metric and nonmetric MDS.

a) Metric dimensional scaling

In 1938, Richardson's book "Multidimensional Psychophysics" was the publication where the term "metric multidimensional scaling" was used for the first time. Before Richardson, several people, including Boyden (1933), had used the concept, but they did not call it multidimensional scaling at that time. Boyden, who is a biologist, had used the method to develop models for the interactions between many common amphibia. Torgerson (1958) rediscovered and expanded upon the metric MDS methodology three decades after the work of Richardson. The algorithm for metric MDS is the same as the one given above.

b) Nonmetric Dimensional scaling

It was Hays in 1964 who first referred to nonmetric MDS (also known as ordinal MDS), which uses rankings instead of distances (Coombs, 1964). Qualitative parallels are used in ordinal MDS. Distances are measured using an ordinal MDS scale. Nonmetric MDS totally depends on ranks as a source of data, and the distances in X are ordered as closely as possible according to their distances from the nearest points (Borg et al., 2013). In a nonmetric MDS, the rank value is the sole criterion for describing a set of curves regardless of how complicated or how basic the curve is, it will always rise (or decrease) from left to right. Unlike the classical method of MDS which depends mainly on metric distances, non-metric MDS uses both a non-parametric monotonic relationship and the low-dimensional location of each item in the low-dimensional space. There are several ways to find the relationship, but isotonic regression is a common one. let x denote the vector of proximities, $f(x)$ a monotonic transformation of x, and d the point distances; then coordinates have to be determined so that it minimizes so-called stress.

$$\text{Stress} = \sqrt{\frac{\sum(f(x) - d)^2}{\sum d^2}}$$

There are a variety of cost functions that can be used. MDS programs automatically reduce stress in order to obtain the MDS solution.

There are a variety of cost functions that can be used. MDS programs automatically reduce stress in order to obtain the MDS solution. There are two stages to the Nonmetric MDS algorithm. The first step is to determine the optimal monotonic transformation of the proximities. As a second consideration, it is essential that the configuration's points be arranged so that their distances approximate the scaled proximities as closely as possible. The following are the basic steps in a nonmetric MDS algorithm :

1. It's first necessary to obtain some sort of random configuration of points, such as by sampling from a normal distribution.
2. Then the distance between the points is calculated.
3. In order to get the most accurate data possible, it is necessary to scale the data appropriately. We should select the best monotonic transformation of the distances between the points $f(x)$
4. In order to lessen the stress on the ideally scaled data and the distances, we find a new configuration of points.

   We compare our current level of stress to a predetermined criterion. The algorithm can be terminated if the stress is is low enough; otherwise, step 2 should be repeated.

4.3. Isomap

It is possible to reduce the dimensionality of nonlinear data using the unsupervised machine learning approach known as isometric mapping (Isomap). It

was introduced in 2000 on a paper by Tenenbaum, Silva, and Langford. According to these researchers, dimensionality reduction methods like PCA and MDS are simple to use, have quick computation times, and ensure that a data's underlying structure is located within or near a linear subspace of the high-dimensional input space. PCA finds a low-dimensional embedding for the data points that best preserves their variance in the high-dimensional input space. MDS is equivalent to PCA when the distances between points are Euclidean, which is the case with classic MDS. Nonetheless, PCA and MDS are unable to discover major nonlinear patterns in many datasets. For instance, neither technique can detect the face data set's inherent three-dimensionality or genuine degrees of freedom.

The extension of classical MDS, which uses a non-linear approach to data analysis is referred to as Isomap. Isomap performs MDS calculations not in the input space, but in the geodesic space of the nonlinear data manifold. When the curved surface of the manifold is measured as if it were flat, the geodesic distances are the pathways that take the least amount of time to go along. An Isomap is produced by creating a connection between each instance and its closest neighbors. Then it attempts to reduce dimensionality while trying to preserve the geodesic distances between the points. Isomap calculates the distances by making use of the geodesic distances that exist between each pair of data points. It is possible to employ the Euclidean distance between neighboring input space points due to the fact that this measure is linear for very minor alterations in pose on a manifold. An estimate of the geodesic distance can be obtained for more distant points by adding the distances that exist between points on the manifold.

The basic idea of Isomap is to apply MDS in the geodesic space of the nonlinear data manifold, instead of doing that in the input space. When the manifold's curved surface is measured as if it were flat, then the geodesic distances represent the shortest pathways among points. Isomap generates a graph by connecting each instance to its closest neighbors; then reducing dimensionality

while attempting to preserve the geodesic distances between the instances. Isomap uses the geodesic distances between all pairs of data points. Euclidean distance can be used for neighboring points in the input space that are close, where the manifold is locally linear for slight changes in pose. The sum of the distances between the points along the way over the manifold is an approximation for geodesic distance for far-away points After that Isomap tries to detect a low-dimensional mapping based on MDS, that maintains pairwise distances rather than straight-line distances.

4.3.1. Isomap Algorithm

There are three steps in the Isomap algorithm that are:

1. Construct a k-nearest neighbor graph for n data points
2. Compute the shortest paths that connect all of the points As an approximation of the geodesic distance $D^{(G)}$.
3. Compute $K = -\frac{1}{2}HD^{(G)}H$, where $H = I - \frac{1}{2}ee^T$ is a centering matrix, $e = \begin{pmatrix} 1 \\ 1 \\ . \\ . \\ 1 \end{pmatrix}_{n\times 1}$
4. Find eigenvectors of K and call it matrix V
5. Find the top p eigenvalues of K and form the matrix Λ . Then the solution is $Y = \Lambda^{\frac{1}{2}}V^T$

4.3.2. The Performance and Limitations of Isomap

The İsomap algorithm has proven an efficient performance for unfolding the nonlinear convex manifolds. The most common use of it is the swiss roll data which is data that has been generated for testing out the different dimensionality

reduction algorithms. The Isomap looks to work most efficiently when the number of data points $n \leq 1000$ (Ma & Fu, 2012). When that data set is larger then Isomap work slowly. A modified version of Isomap, called Landmark Isomap, was used to improve the algorithm's performance.

Assigning "landmark" points to only a small portion composed of m points of the total n data points minimizes redundancy and improves readability. For example, if x_i is one of the m landmark points then s a starting point, we can simply calculate the distances between x_i and the other n points. Landmark Isomap algorithm requires a $m \times n$ matrix of distances as input. The landmark points could be chosen by random sampling or by selecting "representative" points. The researcher has the option of deciding on the number of landmarks, however, $m = 50$ seems to work well. In addition to that Landmark Isomap uses Dijkstra's algorithm instead of Floyd's algorithm, which is slower in sparse graphs. By using Dijkstra's algorithm, landmark Isomap calculates graph distances in a more faster and efficient manner. (De Silva & Tenenbaum, 2003)

4.4. Locally Linear Embedding

The strong graphical nonlinear dimensionality reduction algorithm known as locally linear embedding (LLE) was first proposed in the year 2000 by both Sam Roweis and Saul. LLE is useful for both the embedding of manifolds and the extraction of features (Ghojogh et al., 2019). According to Alpaydın (2014) in LLE the manifold is looked at as a group of small paths and linear approximation may be applied to every small patch of the manifold. Each point may be described as a linear weighted sum of its neighbors when enough data is given.

Roweis and Sam stated that previously used MDS approaches found embeddings that tried to maintain pairwise distances or generalized dissimilarities between data points. These distances are measured along straight lines or, in more advanced MDS applications like Isomap, along shortest paths

restricted to the set of observed inputs. According to Roweis and Sam, when data points are far apart, LLE removes the necessity to calculate the distance between them. On the other hand, and in a different way from other approaches LLE, seeks to recapture the global nonlinear structure from locally linear fits.

4.4.1. The Algorithm of LLE

Let us suppose that we have *n* real-valued *d*-dimensional x_i , *i =1,2.....n*, sampled from an underlying nonlinear manifold. LLE algorithm has the following steps :

1. We assume every data point x_i and its neighbors to be situated on a locally linear patch of the manifold or close to it. Finding the nearest *k*-neighbors for each point is the first step in the process. To do this we choose *k*-nearest neighbors or all points within an extremely small constant radius ε
2. We calculate all of the weights w_{ij} that reconstruct x_i from its neighbors linearly in the best way. Then the cost function used to calculate the reconstruction errors can be given as follows

$$\varepsilon(w) = \min_{y} \sum_{i=1}^{n} \left| x_i - \sum_{j=1}^{k} w_{ij} x_{N_i(j)} \right|^2 \qquad (4.6)$$

$$\text{s.t} \quad \sum_{j=1}^{k} w_{ij} = 1$$

$N_i(j)$: is the index of the j-th neighbor of x_i

w_{ij} : Unkown weights

Let us first define $N_i = \left[x_{N_i(1)} \ldots\ldots\ldots\ldots x_{N_i(k)}\right]_{d\times k}$ the matrix of all neighbors of a point x_i

$w_i = \begin{bmatrix} w_{i1} \\ w_{i2} \\ . \\ . \\ w_{ik} \end{bmatrix}_{k\times 1}$ the vector of the weights of the i-th point, $e = \begin{bmatrix} 1 \\ 1 \\ . \\ . \\ 1 \end{bmatrix}_{k\times 1}$

Then we can write:

$$\sum_{j=1}^{k} w_{ij} x_{N_i(j)} = N_i w_i$$

And if we construct a matrix of the form $[x_1 \ldots\ldots\ldots x_n]_{d\times k} = x_i e^T$ then we can write $x_i = x_i e^T w_i$, and then (4.6) becomes in the following form :

$$\min_y |x_i e^T w_i - N_i w_i|^2 = \min_y |(x_i e^T{}_i - N_i) w_i|^2$$

$$= \min_y w_i{}^T (x_i e^T{}_i - N_i)^T (x_i e^T{}_i - N_i) w_i$$

Let us call $(x_i e^T{}_i - N_i)^T (x_i e^T{}_i - N_i) = G$, G is called a gram matrix. Then the problem will become in the form :

$$\min_y w_i{}^T G w_i$$

$$\text{s.t.} \quad e^T w_i = \sum_{j=1}^{k} w_{ij} = 1$$

which is an optimization problem. To solve it we calculate the lagrangian :

$$L(w_i, \lambda) = w_i{}^T G w_i - \lambda(e^T w_i - 1)$$

$$\frac{\partial L(w_i, \lambda)}{\partial w_i} = 2G w_i - \lambda e = 0$$

$$2G w_i = \lambda e \Rightarrow G w_i = \frac{\lambda}{2} e$$

And we see that from the last form we can find w_i for an arbitrary value of λ

3. The last step will be fixing weights found in the last step and finding low-dimensional embedding (representation) of points y_i in which every point

may be reconstructed, using the same set of weights, with its k-nearest neighbors.

$$\Phi(y) = \min_{y} \sum_{i=1}^{n} \left| y_i - \sum_{j=1}^{n} w_{ij} y_j \right|^2 \tag{4.7}$$

$$\text{s.t.} \quad \frac{1}{n} Y^T Y = I \ \ and \quad \sum_{i=1}^{n} y_i = 0$$

First, we will define a $(n \times n)$ matrix W such that for every point we write the weights of the *k*-nearest neighbors and 0 otherwise, and we define $w_{:i}$ which is a column of this matrix as follows:

$w_{:i} = \begin{bmatrix} 0 \\ . \\ 0 \\ w_1 \\ w_2 \\ . \\ . \\ w_k \\ 0 \\ . \\ 0 \end{bmatrix}_{n \times 1}$, a column of W such that from w_1 to w_k are *k*-nearest neighbors of x_i and zero otherwise,

Let us define the following matrices :

$Y = [y_1 \quad y_2 \quad . \quad . \quad y_n]_{p \times n}$

$I = \begin{bmatrix} 1 & 0 & 0 & . & . & . & 0 \\ 0 & 1 & 0 & . & . & . & 0 \\ 0 & 0 & 1 & . & . & . & 0 \\ . & . & . & 1 & . & . & . \\ . & . & . & . & . & . & . \\ . & . & . & . & . & 1 & . \\ 0 & 0 & 0 & . & . & . & 1 \end{bmatrix}_{n\times n}$, a $n \times n$ identity matrix and $I_{:i} = \begin{bmatrix} 0 \\ . \\ . \\ . \\ 1 \\ 0 \\ . \\ . \\ 0 \end{bmatrix}_{n\times 1}$ which is a vector of the identity matrix that has the value 1 in the *i*-th position and 0 otherwise.

Then we can write $y_i = YI_{:i}$ and $\sum_{j=1}^{n} w_{ij}y_j = Yw_{:i}$ and according to this *the* objective function (4.7) can be written in the form:

$$\begin{aligned} \min_y \sum_{i=1}^{n} |YI_{:i} - Yw_{:i}|^2 &= \min_y |YI - YW|^2 \\ &= \min_y |Y(I - W)|^2 \\ &= \min_y Tr[(I - W)^T Y^T Y (I - W)] \\ &= \min_y [Y(I - W)^T (I - W) Y^T] \end{aligned}$$

Let$(I - W)^T (I - W) = M$ then The problem will be in the form :

$$\min_Y \ Tr[YMY^T]$$

$$s.t. \quad \frac{1}{n} Y^T Y = I \ \ and \ \sum_{i=1}^{n} Y_i = 0$$

which is an optimization problem. the solution of it is the first bottom p eigenvectors of M (where p ≤d). But M will always have one eigenvalue that is equal to zero, the eigenvector which corresponds to this eigenvalue is a constant

vector of ones. Therefore we will pick the (p+1) bottom eigenvectors ignoring the one corresponding to an eigenvalue equal to zero. The origin and direction of the solution for Y can have an arbitrary origin and orientation. That is why the problem cannot be well-posed until both of these variables are eliminated,i.e, two degrees of freedom should be removed. These two degrees of freedom are removed by constraining the coordinates to be centered on the origin $\left(\sum_{i=1}^{n} Y_i = 0\right)$ and putting a constraint on the embedding vectors such that $\left(\frac{1}{n} Y^T Y = I\right)$ which means that they have unit covariance (A.Ghodsi,2006).

4.4.2. Strength Points of LLE

According to Roweis and Saul (2000), LLE has some characteristics that make it more practical to use. These points can be summarized as follows:

1. Many common LLE has a lot of advantageous characteristics which other nonlinear dimensionality reduction learning techniques lack. Other algorithms don't have the same guarantees of global optimality or convergence, and they usually have a larger number of free parameters.
2. The optimizations of LLE are very easy to deal with, unlike other nonlinear methods which rely on deterministic annealing strategies to prevent local minima in their search for optimal solutions.
3. LLE makes a good scaling with the intrinsic manifold dimensionality. It doesn't require restarting LLE to compute higher dimensional embeddings because existing dimensions don't change as more dimensions are added to embedding space.

4. Instead of using global restrictions, pairwise distances, and stress functions like Isomap, LLE analyzes local symmetries, linear coefficients, and reconstruction mistakes. Consequently, it reduces the need to deal with complex problems of dynamic programming, making it both faster and more convenient to use.
5. LLE can save a lot of time and space, and performs well on sparse data and nonconvex manifolds with holes, due to its tendency to accumulate very sparse matrixes.

5. APPLICATIONS ON REAL AND ARTIFICIAL DATA

In this chapter we are going to apply the algorithms that are covered in the fourth chapter of the study on 4 datasets, two of them are real and the other two are artificial datasets. The application is done by using the XLSTAT program and Python programming language.

5.1. Breast Cancer Data Application

First, we applied the principal components method using python programming language on the breast cancer dataset which is openly accessible in the University of California at Irvine (UCI) Machine Learning Repository. The Wisconson Diagnostic Breast Cancer (WDBC) data set contains a total of 569 cases, divided down into two main classes. In the first class, there are 357 patients who do not have cancer, and in the second class, there are 212 patients who do have cancer (Dua, 2017).

In the WDBC data set, there are 10 real-valued features calculated for every cell nucleus in addition to ID diagnostic attributes. There are four important digits in all feature readings. The WDBC data set does not contain any missing attribute values. A digital image of a fine needle aspirate (FNA) of a breast lump was used to calculate these attributes. A summary of these attributes and their categories is given in Table 5.1.

Except for the diagnostic characteristic, the WDBC data set is entirely numerical. There are two possible values for the diagnosis characteristic, which are B and M. The letters B and M denote "benign" and "malignant," respectively. The radius is calculated by taking the average of the distances between the center and the points on the perimeter.

The texture is defined as the standard deviation of gray-scale values. The local change in radius lengths is referred to as Smoothness. Compactness is calculated as $\frac{perimeter^2}{Area} - 1$

Table 5.1. Wisconsin diagnostic breast cancer data set description

No.	Attributes	Attribute Type
1	ID	Numeric
2	Diagnosis (B=benign, M=malignant)	Discrete (Binary outcome)
3	Radius	Numeric
4	Texture	Numeric
5	Perimeter	Numeric
6	Area	Numeric
7	Smoothness	Numeric
8	Compactness	Numeric
9	Concavity	Numeric
10	Concave Points	Numeric
11	Symmetry	Numeric
12	Fractal Dimension	Numeric

The severity of the contour's concave parts is referred to as concavity. The number of concave points on the contour represents the number of concave parts. Fractal dimension is defined as“coastline approximation” -1. A breast mass image's cell nucleus can be identified by looking at the information provided by these features. Thirty features were generated based on the mean, standard error (SE), and worst or highest (mean of the three largest values) values of these attributes. The numerical data is therefore divided into three main groups. The first group includes the mean values of ten basic features, as shown in Figure 5.1; the second group includes the calculations for the standard error(SE); and the third group includes the worst values of these ten parameters. One of the columns has the label "Unnamed: 32." During the analysis, we are going to disregard this feature because

it contains values of the form NaN. First, we have to standardize the data by subtracting the mean and dividing by the standard deviation for each value for each feature. Then we find the covariance matrix which is given in Appendix 1.

After that, we apply the singular value decomposition (SVD) to find the eigenvectors and eigenvalues. The eigenvalues arranged in descending order are given in Table 5.2

Table 5.2. The eigenvalues in a desending order

13.3049907943745	0.0800034044773
5.70137460372613	0.0595036135304
2.82291015500623	0.0527114222101
1.98412751773020	0.0495647002129
1.65163324233011	0.0312142605530
1.20948223980297	0.0300256630904
0.67640888170090	0.0274877113389
0.47745625468950	0.0243836913545
0.41762878210781	0.0180867939843
0.35131087488173	0.0155085271344
0.29443315349116	0.0081920371176
0.26162116136612	0.0069126125791
0.24178242132831	0.0015921360011
0.15728614921759	0.0007501214127
0.09430069560105	0.0001332790566

The accumulative ratio of variance for principal components is given below:

([44.27202561, 63.24320765, 72.63637091, 79.23850582,
84.73427432, 88.75879636, 91.00953007, 92.59825387,
93.98790324, 95.15688143, 96.13660042, 97.00713832,
97.81166331, 98.33502905, 98.64881227, 98.91502161,
99.1130184 , 99.28841435, 99.45333965, 99.55720433,
99.65711397, 99.74857865, 99.82971477, 99.88989813,
99.94150237, 99.96876117, 99.99176271, 99.99706051,
99.99955652, 100.])

We see that % 44.27 of the variation is described by the first PC, % 63.24 of the variation is described by the first two principal components, and so forth. The number of the principal components is determined according to the wanted ratio of explanation we want the model to give. If we take the number of principal components to be 2 then the PCA graph is as shown in Figure 5.1.

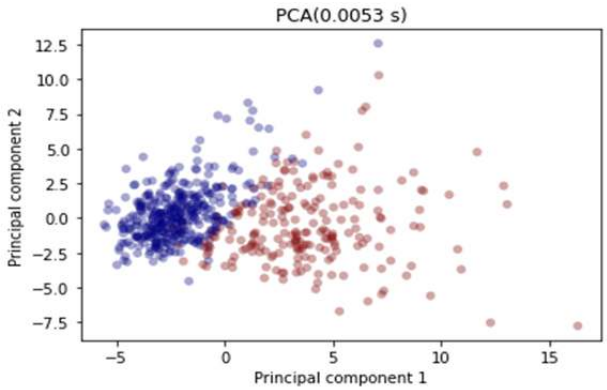

Figure 5.1. PCA graph for breast cancer data set

Let us now try to apply the MDS, Isomap, and LLE methods by taking the number of nearest neighbors to be 5 for Isomap and LLE. The graph of each method is given in Figure 5.2 with the execution time above each graph.

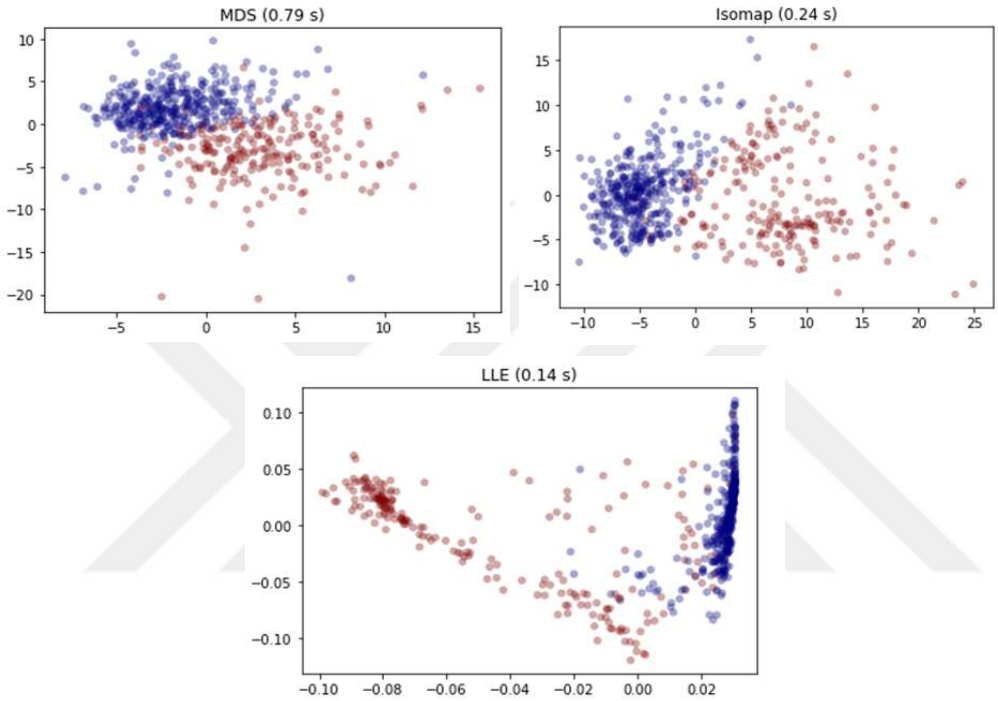

Figure 5.2. Sphere data set graphs from top left to right: a) MDS graph b) Isomap graph c) LLE graph

By comparing these methods we see all the methods produced almost the same graph but LLE produced different results. We noticed that the LLE gave different results when we changed the number of neighbors. We also noticed that the shortest execution time was achieved by using PCA.

5.2. Driving Distances Between Turkish Cities Application

Here we applied the MDS on the approximate pairwise driving distance between 15 Turkish cities which are available on the link (https://www.drivebestway.com/mileage-chart-with-distances-between-cities/tr/). The given matrix is not symmetric but we will make it symmetric by supposing that the measured distances between city *A* and city *B* are equal to the measured distance between city *B* and city *A* as given in Appendix 2

We used XLSTAT to find the MDS representation which is given in Figure 5.3.

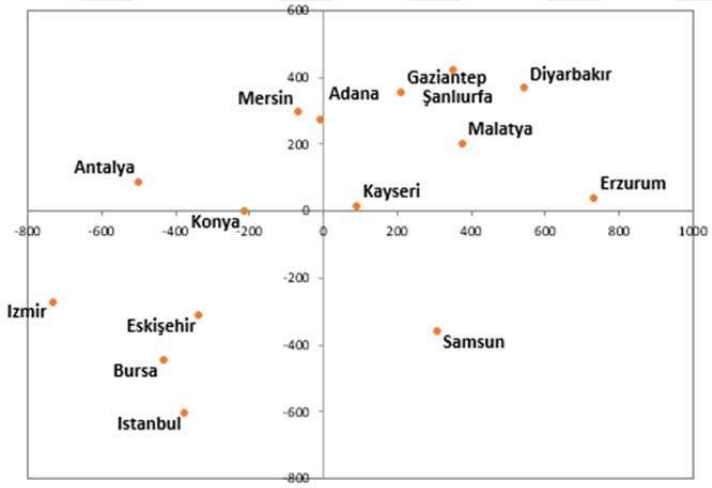

Figure 5.3. MDS representation for Turkish cities' data

The distances measured between cities in the representation space are given in Appendix 3. and the coordinates of cities in the new representation space are in Table 5.3.

Table 5.3. Coordinates of cities in the 2-dimensional space

	Dim1	Dim2
Istanbul	-371.629	-607.997
Izmir	-728.980	-275.585
Bursa	-429.306	-449.148
Adana	-3.316	268.792
Gaziantep	214.494	350.141
Konya	-208.838	-2.879
Antalya	-497.346	80.611
Diyarbakır	546.613	362.894
Kayseri	95.244	10.568
Mersin	-65.192	291.156
Eskişehir	-332.669	-317.436
Şanlıurfa	353.493	419.311
Malatya	379.046	197.855
Erzurum	736.568	35.635
Samsun	311.818	-363.918

We found that the stress which is given in the formula :

$$\text{Kruskal's stress(1)} = \sqrt{\frac{\sum (d_{ij} - \delta_{ij})^2}{\sum d_{ij}^2}} \quad , \; i, j = 1, 2, \ldots\ldots\ldots, 15$$

$$= 0{,}033$$

In general, the stress function gives a value between 0 and 1. The closer the stress to 0 the better representation we have. Since our stress is equal to 0.033 which is quite small, we can say that we have a very good representation.

On the other hand, we note that MDS doesn't consider the directions while representing the data points in a lower dimensional space. In Figure 5.4 we see that the east and west are on the right and left respectively but the north and south are reflected. For making the north upward, the south downward, the east on the right, and the west on the left we will rotate(reflect) the figure 180 degrees around the *X*-axis. Then the figure shows the map of these cities with the true directions as shown in Figure 5.4.

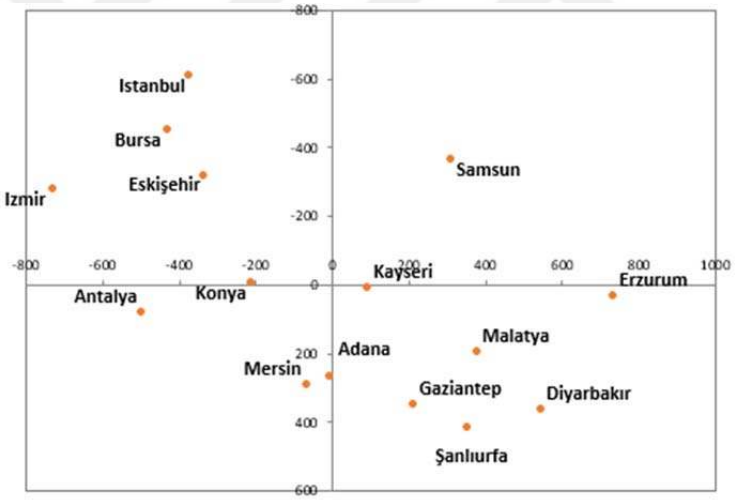

Figure 5.4. MDS representation for Turkish cities data after the rotation around of the *X*-axis

5.3. Sphere Data Applicatıon

We applied PCA, Isomap, and LLE on the sphere, which is a three-dimensional artificial data that is made of 1500 points, using the python programming language. The sphere data is given in Figure 5.5.

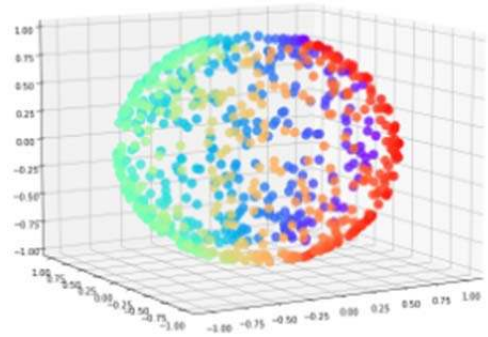

Figure 5.5. Sphere data set

When applying PCA, MDS, Isomap, and LLE (taking the number of neighbors to be 10) We got the results shown in Figure 5.6.

By comparing the graphs of the algorithms we found that the best unfolding for the sphere was given by both Isomap and LLE. The execution time of Isomap was more than twice as long as the LLE algorithm.

On the other hand, we see that both PCA and MDS gave poor results and take a lot of time compared to Isomap and LLE and that is due to the fact that the dataset is lying on a nonlinear manifold. PCA uses the orthogonal projection which leads to compressing the data as shown in the graph. While MDS uses the Euclidean distance, unlike Isomap which uses the geodesic distance, to preserve the distances between points of the sphere and this was the reason behind this poor performance when compared with Isomap and LLE algorithms.

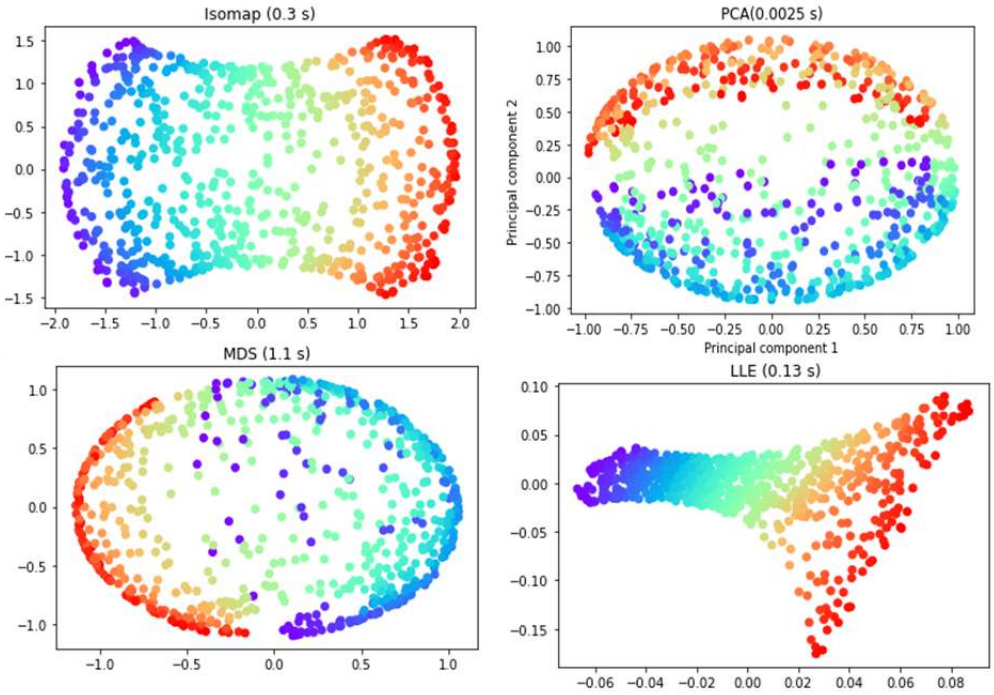

Figure 5.6. Sphere data set graphs from top left to right: a) Isomap graph b) LLE graph c) MDS graph d) PCA graph

5.4. S-curve Data Application

The S-curve dataset is three-dimensional artificial data that lies on a manifold in the shape of an S letter. We used 1000 points to generate the S-curve data set that is given in Figure 5.7

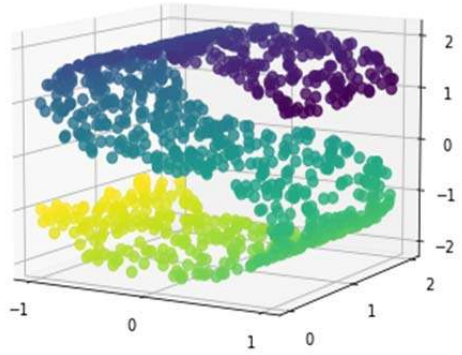

Figure 5.7. S-curve data set

We also used the python programming language for applying dimensionality reduction algorithms by taking the number of nearest neighbors to be 10. The results are given in Figure 5.8.

The same as in the sphere data we see that Isomap and LLE unfolded the S-curve manifold in different ways.Although both of them are graphical methods they work differently for the same purpose which is preserving distances between points in the lower dimensional space. While Isomap tries to find the k-nearest neighbors using the geodesic distance for preserving distances LLE look at this matter from a different perspective by looking at the manifold locally and dividing them into small patches. Then it tries to find the weights of the k-nearest neighbors for points in these patches to use them for representing the points in the lower dimensional space. When looking at the execution time we noticed that LLE is 3.5 times faster than the LLE algorithm.

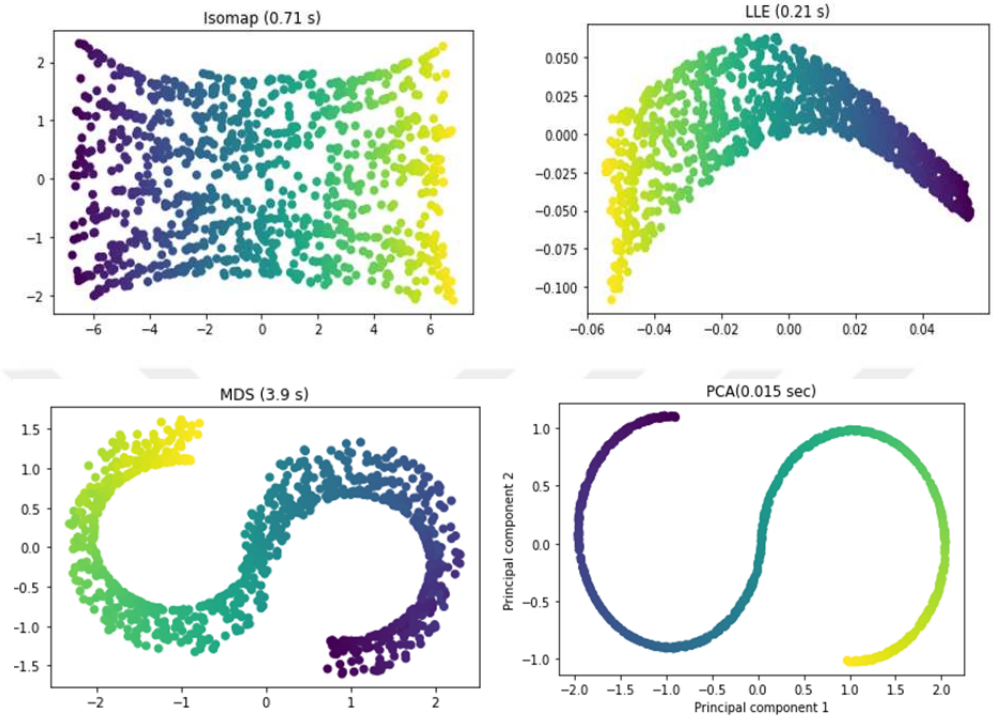

Figure 5.8. S-curve data set graphs from top left to right: a) Isomap graph b) LLE graph c) MDS graph d) PCA graph

On the other hand, we also noticed that both PCA and MDS gave close unsatisfactory results. This comes from the fact that PCA uses the orthogonal projection which becomes unuseful when the data points do not lie on or near a linear subspace of the original space. In the same way, we can say that the classıcal metric MDS doesn't give good results when dealing with a nonlinear dataset since it uses the Euclidean distance which can not preserve distances between distant data points on a nonlinear manifold.MDS requires also a huge amount of pairwise distances calculations and that is why it has the longest execution time when it is compared to other algorithms.

6. CONCLUSIONS

There are many methods in machine learning, data mining, and statistics, that are used for analyzing these data and extracting useful information from them. But if the data is high-dimensional, or when a large number of variables is involved and there is a huge amount of data, then processing this data may become exhausting, complicated, or meaningless. Therefore, applying dimensionality reduction techniques and algorithms to high-dimensional data before processing it is a good thing to start with in order to attain the desired results.

We have discussed the issue of dimensionality reduction and explained how it could be used to select features that have the main characteristics of data or as a way to extract features from data by discovering the relationships and correlations that exist between the variables in order to find a model that represents the data. It is also a way for better representation and visualization of the data when the number of variables is large or when the relationship between them is non-linear. We discussed four of the most common algorithms used for dimensionality reduction, which are principal components analysis, multidimensional scaling, locally linear embedding, and Isometric mapping. We also shed light on the mathematical and statistical foundation on which it is based. Before that, we talked about each of the two methods of projection and manifold learning to better understand the background of these methods.

We first studied the method of principal components analysis, and we saw that it is an unsupervised method which is used for dimensionality reduction and extracting features by identifying the principal components that capture the largest percentage of variability in data, and we talked about its advantages and limitations and the existing modifications that were proposed to overcome these limitations. We then talked about another unsupervised widely used Multidimensional scaling which is used as a method for visualization through calculating the pairwise distances (dissimilarities) between points, such as representing cities on maps using

the measured pairwise distances among them and we saw that MDS is equivalent to PCA when Euclidean distance is used as a metric. These two methods are used when the data is located on or near a linear subspace.

We also discussed Isomap and LLE methods, which are two modern non-linear dimensionality reduction algorithms that were proposed in the last two decades (in 2000) in case the data is non-linear or located on a non-linear manifold. The Isomap method calculates the geodesic distances on the manifold by approximating them by the sum of the Euclidean distances and by making use of the k-nearest neighbor's algorithm. We noticed that the practical applications using Isomap showed that it becomes slow when the number of data is very large and there is a modified variant of it called Landmark Isomap that is suggested in such cases. Finally, we discussed the LLE method and explained the mechanism and algorithm that it uses and the strength points of it.

Last, we applied these algorithms on two real and two artificial datasets and make a comparison between the results based on the accuracy and the execution time for each algorithm. We found that PCA did a good job on the breast cancer data set but it did not give a good result when the data was not on or near a linear lower dimensional space like in sphere and S-curve data. We also noted that MDS gave similar results to PCA in breast cancer data and gave a good representation for mapping cities in Turkish cities data by minimizing the stress but it did not give satisfying results on the nonlinear sphere and S-curve data sets. For Isomap, we noted that it gave good results on all data sets but the execution time increased when the number of data points increased. As for LLE, we noted that it did not give good results on breast cancer data, but it gave good results on both sphere and S-curve data sets. As a result, we can say that algorithms different algorithms gave similar results on some data sets and different results on other data sets based on the type of data sets and that is why it is not true to generalize these results to other data sets.

As a future study, vast research can be conducted to develop new approaches to overcome the restrictions of some of the discussed algorithms and to use the results in this thesis to develop new algorithms for dimensionality reduction.

REFERENCES

Abdi, H., & Williams, L. J. (2010). Principal component analysis. *Wiley interdisciplinary reviews: computational statistics, 2*(4), 433-459.

Alonso, M. C., Malpica, J. A., & de Agirre, A. M. (2011). *Consequences of the Hughes phenomenon on some classification techniques.* Paper presented at the Proceedings of the ASPRS 2001 annual conference.

Alpaydin, E. (2014). *Introduction to machine learning*: MIT press.

Artac, M., Jogan, M., & Leonardis, A. (2002). *Incremental PCA for online visual learning and recognition.* Paper presented at the Object recognition supported by user interaction for service robots.

Bell, J. (2020). *Machine learning: hands-on for developers and technical professionals*: John Wiley & Sons.

Bellman, R. (1957). *Dynamic programming*. Princeton University Press.

Bishop, C. M., & Nasrabadi, N. M. (2006). *Pattern recognition and machine learning* (Vol. 4): Springer.

Borg, I., & Groenen, P. J. (2005). *Modern multidimensional scaling: Theory and applications*: Springer Science & Business Media.

Borg, I., Groenen, P. J. F., & Mair, P. (2013). *Applied Multidimensional Scaling*. https://doi.org/10.1007/978-3-642-31848-1

Boyden, A., & Noble, G. K. (1933). *The relationships of some common Amphibia as determined by serological study. American Museum novitates; no. 606*. https://digitallibrary.amnh.org/bitstream/handle/2246/5358/N0606.pdf?sequence=1

Brucher, M., Heinrich, C., Heitz, F., & Armspach, J. P. (2007, September). Unsupervised nonlinear manifold learning. In *2007 IEEE International Conference on Image Processing* (Vol. 2, pp. II-109). IEE

Camastra, F., & Vinciarelli, A. (2015). *Machine learning for audio, image fduaand video analysis: theory and applications*: Springer.

Cateni, S., Vannucci, M., Vannocci, M., & Colla, V. (2012). Variable selection and feature extraction through artificial intelligence techniques. *Multivariate analysis in management, engineering and the Science*, (6), 103-118.

Chapelle, O., Scholkopf, B., & Zien, A. (2009). Semi-supervised learning (chapelle, o. et al., eds.; 2006)[book reviews]. *IEEE Transactions on Neural Networks*, *20*(3), 542.

Coombs, C. (1964). *A theory of data.* Oxford, England: Wiley.

Cox, T. F., & Cox, M. A. A. (2001). Multidimensional Scaling, Second Edition. *New York*, *88*, 328.

de Leeuw, J., & Heiser, W. (1982). 13 Theory of multidimensional scaling. *Handbook of Statistics*, *2*, 285–316.

Dikko, H., Osi, A. A., & Bello, Y. (2013). Detecting factors impacting crime rates in Nigeria using principal component analysis. *IOSR Journal of Mathematics*, *9*(3), 8–17.

Dua, D. (2017). Graff. *C., "UCI Machine Learning Repository [Http://Archive. Ics. Uci. Edu/Ml]*.

Fisher, R. A. (1922). The systematic location of genes by means of cross-over ratios. *American Naturalist* 56, 406-411.

Géron, A. (2017). Hands-on machine learning with scikit-learn and tensorflow: Concepts. *Tools, and Techniques to Build Intelligent Systems*.

Ghojogh, B., Samad, M. N., Mashhadi, S. A., Kapoor, T., Ali, W., Karray, F., & Crowley, M. (2019). *Feature Selection and Feature Extraction in Pattern Analysis: A Literature Review*. http://arxiv.org/abs/1905.02845

Groenen, P. J. F., & Borg, I. (2013). *The past, present, and future of multidimensional scaling*. Econometric Institute.

Guttman, L. (1941). The quantification of a class of attributes: A theory and method of scale construction. *The Prediciton of Personal Adjustment*.

Hotelling, H. (1933). Analysis of a complex of statistical variables into principal components. *Journal of Educational Psychology*, *24*(6), 417.

Johnston, C. S. (1995). The Rokeach value survey: underlying structure and multidimensional scaling. *The Journal of Psychology*, *129*(5), 583–597.

Khalid, S., Khalil, T., & Nasreen, S. (2014). *A survey of feature selection and feature extraction techniques in machine learning.* Paper presented at the 2014 science and information conference.

Klingberg, F. L. (1941). Studies in measurement of the relations between sovereign states. *Psychometrika* 6, 335-352.

Kohavi, R., & John, G. H. (1997). Wrappers for feature subset selection. *Artificial intelligence*, *97*(1-2), 273-324.

Köppen, M. (2000). *The curse of dimensionality.* Paper presented at the 5th online world conference on soft computing in industrial applications (WSC5).

Kruskal, J. B., & Wish, M. (1978). *Multidimensional scaling* (No. 11). Sage.

Kumar, A. C. (2009). Analysis of unsupervised dimensionality reduction techniques. *Computer science and information systems, 6*(2), 217-227.

Ladha, L., & Deepa, T. (2011). Feature selection methods and algorithms. *International journal on computer science and engineering, 3*(5), 1787-1797.

Li, J., Cheng, K., Wang, S., Morstatter, F., Trevino, R. P., Tang, J., & Liu, H. (2017). Feature selection: A data perspective. *ACM computing surveys (CSUR)*, *50*(6), 1-45.

Linting, M., Meulman, J. J., Groenen, P. J. F., & van der Kooij, A. J. (2007). Nonlinear Principal Components Analysis: Introduction and Application. *Psychological Methods*, *12*(3), 336–358. https://doi.org/10.1037/1082-989X.12.3.336

Ma, Y., & Fu, Y. (2012). *Manifold learning theory and applications* (Vol. 434): CRC press Boca Raton.

Mitchell, T. M. (1997). Machine learning. In: McGraw-hill New York.

Mohri, M., Rostamizadeh, A., & Talwalkar, A. (2018). *Foundations of machine learning*: MIT press.

Mugavin, M. E. (2008). Multidimensional scaling: A brief overview. *Nursing Research*,*57*(1),64-68

Murphy, K. P. (2012). *Machine learning: a probabilistic perspective*: MIT press.

Neto, H. V., & Nehmzow, U. (2005). Incremental PCA: An alternative approach for novelty detection. *Towards Autonomous Robotic Systems*.

Patgiri, R. (2016). *MDS: In-depth insight.* Paper presented at the 2016 International Conference on Information Technology (ICIT).

Pearson, K. (1901). LIII. On lines and planes of closest fit to systems of points in space. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, *2*(11), 559–572.

Preisendorfer, R. W., Mobley, C. D., & Barnett, T. P. (1988). The principal discriminant method of prediction: Theory and evaluation. *Journal of Geophysical Research: Atmospheres, 93*(D9), 10815-10830.

Raducanu, B., & Dornaika, F. (2012). A supervised non-linear dimensionality reduction approach for manifold learning. *Pattern Recognition, 45*(6), 2432-2444.

Rangarajan, L. (2010). Bi-level dimensionality reduction methods using feature selection and feature extraction. *International Journal of Computer Applications, 4*(2), 33-38.

Raschka, S. (2018). Model evaluation, model selection, and algorithm selection in machine learning. *arXiv preprint arXiv:1811.12808*.

Richardson, M. W. (1938). Multidimensional psychophysics. *Psychol. Bull.* 35, 659-660.

Ripley, B. D. (1996). *Pattern recognition and neural networks*: Cambridge university press.

Roweis, S. T., & Saul, L. K. (2000). Nonlinear dimensionality reduction by locally linear embedding. *Science, 290*(5500), 2323-2326.

Samuel, A. L. (1959). Machine learning. *The Technology Review*, *62*(1), 42-45.

Sanguansat, P. (2012). *Principal component analysis*: BoD–Books on Demand.

Saul, L. K., & Roweis, S. T. (2003). Think globally, fit locally: unsupervised learning of low dimensional manifolds. *Journal of machine learning research, 4*(Jun), 119-155.

Schölkopf, B., Smola, A., & Müller, K. R. (1997). Kernel principal component analysis. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *1327*, 583–588. https://doi.org/10.1007/BFB0020217/COVER

Shereena, V., & David, J. M. (2015). Significance of dimensionality reduction in image processing. *Signal & Image Processing: An International Journal (SIPIJ), 6*(3), 27-42.

H., Yin, B., Kang, Y., Shao, C., & Gui, J. (2017). Robust l-Isomap with a novel landmark selection method. *Mathematical Problems in Engineering, 2017*.

Shlens, J. (2010). A tutorial on principal component analysis. 2005. *Institute for Nonlinear Science, UCSD*.

Singh, S. (2014). Introduction to Principle Component Analysis in Applied Research. *New Man International Journal for Multidisciplinary Studies, 12*(1), 67-75.

Stewart, G. W. (1993). On the early history of the singular value decomposition. *SIAM Review*, *35*(4), 551–566. https://doi.org/10.1137/1035134

Steyvers, M. (2002). Multidimensional scaling. *Encyclopedia of cognitive science, 1*.

Sumithra, V., & Surendran, S. (2015). A review of various linear and non linear dimensionality reduction techniques. *Int J Comput Sci Inf Technol, 6*, 2354-2360.

Tenenbaum, J. B., Silva, V. d., & Langford, J. C. (2000). A global geometric framework for nonlinear dimensionality reduction. *Science, 290*(5500), 2319-2323.

Thorstensen, N. (2009). *Manifold learning and applications to shape and image processing.* Ecole des Ponts ParisTech,

Tipping, M. E., & Bishop, C. M. (1998). Mixtures of probabilistic principal component analysers.

Torgerson, W. (1958). *Theory and methods of scaling.* https://psycnet.apa.org/record/1959-07320-000

Turaga, P., Anirudh, R., & Chellappa, R. (2020). Manifold Learning. *Computer Vision: A Reference Guide*, 1-6.

Turing, A. M., & Haugeland, J. (1950). Computing machinery and intelligence. *The Turing Test: Verbal Behavior as the Hallmark of Intelligence*, 29-56.

Van Der Maaten, L., Postma, E., & Van den Herik, J. (2009). Dimensionality reduction: a comparative. *J Mach Learn Res, 10*(66-71), 13.

Xie, H., Li, J., & Xue, H. (2017). A survey of dimensionality reduction techniques based on random projection. *arXiv preprint arXiv:1706.04371*.

Zhou, S. (2003). Probabilistic analysis of kernel principal components: mixture modeling and classification. *Submitted to IEEE Transactions on Pattern Analysis and Machine Intelligence*.

CURRICULUM VITAE

Madaa ALHAJI, graduated from Manbij High School in 2007. Afterward, I started my undergraduate education at Aleppo University, Faculty of Science in 2008. I graduated from the Department of Mathematics in 2012. I am currently following for master's degree in the Statistics Department of the Institute of Natural and Applied Science at Çukurova University.

APPENDICES

APPENDIX 1: Breast Cancer Data Set Covariance Matrix

Columns From 1 to 8							
0.9982	0.3232	0.9961	0.9856	0.1703	0.5052	0.6756	0.8211
0.3232	0.9982	0.3290	0.3205	-0.0233	0.2363	0.3019	0.2929
0.9961	0.3290	0.9982	0.9848	0.2069	0.5560	0.7149	0.8495
0.9856	0.3205	0.9848	0.9982	0.1767	0.4976	0.6848	0.8218
0.1703	-0.0233	0.2069	0.1767	0.9982	0.6580	0.5211	0.5527
0.5052	0.2363	0.5560	0.4976	0.6580	0.9982	0.8816	0.8297
0.6756	0.3019	0.7149	0.6848	0.5211	0.8816	0.9982	0.9198
0.8211	0.2929	0.8495	0.8218	0.5527	0.8297	0.9198	0.9982
0.1475	0.0713	0.1827	0.1510	0.5568	0.6016	0.4998	0.4617
-0.3111	-0.0763	-0.2610	-0.2826	0.5838	0.5644	0.3362	0.1666
0.6779	0.2754	0.6905	0.7313	0.3009	0.4966	0.6308	0.6968
-0.0971	0.3857	-0.0866	-0.0662	0.0683	0.0461	0.0761	0.0214
0.6730	0.2812	0.6919	0.7254	0.2956	0.5479	0.6592	0.7094
0.7346	0.2594	0.7437	0.7987	0.2461	0.4549	0.6163	0.6891
-0.2222	0.0066	-0.2023	-0.1665	0.3318	0.1351	0.0984	0.0276
0.2056	0.1916	0.2503	0.2122	0.3184	0.7374	0.6691	0.4896
0.1939	0.1430	0.2277	0.2073	0.2480	0.5695	0.6901	0.4384
0.3755	0.1636	0.4065	0.3717	0.3800	0.6411	0.6821	0.6146
-0.1041	0.0091	-0.0815	-0.0724	0.2004	0.2296	0.1777	0.0952
-0.0426	0.0544	-0.0055	-0.0199	0.2831	0.5064	0.4485	0.2571
0.9678	0.3520	0.9678	0.9611	0.2127	0.5344	0.6870	0.8289
0.2965	0.9104	0.3025	0.2870	0.0360	0.2477	0.2994	0.2922
0.9634	0.3574	0.9687	0.9574	0.2384	0.5892	0.7283	0.8544
0.9394	0.3429	0.9399	0.9575	0.2064	0.5087	0.6748	0.8082
0.1194	0.0774	0.1503	0.1233	0.8039	0.5645	0.4480	0.4520
0.4127	0.2773	0.4550	0.3897	0.4716	0.8643	0.7536	0.6663
0.5260	0.3005	0.5629	0.5117	0.4342	0.8148	0.8825	0.7511
0.7429	0.2948	0.7699	0.7207	0.5022	0.8141	0.8598	0.9086
0.1637	0.1048	0.1888	0.1433	0.3936	0.5093	0.4087	0.3751
0.0071	0.1190	0.0509	0.0037	0.4984	0.6862	0.5140	0.3680

Columns From 9 to 16							
0.1475	-0.3111	0.6779	-0.0971	0.6730	0.7346	-0.2222	0.2056
0.0713	-0.0763	0.2754	0.3857	0.2812	0.2594	0.0066	0.1916
0.1827	-0.2610	0.6905	-0.0866	0.6919	0.7437	-0.2023	0.2503
0.1510	-0.2826	0.7313	-0.0662	0.7254	0.7987	-0.1665	0.2122
0.5568	0.5838	0.3009	0.0683	0.2956	0.2461	0.3318	0.3184
0.6016	0.5644	0.4966	0.0461	0.5479	0.4549	0.1351	0.7374
0.4998	0.3362	0.6308	0.0761	0.6592	0.6163	0.0984	0.6691
0.4617	0.1666	0.6968	0.0214	0.7094	0.6891	0.0276	0.4896
0.9982	0.4791	0.3028	0.1278	0.3133	0.2236	0.1870	0.4209
0.4791	0.9982	0.0001	0.1639	0.0398	-0.0900	0.4013	0.5589
0.3028	0.0001	0.9982	0.2129	0.9711	0.9502	0.1642	0.3554
0.1278	0.1639	0.2129	0.9982	0.2228	0.1114	0.3965	0.2313
0.3133	0.0398	0.9711	0.2228	0.9982	0.9360	0.1508	0.4156
0.2236	-0.0900	0.9502	0.1114	0.9360	0.9982	0.0750	0.2843
0.1870	0.4013	0.1642	0.3965	0.1508	0.0750	0.9982	0.3361
0.4209	0.5589	0.3554	0.2313	0.4156	0.2843	0.3361	0.9982
0.3420	0.4458	0.3318	0.1947	0.3618	0.2704	0.2682	0.7999
0.3926	0.3406	0.5124	0.2299	0.5553	0.4150	0.3279	0.7428
0.4483	0.3444	0.2401	0.4109	0.2660	0.1339	0.4128	0.3940
0.3312	0.6869	0.2274	0.2792	0.2437	0.1268	0.4266	0.8019
0.1854	-0.2532	0.7138	-0.1115	0.6960	0.7560	-0.2303	0.2042
0.0905	-0.0512	0.1945	0.4083	0.2000	0.1962	-0.0746	0.1428
0.2188	-0.2048	0.7184	-0.1021	0.7198	0.7599	-0.2169	0.2601
0.1769	-0.2314	0.7502	-0.0830	0.7294	0.8100	-0.1819	0.1990
0.4259	0.5041	0.1417	-0.0735	0.1298	0.1252	0.3139	0.2270
0.4724	0.4580	0.2866	-0.0923	0.3413	0.2828	-0.0555	0.6776
0.4330	0.3456	0.3799	-0.0688	0.4182	0.3844	-0.0582	0.6380
0.4295	0.1750	0.5301	-0.1194	0.5539	0.5372	-0.1018	0.4824
0.6986	0.3334	0.0944	-0.1280	0.1097	0.0740	-0.1072	0.2774
0.4376	0.7659	0.0495	-0.0456	0.0853	0.0175	0.1013	0.5899

Columns From 17 to 24							
0.1939	0.3755	-0.1041	-0.0426	0.9678	0.2965	0.9634	0.9394
0.1430	0.1636	0.0091	0.0544	0.3520	0.9104	0.3574	0.3429
0.2277	0.4065	-0.0815	-0.0055	0.9678	0.3025	0.9687	0.9399
0.2073	0.3717	-0.0724	-0.0199	0.9611	0.2870	0.9574	0.9575
0.2480	0.3800	0.2004	0.2831	0.2127	0.0360	0.2384	0.2064
0.5695	0.6411	0.2296	0.5064	0.5344	0.2477	0.5892	0.5087
0.6901	0.6821	0.1777	0.4485	0.6870	0.2994	0.7283	0.6748
0.4384	0.6146	0.0952	0.2571	0.8289	0.2922	0.8544	0.8082
0.3420	0.3926	0.4483	0.3312	0.1854	0.0905	0.2188	0.1769
0.4458	0.3406	0.3444	0.6869	-0.2532	-0.0512	-0.2048	-0.2314
0.3318	0.5124	0.2401	0.2274	0.7138	0.1945	0.7184	0.7502
0.1947	0.2299	0.4109	0.2792	-0.1115	0.4083	-0.1021	-0.0830
0.3618	0.5553	0.2660	0.2437	0.6960	0.2000	0.7198	0.7294
0.2704	0.4150	0.1339	0.1268	0.7560	0.1962	0.7599	0.8100
0.2682	0.3279	0.4128	0.4266	-0.2303	-0.0746	-0.2169	-0.1819
0.7999	0.7428	0.3940	0.8019	0.2042	0.1428	0.2601	0.1990
0.9982	0.7704	0.3089	0.7261	0.1866	0.1001	0.2263	0.1880
0.7704	0.9982	0.3122	0.6100	0.3575	0.0866	0.3943	0.3417
0.3089	0.3122	0.9982	0.3684	-0.1279	-0.0773	-0.1036	-0.1101
0.7261	0.6100	0.3684	0.9982	-0.0374	-0.0032	-0.0010	-0.0227
0.1866	0.3575	-0.1279	-0.0374	0.9982	0.3593	0.9920	0.9823
0.1001	0.0866	-0.0773	-0.0032	0.3593	0.9982	0.3645	0.3452
0.2263	0.3943	-0.1036	-0.0010	0.9920	0.3645	0.9982	0.9759
0.1880	0.3417	-0.1101	-0.0227	0.9823	0.3452	0.9759	0.9982
0.1682	0.2150	-0.0126	0.1703	0.2162	0.2250	0.2364	0.2088
0.4840	0.4521	0.0601	0.3895	0.4750	0.3602	0.5285	0.4375
0.6614	0.5486	0.0371	0.3793	0.5730	0.3677	0.6173	0.5424
0.4397	0.6014	-0.0304	0.2148	0.7860	0.3591	0.8149	0.7461
0.1974	0.1429	0.3887	0.1109	0.2431	0.2326	0.2690	0.2088
0.4386	0.3101	0.0779	0.5903	0.0933	0.2187	0.1387	0.0795

Columns From 25 to 30					
0.1194	0.4127	0.5260	0.7429	0.1637	0.0071
0.0774	0.2773	0.3005	0.2948	0.1048	0.1190
0.1503	0.4550	0.5629	0.7699	0.1888	0.0509
0.1233	0.3897	0.5117	0.7207	0.1433	0.0037
0.8039	0.4716	0.4342	0.5022	0.3936	0.4984
0.5645	0.8643	0.8148	0.8141	0.5093	0.6862
0.4480	0.7536	0.8825	0.8598	0.4087	0.5140
0.4520	0.6663	0.7511	0.9086	0.3751	0.3680
0.4259	0.4724	0.4330	0.4295	0.6986	0.4376
0.5041	0.4580	0.3456	0.1750	0.3334	0.7659
0.1417	0.2866	0.3799	0.5301	0.0944	0.0495
-0.0735	-0.0923	-0.0688	-0.1194	-0.1280	-0.0456
0.1298	0.3413	0.4182	0.5539	0.1097	0.0853
0.1252	0.2828	0.3844	0.5372	0.0740	0.0175
0.3139	-0.0555	-0.0582	-0.1018	-0.1072	0.1013
0.2270	0.6776	0.6380	0.4824	0.2774	0.5899
0.1682	0.4840	0.6614	0.4397	0.1974	0.4386
0.2150	0.4521	0.5486	0.6014	0.1429	0.3101
-0.0126	0.0601	0.0371	-0.0304	0.3887	0.0779
0.1703	0.3895	0.3793	0.2148	0.1109	0.5903
0.2162	0.4750	0.5730	0.7860	0.2431	0.0933
0.2250	0.3602	0.3677	0.3591	0.2326	0.2187
0.2364	0.5285	0.6173	0.8149	0.2690	0.1387
0.2088	0.4375	0.5424	0.7461	0.2088	0.0795
0.9982	0.5672	0.5176	0.5467	0.4930	0.6165
0.5672	0.9982	0.8907	0.7997	0.6134	0.8090
0.5176	0.8907	0.9982	0.8539	0.5316	0.6853
0.5467	0.7997	0.8539	0.9982	0.5016	0.5102
0.4930	0.6134	0.5316	0.5016	0.9982	0.5369
0.6165	0.8090	0.6853	0.5102	0.5369	0.9982

APPENDIX 2: Driving Distances Between 15 Turkish Cities

	Istanbul	Izmir	Bursa	Adana	Gaziantep	Konya	Antalya	Diyarbakır	Kayseri	Mersin	Eskişehir	Şanlıurfa	Malatya	Erzurum	Samsun
Istanbul	0	463	154	928	1146	710	691	1323	760	941	298	1286	1092	1226	729
Izmir	463	0	346	893	1111	556	453	1412	842	906	415	1251	1174	1581	1085
Bursa	154	346	0	804	1021	504	545	1323	691	817	152	1162	1022	1259	763
Adana	928	893	804	0	223	344	548	524	304	68	653	363	388	804	719
Gaziantep	1146	1111	1021	223	0	560	764	311	335	299	869	150	240	629	706
Konya	710	556	504	344	560	0	305	863	305	357	342	702	636	930	587
Antalya	691	453	545	548	764	305	0	1066	610	467	414	905	930	1236	893
Diyarbakır	1323	1412	1323	524	311	863	1066	0	564	601	1170	180	232	323	762
Kayseri	760	842	691	304	335	305	610	564	0	316	537	482	334	630	449
Mersin	941	906	817	68	299	357	467	601	316	0	658	435	460	876	724
Eskişehir	298	415	152	653	869	342	414	1170	537	658	0	1010	870	1105	638
Şanlıurfa	1286	1251	1162	363	150	702	905	180	482	435	1010	0	214	497	849
Malatya	1092	1174	1022	388	240	636	930	232	334	460	870	214	0	417	582
Erzurum	1226	1581	1259	804	629	930	1236	323	630	876	1105	497	417	0	553
Samsun	729	1085	763	719	706	587	893	762	449	724	638	849	582	553	0

APPENDIX 3: Distances Measured Between Cities In The Representation Space

	Istanbul	Izmir	Bursa	Adana	Gaziantep	Konya	Antalya	Diyarbakır	Kayseri	Mersin	Eskişehir	Şanlıurfa	Malatya	Erzurum	Samsun
Istanbul	0.00	488.06	169.00	951.01	1123.20	626.63	699.99	1336.34	774.98	949.94	293.16	1257.44	1101.32	1281.55	725.72
Izmir	488.06	0.00	346.31	907.16	1132.11	587.30	424.89	1426.46	872.49	872.82	398.52	1286.32	1204.94	1498.23	1044.54
Bursa	169.00	346.31	0.00	834.81	1026.32	497.76	534.11	1269.58	697.49	825.00	163.36	1169.19	1035.40	1262.65	746.01
Adana	951.01	907.16	834.81	0.00	232.51	340.65	528.66	557.92	276.39	65.79	672.41	387.26	388.89	775.75	706.85
Gaziantep	1123.20	1132.11	1026.32	232.51	0.00	551.21	761.16	332.36	359.90	285.84	863.16	155.26	224.21	609.49	720.66
Konya	626.63	587.30	497.76	340.65	551.21	0.00	300.35	839.34	304.38	327.25	338.05	703.18	621.21	946.19	633.59
Antalya	699.99	424.89	534.11	528.66	761.16	300.35	0.00	1081.45	596.72	480.72	430.77	915.78	884.20	1234.73	923.23
Diyarbakır	1336.34	1426.46	1269.58	557.92	332.36	839.34	1081.45	0.00	572.60	616.00	1111.75	201.19	235.19	378.39	763.80
Kayseri	774.98	872.49	697.49	276.39	359.90	304.38	596.72	572.60	0.00	323.22	539.16	483.49	340.03	641.81	432.60
Mersin	949.94	872.82	825.00	65.79	285.84	327.25	480.72	616.00	323.22	0.00	664.78	437.86	453.93	841.49	755.82
Eskişehir	293.16	398.52	163.36	672.41	863.16	338.05	430.77	1111.75	539.16	664.78	0.00	1006.78	878.67	1126.02	646.16
Şanlıurfa	1257.44	1286.32	1169.19	387.26	155.26	703.18	915.78	201.19	483.49	437.86	1006.78	0.00	222.93	542.18	784.34
Malatya	1101.32	1204.94	1035.40	388.89	224.21	621.21	884.20	235.19	340.03	453.93	878.67	222.93	0.00	392.60	565.78
Erzurum	1281.55	1498.23	1262.65	775.75	609.49	946.19	1234.73	378.39	641.81	841.49	1126.02	542.18	392.60	0.00	583.14
Samsun	725.72	1044.54	746.01	706.85	720.66	633.59	923.23	763.80	432.60	755.82	646.16	784.34	565.78	583.14	0.00