João Paulo dos Santos Pires

Bachelor in Computer Science Engineering

Representing Amino Acid Contacts In Protein Interfaces

Dissertation submitted in partial fulfillment
of the requirements for the degree of

Master of Science in
Computer Science and Informatics Engineering

Adviser: Ludwig Krippahl, Assistant Professor,
NOVA University of Lisbon

FACULDADE DE
CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE NOVA DE LISBOA

April, 2020

Representing Amino Acid Contacts In Protein Interfaces

This document was created using the (pdf)LaTeX processor, based in the "novathesis" template[1], developed at the Dep. Informática of FCT-NOVA [2].
[1] https://github.com/joaomlourenco/novathesis [2] http://www.di.fct.unl.pt

Acknowledgements

I would like to thank professor Ludwig Krippahl for his great help and great patience in the making of this dissertation beforehand. To my family who always was there for me and without them I would not accomplish nearly as much as I did in my life. In last but not least to my girlfriend Patricia, the person that supported me in many ways and never stopped believing me. To her, all of my love.

Abstract

Proteins are composed of twenty different types of amino acids, small organic molecules with different chemical and physical properties resulting from different groups of atoms. Protein interactions are mediated by the affinity between groups of atoms belonging to amino acid residues at the surface of each protein, in the interface region. However, it is not clear at what level these contacts are best evaluated, whether by grouping similar amino acids together, considering parts of each amino acid or even individual atoms. The number of databanks and extracted features continue to increase, this means very rich data, but that also brings the problem of the sheer amount of different features and what do they really represent in the *big picture* of protein interactions.Since the data itself is collected by scientific communities all around the globe, there is a vast amount of information but with that there is also a great diversity of the measured or calculated attributes. This creates a need to learn at which level these contacts occur and what is the best way to combine the information in the literature to learn a valuable representation. With the rise of machine learning algorithms making possible to work with data in various ways that were not previously possible due to practical limitations, various areas are using these algorithms to capture information about the data that was inaccessible before, bioinformatics being one of them. The goal of this work is to use unsupervised deep learning techniques that transform the data in a way that is intended to be informative and non-redundant, facilitating the subsequent learning for other algorithms of classification or regression that will perform better on processed data like this. The transformation involves finding encodings for the collected features that best capture which are the ones that are actually relevant to construct these encodings. These encondings can be latent in relation to the already known information in the area, meaning that they most likely will not be *human friendly*, in the sense that they will lack interpretability for humans, but can increase the performance of machine learning algorithms.

Keywords: Protein, Amino Acid, Atom, Protein Interface, Protein interaction, Deep Learning, Unsupervised Learning, Feature Extraction ...

Resumo

Proteínas são compostas por vinte tipos diferentes de amino ácidos, pequenas moleculas orgânicas e com diferentes propriedades químicas e fisicas resultantes de diferentes grupos de amino ácidos. As interações de proteínas são mediadas entre afinidades entre grupos de átomos pertencentes ao resíduos de amino ácidos à superficie de cada proteína, na região da interface. Mas não é claro a que nível é que estes contatos são melhor avaliados, se por agrupar amino ácidos juntos, considerando apenas partes de cada amino ácido ou ainda átomos individuais. O número de bancos de data e características extraídas continuam a aumentar, significando data muito enriquecida, mas também carrega o problema da quantidade de características e o que elas realmente representam na visão geral das interações de proteínas. Como a data por si própria é colectada manualmente por comunidades científicas por todo o mundo, existe uma grande quantidade de informação mas com isso também uma diversidade elevada de atributos medidos ou calculados. Isto cria uma necessidade de aprender a que nível estes contatos ocorrem e qual é a melhor maneira para combinar a informação na literatura para aprender um representação mais valiosa. Com a subida de algoritmos de *machine learning* deixando possível trabalhar com data em maneiras variadas que não eram possíveis anteriormente devido aa limitações práticas, várias áreas estão a usar estes algoritmos para capturar informação sobre a data que estava inacessível antes, com bioinformática sendo uma dessas áreas. O objectivo deste trabalho é usar técnicas de *deep learning* não supervisionado para transformar a data numa maneira que se pretende que seja informativa e não seja redundante, facilitando assim aprendizagem subsequente para outros algoritmos de classificação e regressão que oferecem melhores resultados em data processada como esta. A transformação envolve encontrar encodações para as características recolhidas que melhor capturam quais são as que são realmente relevantes para construir estas encodações. Estas encodações podem ser latentes em relação à informação já conhecida na área, significando que não vão ser *human friendly*, no sentido que não vão ter interpretabilidade para humanos, mas podem aumentar a performance dos algoritmos de *machine learning*.

Palavras-chave: Proteínas, Amino Ácidos, Interface de Proteínas, Interações de Proteínas, Deep Learning, Aprendizagem Não Supervisionada, Extração de Features ...

Contents

List of Figures

List of Tables

Glossary

Amino Acid Index	One of the major online databanks of amino acid features.
Amino Acid	The building blocks of proteins that are organic compounds containing amine and carboxyl functional groups, along with a side chain specific to each amino acid.
Feature Extraction	Transforming the original data features into more useful features.
Manifold	Topological space that resembles euclidean space near each point, and can be perceived like a surface of any shape.
Neural Networks	A network of simple elements called neurons, which receive input, change their internal state according to that input, and produce output depending on the input.
Protein Data Bank	One of the major online databanks of protein structures.
Protein Hot Spot	Resisues inside the interface that contribute more than the rest of the residues to the interactions between proteins.
Protein Interface	The part of a protein where the interactions with another protein happen.
Solvent Accessible Surface	Area of the surface that is accessible to a solvent, in most cases water.
Unsupervised Learning	A section of machine learning that tries to find a representation of data that is more useful than the data itself.
Van der Walls Force	Force dependent on the distance between the atoms or molecules that are interacting.

Acronyms

AAIndex	Amino Acid Index.
ASA	Accessible Surface Area.
CAS	Computer Algebra System.
ExPASy	Expert Analysis System.
KNN	K-Nearest Neighbours.
LTSA	Local Tangent Space Alignment.
MDS	Multi Dimensional Scaling.
NFLT	No Free Lunch Theorems.
PCA	Principal Component Analysis.
PDB	Protein Data Bank.
PISA	Protein Interfaces Surfaces and Assemblies.
t-SNE	t-Distributed Stochastic Neighbor Embedding.

Chapter 1

Introduction

In this introduction it will be explained the context involving protein interactions and the motivation and objectives behind this subject that is one of the most researched in the bio-informatic field.

1.1 Objectives

The goal of this work is to use unsupervised deep learning techniques, more especially autoencoders, to reduce the number of features used to describe amino acid residues to model contacts between proteins. The data used for this work will be collected from online databanks, specified in section 2.1.1, that have stored a great amount of protein structures with features, measured and calculated manually using experimental techniques, such as hidrophobicity and Van der Walls force, two examples of a set of features with around 70 features, many of each have different measurements and formulas for obtation for the same feature, collected from verified sources specified in section 2.1.2. A comparison with other unsupervised deep learning techniques that are specified in section 2.2, Principal Component Analisys and Isomap between others, will also be made, so various baselines can be established for the reduction comparison. Finally a estimation of improvements using classification algorithms, already created by third-parties, mentioned in section 2.2.6, with the reduced features will also be realized and the results are going to be evaluated and reported in the next part of this dissertation.

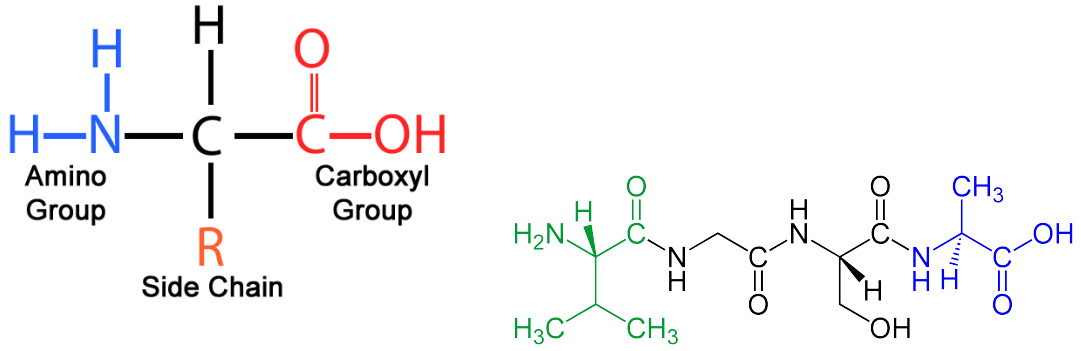
1.2 Context

Proteins are highly complex molecules that do a vast array of functions in biochemistry and are directly involved with the processes essential to life. The term, coined by Jons

Jacob Berzelius derives from the Greek word *proteios*, meaning "holding first place"[1].

1.2.1 Amino Acids

Amino acids are the building blocks of proteins. There have been found more than 300 different amino acids but, of those, only 20 are involved in protein synthesis[2]. Amino acids are composed of a central carbon (**C**) atom bonded with a carboxyl group(**COOH**), an amino group(**NH2**) and a side-chain or **R** group, that group being what differentiates all amino acids. Amino acids are joined together by a condensation process in which the amino group of one amino acid forms a peptide bond with the carboxyl group of another amino acid. This process, happening several times, results in a chain of amino acids, by the name polypetide chain. One single polypetide chain can originate a protein, but normally these chains group together to form more complex proteins[2].



(a) General representation of an amino acid (b) Three amino acids forming a polypetide chain

Figure 1.1: Amino acids exemplified [3]

1.2.2 Structural Regions of Proteins

To understand the relationship between the sequence, structure and function of a protein is one of the main focus of biochemistry[4]. Associated with the structural part of the relationship is the need to segregate regions of the proteins accordingly with their functions. The greater majority of the scientific community agrees in the division of a protein, in an interaction context, in three main sectors: surface, interior and interface[4–6].

1.2.2.1 Surface and Interior

In a simplistic manner the interior of a protein is the region of that protein that is buried beneath the surface. Along the years different approaches to find features that best separate these two regions have been studied. Chothia concluded, in 1976, in his studies, that the average residues in the surface are polar and the ones in the interior are apolar [7]. Young, Jernigan and Covell investigated, in 1994, the hidrophobicity of

the residues from the two regions concluding that the surface is mainly constitued by hidrophilic residues and the interior by hidrophobic ones [8]. Bogan and Thorn more recently, in 1998, considered a residue to be buried if his relative accessible surface area is below a certain threshold that best divides these two regions [5]. Acessible surface area, or ASA, is an area accessible to water, the most common biological solvent, where the perimeter is defined around the van der Waals surface, which is explained in chapter 2, with a probe sphere of 1.4Å (Ångström), being 1Å = $10^{-10}m$, that rolls around it. This value is an aproximation to the radius of a water molecule.

1.2.2.2 Interface

As stated before, most proteins interact with other proteins to perform biological functions. Not all of the residues in a protein contribute in the same manner to those interactions. One of the earliest mention of protein's interfaces depicted them as residues that are below a threshold value of distance to the residues of another protein. Other distinction between interface and the other regions is often depicted by a $\Delta rASA$ cutoff value that is equal to the difference between the $rASA$ (relative ASA) in monomer and the $rASA$ in complex state[4].

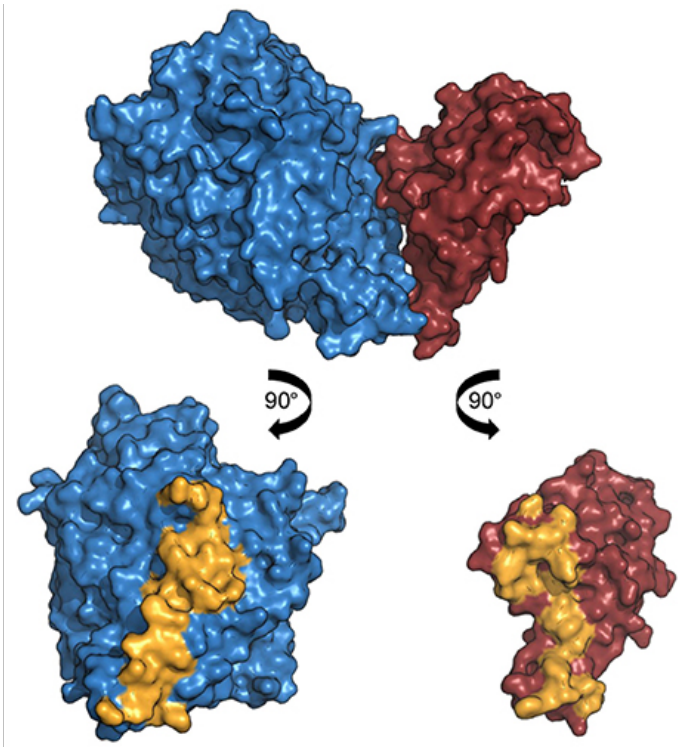


Figure 1.2: Interface of two proteins shown in yellow [9]

The standard size of an interface sits roughly between 1200 and 2000 Å^2, whereas smaller interfaces have low-stability and have a short life and bigger interfaces occur mostly between G-proteins and other components of signal transducers and between proteases and one class of their inhibitors [6, 10, 11]. On the topic of interaction stability,

one can separate interfaces in two types: permanents and transients. As the names suggest, the permanent interfaces need another protein interface to maintain itself on a complexed state in order to keep their structure and functions. They can't be found *in vivo* uncomplexed. Transients interfaces can exist in both complexed and uncomplexed states, making possible to these interfaces to interact with different molecules through their life.

1.2.3 Hot Spots

As stated before not all residues in a protein contribute in the same manner to the binding free energy of an interaction. That's true also for the interface where there are residues that contribute more than others. Ofran and Rost[12] divided protein-protein interfaces into six types: intra-domain, domain-domain, homo-oligomer, hetero-oligomer, homo-complex, and hetero-complex. The division was based on structural differences, and based on that division, they analized the type of contacts that the residues have with other residues of the different regions. This viewpoint already let us see the different residues properties in the interface although it really doesn't "pick"important residues and segregates them of all the residues in the interface, it segregates all residues in groups.

The first reference to these important residues was from Clackson and Wells[13] that coined the term *Hot Spots*. They tested with the alanine mutation technique, which consists in mutate the side-chains of the peptyde, deleting that way, all interactions made by atoms beyond the β carbon revealing the contribution of binding energy of the removed portion of the side chain. At this point they did not know what type of features were most relevant to identify hot spots.

Chakrabarti and Janin[10] dissected the interfaces into a core and a rim based on solvent accessibility. The core contains atoms that are buried on complex formation and is surrounded by a rim of atoms that remains partly accessible. If the residue contains at least one buried interface atom it is considered to be part of the core else it is considered part of the rim. They also noted that the atom compositions of the rim resembles the rest of the protein surface and the core with an excess of aromatic residues and a deficit of charged residues except *Arg*.

Levy[4] continued this concept of rim-core differentiation and suggested a third region called support. It was considered the accessible surface area for the distinction where the support residues are already largely buried in the monomer, when the proteins are not interacting, and become more buried in the complex, when the proteins are interacting. The rim residues are largely exposed in the monomer and remain exposed in the complex. The core residues shift from being exposed in the monomer to being buried in the complex. With this new model the amino acid composition of the rim and the support are nearly identical with those of the surface and the interior, leaving the core residues the most distant in this manner. It was noted that the relative contribution of the interface and rim decreases wit interface size, which helps explain why smaller interfaces

are generally more polar than larger interfaces.

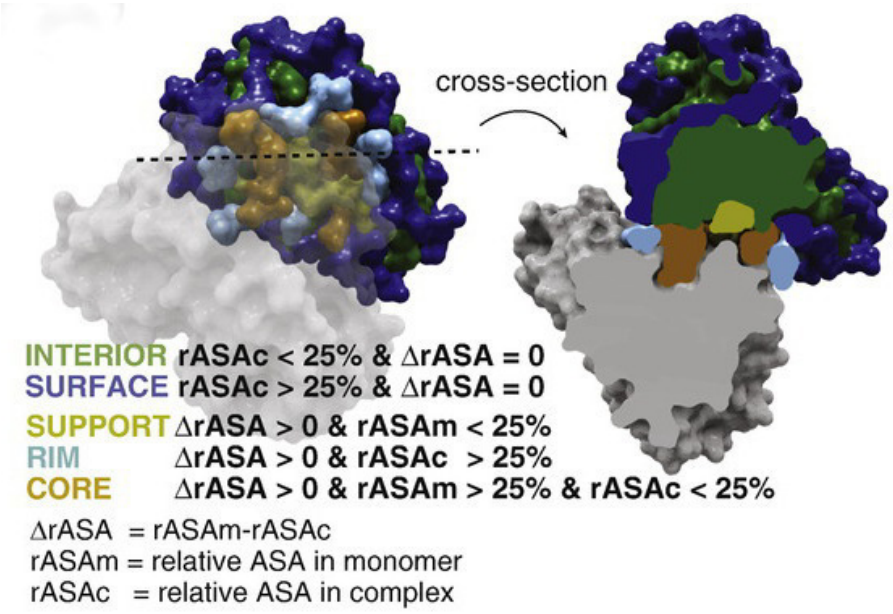


Figure 1.3: Cross section of a protein complex [4]

1.3 Motivation

Proteins are composed of more than twenty different types of amino acids, small organic molecules with different chemical and physical properties resulting from different groups of atoms [1]. Protein interaction prediction is important for the investigation of intracellular signaling pathways, modelling of protein complex structures which are a group of two or more associated polypeptide chains formed by short chains of amino acid monomers[14], and for gaining insights into various biochemical processes[15]. These protein interactions are mediated by the affinity between groups of atoms belonging to amino acid residues at the surface of each protein, in the interface region[16]. However, it is not clear at which level these contacts are best evaluated, whether by grouping similar amino acids together or considering only certain parts of each amino acid or even individual atoms [4–6, 17, 18]. Protein-protein interactions are essential in all cellular processes. Mutations on the genetic code can cause proteins to disrupt themselves which often leads to some form of disease. Perceiving at what level these interactions occur is growing in importance in today's molecular biology community. The computers *boom* in the early 2000's turned possible for people from all over the world to contribute to this field, by using computational methods for modeling protein complexes requiring the data from the structure of a protein's components combined with the sequences of amino acids data extracted from other methods[19]. Although the number of features of data related to proteins is already very broad and have an extensive literature associated the data itself can be raw, unstructured, or noisy. For that reason it is of great importance to

extract salient and informative features from the input data while discarding redundant and noisy information, so that they can be used further in predictive algorithms. While most of the time these salient features can be uninterpretable to humans, to machines, they can be interpreted and have much more usefulness.

1.4 Thesis Structure

This thesis consists of 4 core components:

Context: In this section it is presented pertinent information about the world of proteins that will be utilized in the next sections.

State of the Art: In this section it is presented several techniques that explore the problem in question in numerous ways and they will be studied and performance metrics will be utilized to judge them.

Experimental Work: In this section it is presented the experimental work that was done for the preparation of this dissertation

Conclusion and Future Work: In this final chapter it is presented the conclusions that are relative to the results and the future work that can be done following this dissertation.

Chapter 2

State of Art

2.1 Data

Every deep learning algorithm needs data, this case not being an exception. Protein data banks were created to storage data collected generally from individual scientific publications that validate interactions between some proteins. A good dataset needs to fill certain characteristics, namely, the data needs to be directly relevant to the problem imposed, resembling as much as possible real-world data. It also needs to have a good coverage of the input space that we care about, meaning a big representation of values across all the features. For these reasons this section as the purpose of showing different reliable sources of data and the most common relevant features used in the literature.

2.1.1 Data Sources

In this subsection there will be presented the different data sources considered for this work. Many of the features can be extracted from more than one source, but there are a significant number of features that are represented in only one of the sources. For this reason is important to gather a reasonable number of diferent sources to enrich the number of features for this work.

2.1.1.1 Protein Data Bank

Created in 1971, the PDB exists for having a free and publicly available to the community around the world single source of information about the 3D structures of proteins, nucleic acids, and complex assemblies [20]. Software developers and users of the PDB will be presented with consistent data consequence of the formal mechanism for standardizing the presentation of the data. As of 26 June 2018 this data bank has approximately 147000 structures where around 90% is protein structures. This is one of the most utilized

data bank with around 679,421,200 downloads of the information just in 2017. Also, for each protein that enters this data bank, it is assigned a unique **PDB_ID**, created with the purpose of unifying diferent databanks under the same standard. This feature makes this databank very useful since other databanks that are going to be explored identify protein complexes with the **PDB_ID** also. It will be the principal source for protein structures altough it does not have many features available.

2.1.1.2 Dockground

Dockground is a database with several curated complexes [21], which, for the purpose of this dissertation will be useful to acquire complexes with one base chain and for each one of these base chains has one chain that is correct in the meaning that can interact with the first chain and it has others chains that are used as decoys to be tested against the correct one. This data bank is used to benchmark classifiers so they can identify correctly the correct ones. For the purpose of this dissertation it can be used to create correct and false contacts to predict them in the classifier.

2.1.1.3 AAIndex

Amino Acid Index, or AAIndex, is a database that holds information for representing various physicochemical and biochemical properties of amino acids and pairs of amino acids [22]. Currently it is subdivided in three sections, two of each will be of use for the context of this work:

- **AAIndex1:** Containing, as of today, 544 amino acid indices, each entry consisting in a description of the index, references with information about it and the values for the properties of the 20 amino acids. In addition to this information, there is, for each entry, cross-links to other entries with a value for the correlation coefficient of 0.8 or larger. This is useful since it enables the users to identify a set of entries with similar properties.
- **AAIndex3:** Containing, as of today, 47 amino acid contact potencial matrices, the entry contains 210 values for a symmetric matrix and 400 or more values for a non-symmetric matrix, with each value representing the statistical contact potential between the amino acids.

This databank will prove very useful for extraction of values from different properties from each type of amino acid, that will be used for training the models in section 2.2.

2.1.2 Features

In this subsection it will be presented some features that are relevant for representing the proteins. There are several more, with *AAIndex*, having a list of more than five hundred different features (altough much of them are calculated from others, which are going to be mostly discarded by the algorithms explained in section 2.2), but these ones are the most referenced in the literature.

- **Ionic Bonding:** A chemical bonding that happens between oppositely charged ions,by a electrostatic attraction, and is the primary interaction in ionic compounds. Resulting from a redox reaction when atoms of an element, whose ionization energy is low, give some of their electrons to reach a stable electron configuration, forming cations for the atom that gave the electrons and anions for the atom that receive them.
- **Van der Walls Force:** This force is dependent on the distance between the atoms or molecules that are interacting. Unlike ionic bonding, this force is not electrochemical and is more susceptible to being disturbed, vanishing at longer distances between interacting molecules. This kind of force results from a transient shift in electron density.
- **Hydrogen Bonding:** A hydrogen bond is a partially electrostatic attraction between a hydrogen (H) atom which is bound to a more electronegative atom or group, such as nitrogen (N), oxygen (O), or fluorine (F),referred to as the hydrogen bond donor, and another adjacent atom bearing a lone pair of electrons, referred to as the hydrogen bond acceptor. Hydrogen bonds can be intermolecular, where they occur between separate molecules or intramolecular, where they occur among parts of the same molecule. They are stronger than a Van der Waals interaction, and weaker than ionic bonds.
- **Salt Bridges:** A salt bridge is a non-covalent interaction between two ionized sites of a molecule. It has two components: a hydrogen bond and an electrostatic interaction. In a salt bridge, a proton migrates from a carboxylic acid group to a primary amine or to the guanidine group. Of all the non-covalent interactions, salt bridges are among the strongest.
- **Solvent Accessible Surface:** The solvent accessible surface area is the area of the surface that is accessible to a solvent, in most cases water. To calculate this value one needs to consider the radius of a water molecule, which is 1.4 Ångström approximately, and drawing an equidistant line from each atom of the molecule beyond the van der Walls radius, which can be considered like rolling a ball along the surface.
- **Hidrophobic Interactions:** The hydrophobic interaction is an entropic effect originating from the disruption of the dynamic hydrogen bonds between molecules of liquid water and the nonpolar solutes. The structure formed is more highly ordered than free water molecules due to the water molecules arranging themselves to interact as much as possible with themselves, and thus results in a higher entropic state which causes non-polar molecules to clump together to reduce the surface area exposed to water and decrease the entropy of the molecular system.

2.2 Unsupervised Learning Algorithms

The task of an unsupervised learning algorithm normally is to find a representation of data that preserves as much information about the data but with some type of constraint

or penalty with the objective to make the representation more accessible and manageable than the data itself. Being more accessible is a ambiguous way of defining the representation, nevertheless there are three common ideas, reduction of the dimensional representation where the information about the data is compressed, distribution of the data along the axes of the representation space and trying to find a representation that have statistically independent dimensions [23]. Below are described some algorithms that can be proved useful for the purpose of this dissertation.

2.2.1 First Considerations

Prior to the explanation of the algorithms that can be used to represent the data set, first there is a need to tackle some concepts that are of relevance to the problem in hands and are in a way or another involved with each of the algorithms.

2.2.1.1 No Free Lunch Theorems

The no free lunch theorems, or NFLT, are a set of mathematical proofs that examine general-purpose algorithms, or *black-box* algorithms, and the problems that they are trying to solve. Generally speaking, the main idea behind these theorems is that, given an algorithm that searches for an optimal cost or fitness solution is not universally superior to any other algorithm. As stated by Wolpert and Macready [24],

"NFL theorems mean that if an algorithm does particularly well on average for one class of problems then it must do worse on average over the remaining problems. In particular, if an algorithm performs better than random search on some class of problems then it must perform worse than random search on the remaining problems. Thus comparisons reporting the performance of a particular algorithm with a particular parameter setting on a few sample problems are of limited utility. While such results do indicate behaviour on the narrow range of problems considered, one should be very wary of trying to generalize those results to other problems."

With these theorems rises the question of which type of algorithm one should try to model, one which primes to have a high level of generality, where the algorithm is a *jack-of-all-trades*, but a master of none, or another, which primes to have a high level of specificity, where the space of problems is reduced to only a cluster of very similar problems [25].

To avoid the problem of overfitting the data, with a model *memoryzing* the specific input data instead of lerning from it, there are some techinques that one can use. The most common technique that can be applied to almost every machine learning problem consists in dividing the dataset into three parts, the training set, validation set and test set. The model will only learn from the training set while using the validation set to track the progress to select models or optimize hyperparameters. The test set is then used after the training of the model to evaluate the performance of it. It is important that the validation set and the test set come from the same distribution and that they reflect the data that the

model is supposed to receive in the future. However, by partitioning the available data into three sets, the number of training samples is reduced which can be affect the learning of the model. Another approach is k-fold cross validation, where the training set is split into k smaller sets, called folds, and each one goes through the same procedure, in which the model is trained using $k-1$ folds as training data. The model is then validated on the remaining fold. The performance measure of the k-fold cross validation is the average of the values computed. This approach can be expensive but does not waste too much data like the other solution.

2.2.1.2 Curse of Dimensionality

Another concept inherent to machine learning is the curse of dimensionality. First of all, the definition of dimensionality refers to the minimum number of coordinates needed to specify any point within a space. Data has dimensionality to it. The more dimensions, features like hydrophobicity and charge, that are in a data set, the more sparsity is observed in the data as well, making the job to find patterns in the data more difficult and complex leading the algorithms designed to deal with high-dimensional data to have a very high time complexity. As shown in figure reffig:protein-interface the exponential increase in the size of the learning data needed by an algorithm is one consequence of the curse of dimensionality [26].

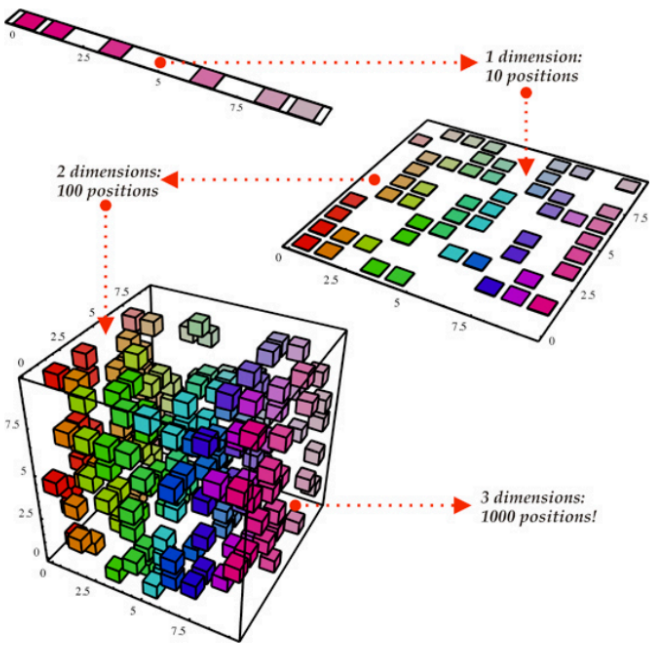


Figure 2.1: Demonstration of the curse of dimensionality paradigm [27]

In other words, relevant generalization is possible from interpolation, the numerical method of calculation of values that lie somewhere in the middle of the given discrete set of data points, but not from extrapolation, the numerical method of that calculates

points that are outside the range of the given set of discrete data points by using relevant methods of assumption [28]. The Hughes phenomenon states that, with a fixed size of training data, a classifier or regressor predictive power will increase and then decreases as the number of dimensions grow larger [29]. To summarize, the curse of dimensionality is the expression of all phenomena that appear with high-dimensional data, and that in most cases, have unwanted consequences on the behaviour and performance of learning algorithms [30].

2.2.1.3 Dimensionality Reduction

As stated in section 2.2.1.2 the curse of dimensionality is an unavoidable challenge when one is trying to model an algorithm that deals with high-dimensional data. The idea behind this method lies in the belief that there may be too many features for the available data, leading to overfitting or some features may be too noisy or even they are costly to measure [31].

There are two main ways to reduce dimensionality:

- **Feature Selection:** By only keeping the most relevant variables from the original dataset the computational load is reduced making the algorithm achieve greater performance. More importantly the irrelevant features may lead to overfitting, leading the model infering false conclusions about their relationship with the data. There are also methods that the main focus is to find out how many features are necessary to represent the data without losing much information, employing heuristics to locate the optimal number and combination of features [32].
- **Feature Extraction:** The main idea is to transform the original data into a more useful data set. This is achieved by using a function that combines, linearly or non-linearly, the original data, the input, into a new set of features, the output. The new set is intended to be informative and non-redundant, facilitating the subsequent learning and generalization step [33].

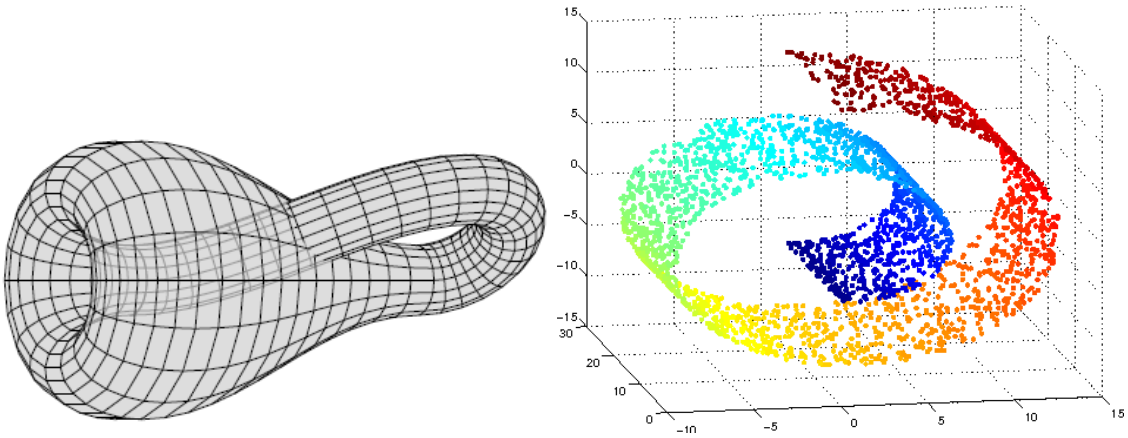
The method that will be needed to help solve the problem in this dissertation is feature extraction since we want to extract meaningful information so classification algorithms can achieve better results and performance. Also the goal of this work is to reduce the number of features by transforming them in new features instead of discarding features and maintaining others.

2.2.1.4 Manifold

As stated in section 2.2.1.3, a given high-dimensional dataset may contain many features that are all from measurements taken, that are related to the same underlying cause. The manifold hypothesis [34, 35] describes that the data generating distribution is assumed to concentrate near regions of low dimensionality.

In mathematics, a manifold is a topological space that resembles Euclidean space near each point [36], and can be perceived like a surface of any shape, in layman's terms.

The use of the term manifold in machine learning is looser than its use in mathematics for practical reasons, where the data may not be strictly on the manifold, but only near it, the dimensionality may not be the same everywhere and the notion referred to in machine learning extends to discrete spaces [23]. The dataset lies along a low-dimensional manifold embedded in a high-dimensional space, where the low-dimensional space reflects the underlying parameters and high-dimensional space is the feature space.



(a) A klein bottle is an example of a manifold [37]. (b) A manifold in a shape of a swiss-roll [38].

Figure 2.2: Manifold exemplified

2.2.2 Principal Component Analisys

Principal Component Analisys, or PCA, is a method of identifying patterns in data, excelling at data with high dimensionality where graphical representation is not an option. The goal of PCA is to extract the most important information, while compressing the size of the dataset and then analyze the structure of the observations reducing this way the possibility of overfitting [39].

First consider a X dataset that as mxn size where m represents the samples, a protein residue is a sample for example, and n represents the various features. PCA gives a choice on which type of matrix to use for analizying the data points, the covariance matrix and the correlation matrix. The covariance matrix retains the units of measurement, meaning that the different features must be comparable between themselves, it also makes changes to the scale, even by a similar constant, of the features, resulting in different results. The correlation matrix is dimensionless since it divides the value of covariance by the product of standard deviations which have the same units and the result is not influenced by a changing in the scale of the values. Since the features presented in section 2.1.2 do not have the same type of unit measurement, the use of correlation matrix may be more appropriate [40].

The *core* of PCA are the eigenvectors, or in this algorithm also called principal components, that represent the directions of the new feature space and the eigenvalues

that represent their magnitude explaining the variance of the data along the new feature axes. These are obtained by performing a eigendecomposition on the covariance matrix Σ, a matrix of size dxd with each element representing a covariance between two features. After having both eigenvalues and eigenvectors it is time to reduce the dimensionality of the original feature space. Since the eigenvectors have all a unit length of one, they can't be used to predict which ones are better than others at conserving the most variance between the data points as possible, there is a need to analize the correspondent eigenvalues. This analysis consists in sorting the values in a descent manner by value and choose the top eigenvectors pretended. The number of eigenvectors to use depends for each context case and which level of variance is pretended to be saved. The most common way to find this number is just by calculating the explained variance from the eigenvalues that reveals how much variance can be related to each principal component.

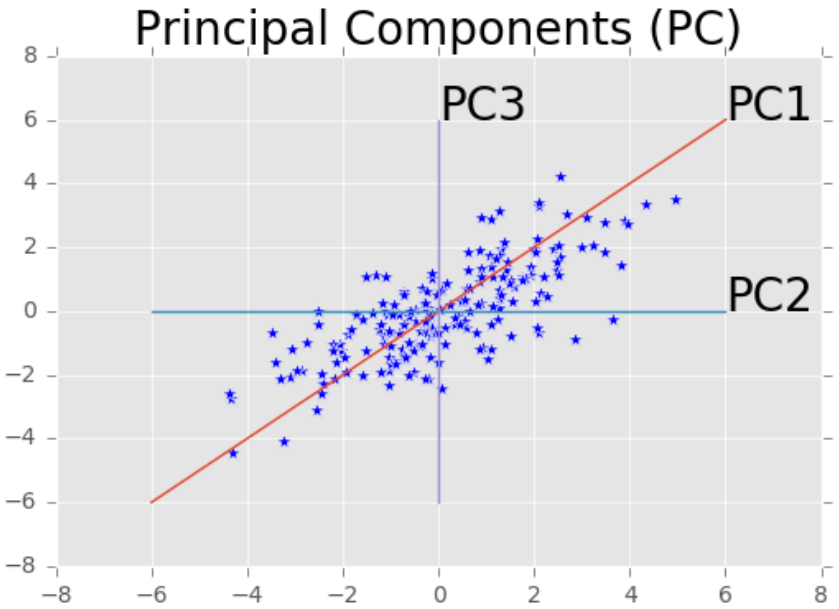


Figure 2.3: Diagram showing three principal components. The order of the principal components follow the highest variance of the data [41].

PCA tries to find a linear subspace of lower dimensionality, such that the largest variance of the original data is kept. However, it has to be noted that the largest variance of the data does not necessarily represent the most discriminative information. Linear subspaces may not adequately represent the the underlying manifold on a dataset which may have some other nonlinear structure. Even when a linear projection on the data can find a representation with some number of dimensions, there is still a possibility of finding a more efficient representation that captures the data even better using a lower dimensional manifold. For this reason another algorithm is needed, one that finds a nonlinear subspace. It will be interesting to analyze what kind of lower dimension best represents the dataset.

2.2.3 Isomap

Isomap stands for isometric mapping and it is a nonlinear dimensionality reduction algorithm by trying to preserve the geodesic distances in a lower dimension. Due to euclidean distances being highly misleading in a nonlinear data structure, Isomap uses the geodesic distance, a distance that "follows"the manifold and because of that, holds off on the same data structure, giving a better estimation on the distance of two points [42]. This can be observed in figure 2.4.

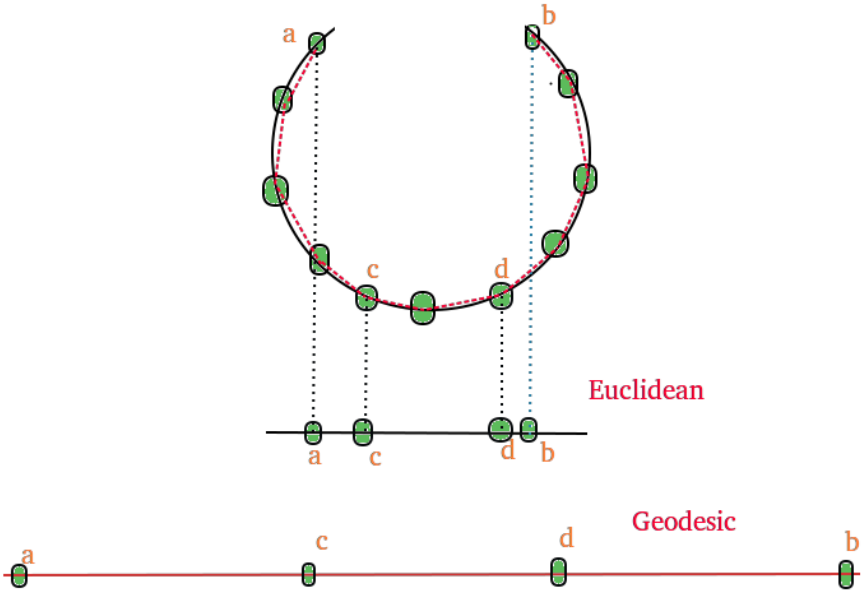


Figure 2.4: Differences between 1-D mappings of the two distance metrics. The mapping obtained with the euclidean distance gives off incorrect distances between the points while the geodesic distance gives a very accurate distance between the same points [43]

For this, first the algorithm must determine the neighbors of each point, either by a fixed radius or by using the k-nearest neighbors algorithm, or k-NN. In both situations one has to choose the lenght of the radius or the number of k neighbors. In k-NN a point is classified by a plurality vote of its neighbors, with the object being assigned to the class most common among its k-nearest neighbors. Then these neighborhood relation are used to construct a weighted graph. For the next step, an estimation of the geodesic distance between all pairs of points is needed while making sure that the resulting graph is fully connected. To calculate the shortest distance one can use the Djkstra's algorithm, an algorithm that finds a shortest path tree from a single source node, by building a set of nodes that have minimum distance from the source. Finally one needs to compute a lower-dimensional embedding. This can be achieved by using the multidimensional scaling

algorithm, or MDS, an algorithm that tries to find a set of vectors in p-dimensional space such that the matrix of euclidean distances among them corresponds as closely as possible to some function of the input matrix according to a criterion function called stress. The smaller the value returned by the stress function, the greater the correspondance between the two points. This algorithm can be perceived as a mathematical operation that converts a point-by-point matrix into a point-by-feature matrix.

Isomap gives the advantage of finding a nonlinear representation of the data, whereas PCA cannot. But this comes with extra concerns, namely, the sensitivy to noise where a few outlier points can break the mapping, the few free parameters that one can change making the algorithm rely mostly on the choice of the radius length, or k for the k-NN, and also the fact that Isomap usually performs poorly when the manifold is not well sampled and contains holes. For these reasons a final algorithm will be presented, which can function in a nonlinear subspace and do not have some of the requirements of Isomap.

2.2.4 Other Algorithms

In this subsections it will be presented algorithms that are fully implemented by python packages referred in section 2.3.5.They will be used as comparison to the autoencoders that will be the main focus of this work.

2.2.4.1 t-Distributed Stochastic Neighbor Embedding

The aim of t-Distributed Stochastic Neighbor Embedding, or t-SNE, is to extract clustered local groups of samples, which can be beneficial to disentangle data that have many manifolds associated to them [44]. To achieve this t-SNE converts similarities between data points to joint probabilities, and with that, tries to minimize the Kullback-Leibler divergence between the joint probabilities of the higher-dimensional data and the lower-dimensional projected data. Since this divergence is not convex, diferent runs of the algorithm will return different results, but it is perfectly fine to run t-SNE several times, and select the solution with the lowest Kullback-Leibler divergence. This algorithm is computationally expensive, so passing first the data through a PCA algorithm can improve significatily that.

2.2.4.2 Multi Dimensional Scaling

The goal of Multi Dimensional Scaling, or MDS, is to place each data point in a lower-dimensional space such that the distances between the data points are preserved as well as possible in relation to the original space [45]. There are different variants of this algorithm, including the metric multi dimensional scaling and the non-metric multi dimensional scaling. The first preserves the original distance metric, between points, as well as possible. That means that the distances in the higher-dimensional space are in the same metric as the ones of the lower-dimensional projected data. To the second

variant the important is not the metric of a distance value, but its value in relation to the distances between other pairs of data points. This means that if the distances between two different data points rank x^{th} in the higher-dimensional space then they also have to rank x^{th} in the lower-dimensional projected data.

2.2.4.3 Local Tangent Space Alignment

The goal of Local Tangent Space Alignment, or LTSA, is based on the intuition that when a manifold is correctly unfolded, all of the tangent hyperplanes to the manifold will become aligned [46]. Like others algorithms described in this section first it will start by computing the neighborhood for each data point. Next it will calculate the local geometry of each neighborhood by its tangent space. Finally the algorithm will perform a global optimization to align all the local tangent spaces.

2.2.5 Auto Encoders

With the rise of processing power of computers neural networks became a recurrent topic in deep learning, since it offers advantages in pattern recognition that other unsupervised learning algorithms cannot, like the performance of the algorithm increase with the quantity of data available. An autoencoder is essentally a neural network that has the objective of copying is input to the output. The usefulness of an autoencoder relies on having a hidden layer that is smaller than the input layer, imposing the creation of a more compressed representation of the data. When the data has some latent representation structure, i.e. correlations between input features, it can be learned in the bottleneck of the autoencoder, also refered as code size, which is the smaller layer of an autoencoder. There are two important operations in an autoencoder: the encoder, responsible for compressing the input into a latent-space representation and the decoder, responsible for reconstructing the output from the latent-space representation. The decoder is symmetric to the encoder in terms of the layer structure [23].

With this the autoencoder can be represented as $o = g(f(i))$, where o tries to be as close as possible to i. That means that the algorithm will have to encapsulate the information from the input in h, saving in this representation the information in a way that can be used to generate an output very close to the input. An autoencoder can be trained by minimizing the reconstruction error, $L(o, i)$, which measures the differences between the input and the reconstruction. When constructing the model off an autoencoder one must balance the sensivity to the needs of the problem. The model must be sensitive enough in relation to the input so its reconstruction is accurate, but not so sensitive so the model does not simply copy the training data and be overfitting. This trade-off can be accomplished by using a loss function, a function that punishes the model when the output deviates from the input, combined with a regularizer, a parameter that tries to battle overfitting of the data. In real life problems, a scaling parameter can be added in front of the regularization term so that the trade-off can be more easily manipulated.

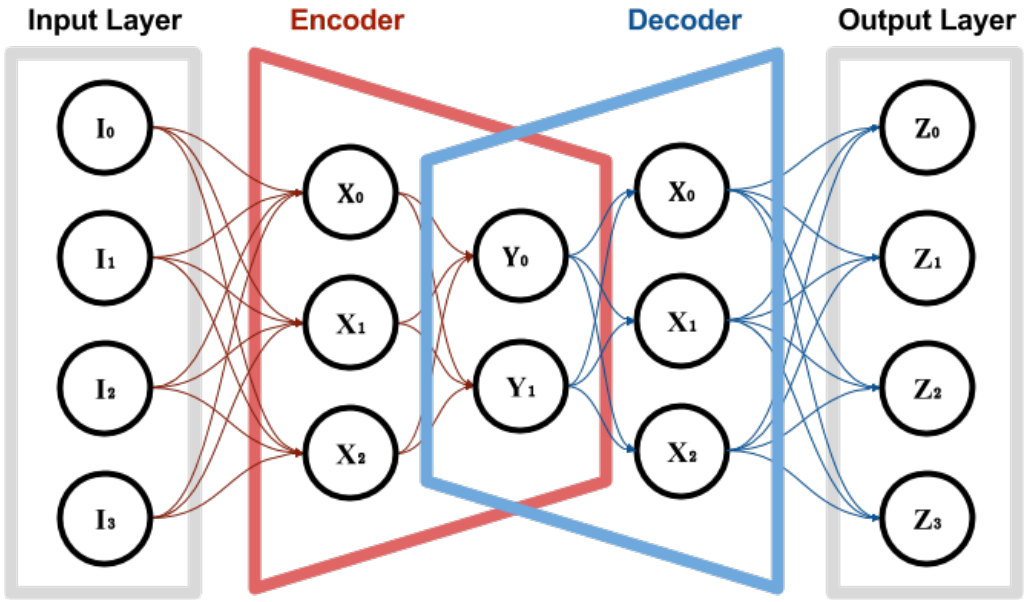


Figure 2.5: Example of an autoencoder with the diferent parts represented. The bottleneck is formed where the encoder intersects the decoder, Y [47].

The most relevant properties of an autoencoder is the specificity of the data, where an autoecoder is only able to compress data similar to the training data, the lossiness of the output compared to the original inputs and the specialization of training instances that will perform better on specific types of input.

- **Undercomplete Autoencoders:** This type of autoencoders rely in a loss or penalizing function that the algorithm tries to minimize. This minimization can be expressed by $L(i, g(f(i)))$ where L penalizes $g(f(i))$ for being dissimilar in relation to i. This way the autoencoder can learn the most salient features. Since it does not have a regularization term, the model needs to have the number of nodes in the hidden layers restricted, so it won't overfit the the training data.
- **Sparse Autoencoders:** This type of autoencoders can provide an information bottleneck without the need to reduce the number of nodes in the hidden layers. For this to happen it takes a training criterion with a sparsity penalty, Ω, on the code layer, h, in addition to the reconstruction error $L(i, g(f(i))) + \Omega(h)$, where $g(h)$ is the decoder output and h the encoder output. This is useful when the objective is to learn features for other tasks like classification. This way the loss function penalizes activations within a layer makes the model become more sensitive to specific attributes of the input data.
- **Denoising Autoencoders:** The idea behind this type of autoencoder is that the representation should be robust to the introduction of noise. For this the input must pass through a function that adds noise to it, that can be a random assignment of a subset of inputs to 0 with an arbitrary probability or can be also a gaussian noise that is a statistical noise having a probability density function equal to that of the normal distribution. Then the output is reconstructed from the corrupted input

and finally the loss function compares the output with the original input without noise.

Autoencoders are very flexible, in the sense that one can introduce nonlinear problems by using a nonlinear activation function. This and the fact that the increase of features will result in a slower processing performance of PCA comparing with autoencoders. Also the dataset does not have to fit into memory, and can be dynamically loaded up and trained with some variant of stochastic gradient descent, which is not the case for Isomap that forces the dataset to exist in memory. The main disadvantage of an autoencoder is the fact that it is extremely uninterpretable, making nearly impossible to a human to visualize and understand the latent features. The different types of autoencoder have each different utilities based on their functioning. The undercomplete autoencoders have a smaller dimension for hidden layer compared to the input layer which helps to obtain important features from the data. The sparse autoencoders have the sparsity constraint that prevents the output to be just a copy of the input, making the model less likely to overfit. The denoising autoencoders ensures that a a good representation can be robustly derived from corrupted or noisy input data and that helps with the task of recovering a clean input that corresponds to the corrupted one. These differences between the functioning of the variations of the autoencoders will be empirically tested.

2.2.6 Last Considerations

The three chosen algorithms have all their advantages and disadvantages in relation to one another that are declared in the last chapters where the algorithms are described. It is important to refer that there is not one that is fully superior to other, making the dataset itself the chooser of the most appropriate algorithm.

Finally to assess the quality of the resulting low-dimensional data representations, one can measure the performance of the algorithms with *trustworthiness* and *continuity* to evaluate to what extent the local structure of the data is retained [48]. The trustwhortiness measures the proportion of points that are close togheter in the low-dimensional space:

$$T(k) = 1 - \frac{2}{nk(2n-3k-1)} \sum_{i=1}^{n} \sum_{j \in U_i^{(k)}} (r(i,j) - k),$$

where $r(i,j)$ is the the rank of the low-dimensional datapoint j according to the pairwise distances between the low-dimensional datapoints. The variable $U_i^{(k)}$ indicates the set of points that are among the k nearest neighbors in the low-dimensional space but not in the high-dimensional space.

The continuity is measured by:

$$C(k) = 1 - \frac{2}{nk(2n-3k-1)} \sum_{i=1}^{n} \sum_{j \in V_i^{(k)}} (\hat{r}(i,j) - k),$$

where $\hat{r}(i, j)$ is the rank of the high-dimensional datapoint j according to the pairwise distances between the high-dimensional datapoints and $V_i^{(k)}$ is the set of points that are among the k nearest neighbors in the high-dimensional space but not in the low-dimensional space.

Besides these techniques to compare the performance of the different reductions there is a need to use classification algorithms used to classify the contacts between the molecules. One of the algorithms is a Naive Bayes Classifier with the addition of a constraint-based method for improving protein docking results, used in another work [49], that utilizes combinations of features for improving protein docking. Another work worth to be cited is HawkRank [50], a scoring function used in the sampling stage of protein–protein docking using energy terms, including van der Waals potentials, electrostatic potentials and desolvation potentials. This function uses weighted potentials from different features and sums everything to the final score. It is also worth mentioning the dataset used for the benchmarking, the ZDOCK benchmark collection, more specifically the fourth version [51]. This benchmark is a collection of distinct protein docking test cases that was used for evaluating HawkRank to other algorithms. These two algorithms are examples of how to test the uselfuness of the extracted features. There are other examples that can be chosen, using a criterion of how simple is to adapt the models to the extracted features.

2.3 Tools

In this section it will be presented the software technology that exists that can be used to build the machine learning algorithms to solve the problem that is presented in this thesis. Since the programming language chosen is Python, due to the sheer amount of packages related to machine learning and also parsing and plotting data, the frameworks considered must allow to work with Python.

2.3.1 Theano

Theano [52]is a Python library that is used to define, optimize, and evaluate mathematical expressions, especially the ones with tensors. Using Theano, it is possible to surpass the language C on a CPU by many orders of magnitude by taking advantage of recent GPUs. The combination of computer algebra system (CAS) with optimizing compilation is particularly useful for tasks in which very complex mathematical expressions are evaluated repeatedly and evaluation speed is of most importance. For situations where many different expressions are each evaluated once, Theano can minimize the amount of compilation/analysis overhead, but still provide symbolic features such as automatic differentiation. The Theano project stopped having new relases after version 1.0.0 in 2017, which can be a deciding factor in the selection of a framework.

2.3.2 Tensorflow

TensorFlow [53] was originally developed by researchers on the Google Brain team within Google's Machine Intelligence Research organization for the purposes of conducting machine learning and deep neural networks research. A computational framework, with a stable Python and C API, for building machine learning models using data flow graphs. The nodes of the graph are mathematical operations, whereas the edges represent tensors that flow between them. One of the key functionalities of Tensorflow is it's flexible architecture enables deployment computation to one or more CPUs or GPUs in a desktop, server, or mobile device without rewriting code. It can be used to lower-level APIs to build models by defining a series of mathematical operations or can be used for higher-level APIs to specify predefined architectures, such as linear regressors or neural networks. These reasons make Tensorflow a good framework to use in this work.

2.3.3 PyTorch

PyTorch [54] is an open source machine learning framework for python developed by Facebook research group. It allows one to flexible experiment and produce in an efficient manner through a hybrid front-end, distributed training and a vast amount of tools and libraries. It takes advantage of native support for asynchronous execution of collective operations and peer-to-peer communication. These reasons makes PyTorch, as it happened with Tensorflow, to be considered to be used in this work. This framework will probably be used if the work cannot run on top of Tensorflow, making this framework a *safety net*.

2.3.4 Keras

Keras [55] is more of an interface rather than a standalone machine-learning framework that was developed with the objective of enabling faster experimentation. It offers a high-level set of abstractions that make it easy to develop deep learning models on top of Tensorflow or Theano. For these reasons this framework will be tested on top of Tensorflow.

2.3.5 Python Libraries

These are language dependent libraries that are going to be used, many of them widely used in the machine learning context.

- **Scikit-Learn:** a Python free to use library with a large number of state-of-the-art machine learning algorithms for supervised and unsupervised problems[56].
- **SciPy:** a free and open-source library built for Python that contains a wide array of tools for optimization, linear algebra, integration, interpolation, special functions, signal and image processingand other tasks common in science and engineering[57].

- **Matplotlib:** a free and open-source plotting Python library which produces a very wide variety of graphs and plots namely like histograms, bar charts, power spectras, error charts between others[58].
- **Pandas:** Pandas, or Python Data Analysis Library, is a free Python library under the BSD license that is useful for data manipulation and analysis. One of the main features of this library is the existence of DataFrame objects that make the data manipulation more accessible to deal with [59].
- **OpenBabel:** a open Python library under the GNU GPL license where the main focus is to search, convert, analyze, or store data from molecular modeling, chemistry, solid-state materials, biochemistry and other related areas. OpenBabel version 2.3 interconverts over 110 formats[60].
- **BioPython:** a Python library under the bioinformatics license that allows the creation of reusable modules and classes and includes parsers for various file formats, access to online services, interfaces to a big number of programs between other features [61].
- **PyMOL** a Python library for visualization of molecular complexes free of use and distributed with a Python license[62].

Chapter 3

Experimental Work

This chapter is composed by the details about the implementation of the whole system used in this thesis including the preprocessing of the data algorithms and presentation of results.

3.1 Implementation

The implementation of the system consists of the following parts:

- **Preprocessing:** Starting with *.pdb* files downloaded from Dockground and features downloaded from the AAIndex, this part will transform to *.csv* files which consist of the data points of the contacts and the neighbours with the corresponding features and labels that will be used by the algorithms.
- **Algorithms:** The algorithms used are an implemented Autoencoder and a PCA for dimensionality reduction and a Naive Bayes classifier to evaluate how much of a improvement the reduced features will be.
- **Presentation:** Finally the graphic representation of the results are also implemented to a more easy human interpretation.

In figure 3.1 it is represented a diagram with the principal components of the implementation.

3.2 Data

This section will discuss the treatment of the data since its origin to the moment that will be used by the algorithms implemented.

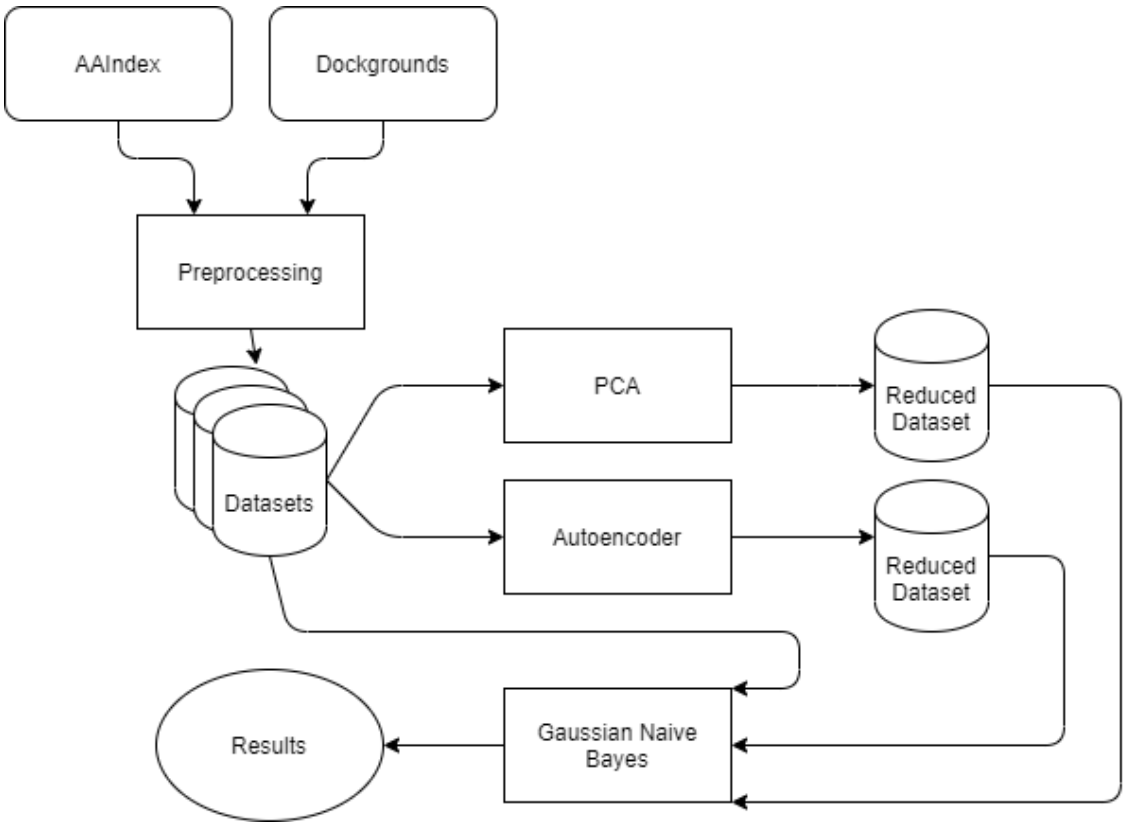


Figure 3.1: Diagram of the principal pipeline of the project

3.2.1 Data Source, Features and Labels

The proteins files are downloaded from the *Dockground* website [21], where each protein file consists of a set composed by two parts: the first part is a *.pdb* file with the first part of the complex and the second part includes a one near native and ninety nine incorrect docking poses for the specific protein-protein complex in which the first model is the correct one. The starting total of complexes is 164 but after eliminating files which originated problems with the parsing the final number of complexes processed decreases to 156. Table 3.1 shows the complete set of complexes that are used in this dissertation.

The files names represent the code id of the complex with four alphanumeric characters. The code of the complex is the same for both the receptor and the ligand, with the only difference being the chain for each one. The receptor has one chain and the other ligand models are all represented by a different chain from the receptor but equal between them.

The features used are extracted from the *AAIndex* database, in which two parts of the database are used, the first and the third that are composed of the amino acid index of 20 numerical values and the statistical protein contact potentials, respectively. In total there are 566 features from the first part and 33 features from the third part, which were analyzed to choose the ones that are relevant.

Complexes
'1a9n', '1blx', '1brs', '1dj7', '1dlf', '1fbv', '1fm0', '1fr2', '1fxw', '1h59', '1i8l', '1iar', '1jat','1jkg', '1k5n', '1ki1', '1krl', '1ksh', '1m9x', '1mbx', '1npe', '1nvp', '1o7n', '1oph', '1oqm', '1pau', '1pvh', '1pxv', '1qav', '1sb2', '1spp', '1sq2', '1stf', '1sv0', '1syx', '1t0p', '1ta3', '1tdq', '1tnr', '1tue', '1uad', '1ugh', '1us7', '1uuz', '1uw4', '1v74', '1wmh', '1wqj', '1xg2', '1yvb', '1zt2', '2a1j', '2a5t', '2aw2', '2bcg', '2bov', '2c35', '2d5r', '2fhz', '2gmi','2grr', '2gwf', '2hrk', '2ido', '2ik8', '2j9u', '2jki', '2npt', '2oxg', '2p7v', '2pqa', '2qby', '2qkl', '2r25', '2rex', '2uy7', '2v5q', '2v8s', '2vdw', '2w2x', '2wbw', '2wd5', '2wjz', '2wmp', '2x9a', '2xg4', '2xxm', '2y9m', '2y9w', '2yho', '2z3q', '2z8v', '2za4', '2zae', '3aa0', '3aev', '3aqb', '3b08', '3byy', '3c7k', '3cip', '3cki', '3d3c', '3dbx', '3dgc', '3dlq', '3e33', '3eo9', '3f6q','3fmo', '3fpn', '3g5y', '3g9a', '3gcg', '3gtu', '3h2u', '3h6s', '3h7h', '3hct', '3iey', '3ijs', '3jv6','3k1i', '3k2m', '3kcp', '3kf6', '3kf8', '3kz1', '3lxr', '3m18', '3mc0', '3mcb', '3mdy', '3n4i', '3o0g', '3o2q', '3oed', '3oq3', '3p9w', '3ph0', '3qb4', '3qbt', '3qdr', '3qhy', '3s97', '3soh', '3sxu', '3tdu', '3tgx', '3u1j', '3u82', '3ulr', '3v96', '3zyj'

Table 3.1: The complexes used in this dissertation extracted from *Dockground*

3.2.2 Data Preprocessing

Since the data necessary to be used in the algorithms originates from various sources there is a need to collect and preprocess all of the information necessary to the creation of the dataset. As stated in [16], almost all contacts are made by the residues in the complex that are in the surface. Using the *BioPython* library, the residues that are present in the surface can be collected by their ASA value and stored for the next steps. The value of ASA used as a threshold to define if a residue is in the surface or interior for the purpose of this dissertation is 30%.

The next step is to find the residues on the surface of both complexes that are close to each other. In [63] it is mentioned that the contacts between residues can be at maximum 8 angstroms. There are going to be tested distances of 2, 4, 6 and 8 to better compare how the distance influences the classifier predictions. The residues neighbours considered are also only in the surface of the complex and as what happens with the distance of contacts. A range of values between 2 and 6 angstroms are used and there are several different distances used to define the neighbourhood of residues of the residue in contact.

Now that the contacts and their respective neighbourhoods are selected, the next step is to find the corresponding features for each of them. The first features that are used are the ones from *AAindex3* and each feature has a corresponding value for each type of contact. There are 33 features that were added to each contact. The second type of features are the ones from *AAindex1* and these have a value associated with each residue. There are 566 existing features, but since many of them are only valid for certain cases, like *Helix termination parameter at position j-2,j-1,j* and repetitions in the indices as result

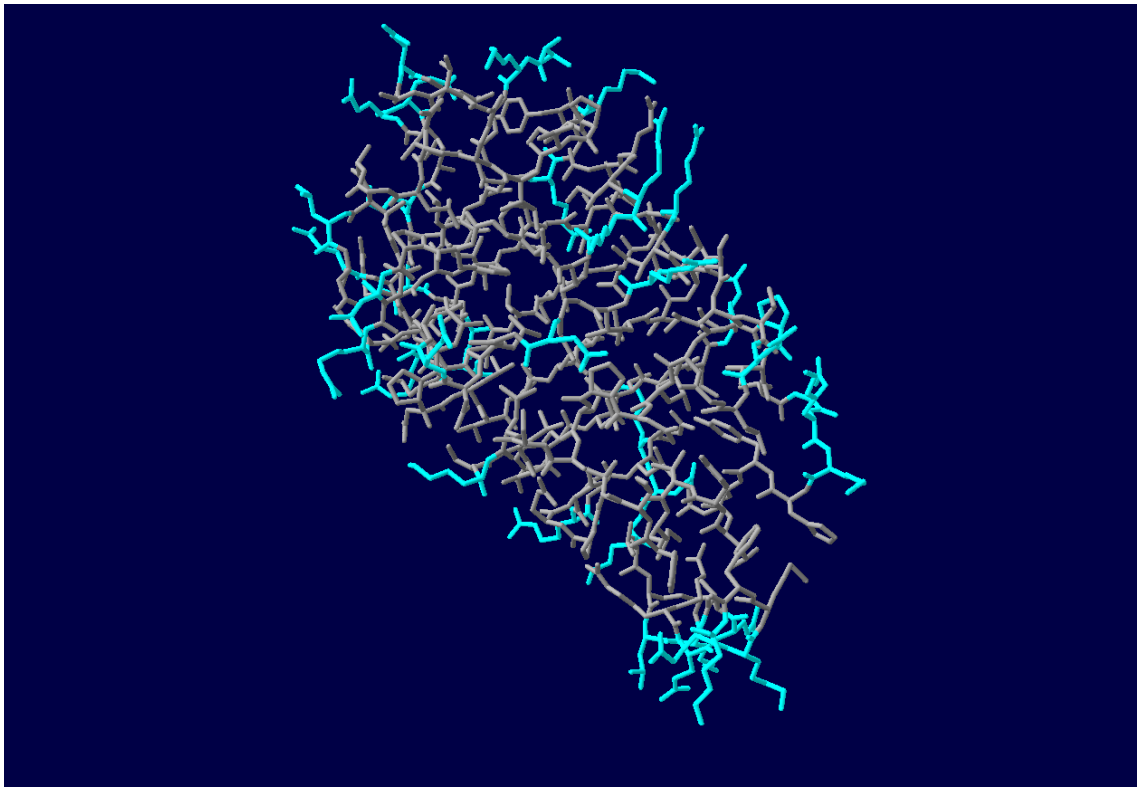


Figure 3.2: Representation of the surface residues of a protein. The light blue residues represent the surface. This image was taken from the A chain from protein 1a9n with the help of the Swiss PDB Viewer.

of different experiments, the number of features utilizes will have to be trim down. After this step the number of features are 97 and each of them are used twice for each contact since each contact is represented by two residues resulting in 194 features for each contact. Additionally the minimum, maximum and mean values of each of the *AAindex1* are used in each neighbourhood of each of the contacts residues performing more 582 features. The last type of features are performed by an algorithm developed that simply returns the number of different types of residues for each complex and each neighbourhood of the contacts. Considering that there are twenty types of amino acids, at least for the purpose of this dissertation, each contact will have eighty additional features for each contact, 20 for each contact residue and 20 to each contact residue neighbourhood. In the end each contact will have a total of 695 distinct features.

For each contact it is added a label designating the name of the protein at which the contact belong. These are useful in the next steps for analyzing the data. These steps result in a *.csv* file with each row representing a contact and the columns representing the features extracted in the previous steps. The labels that serve to distinguish the true contacts from the false will be simply the value of 1 to the true contacts and 0 for the false ones.

3.3 Dataset Splits

To have unbiased results, 10 complexes, chosen at random, were left out of the training of the dimensionality reduction algorithms, and are going to be saved to the classifier test. These complexes we used in the classifier for the test part. This step leaves out 146 complexes. From these 146 complexes 100 are used to train the dimensionality reduction algorithms and after they are trained the 46 complexes that were left out are going to be the test set for these algorithms. After the test set has had is features reduced, these 46 complexes are going to be the training set for the classifier and the 10 complexes that were saved in the beginning are the validation set for the classifier, after having their features reduced as well.

3.4 Algorithms

In this section it will be discussed the machine learning and deep learning algorithms used in this work. There are in each one the principal algorithm and others that are used to create a baseline for comparison.

One matter before the algorithms is the necessity of using scaling methods. Considering the different scales of the features, one should experiment with scaling methods to try to increase the algorithms performance.

The scaling method used in this dissertation, standardization, results in features rescaled so that they have the properties of a normal distribution with the mean equal to zero and the standard deviation equal to one. The formula to calculate this rescalling is:

$$z = \frac{x - \mu}{\sigma}$$

where μ is the mean and σ is the standard deviation.

One important note about rescalling methods is that there are no guaranties that the performance of the algorithms will increase or even decrease in some cases.

3.4.1 Dimensionality Reduction

The first algorithm to be analyzed is the autoencoder. As explained in section 2.2.5, an autoencoder is a type of neural network that tries to copy the input to an output and retrieving the middle layer of nodes that has a reduced number of features. As it happens in most machine learning algorithms, the autoencoders have hyperparameters that have to be tuned, in this case namely:

- Code size
- Number of hidden layers
- Loss function
- Activation function
- Optimizer

- Epochs

Each and everyone of this hyperparameters contribute for the final result of the autoencoder and as such they have to be experimented on to be optimized. The most common way to perform hyperparameter optimization is trough grid search, which is a exhaustive search of a manually specified subset of the hyperparameter space. The subsets of hyperparameters values can be found in **??**.

These subsets of values were chosen after a few tests to check how the algorithm handles the datasets with these hyperparameters. Other hyperparameters were excluded since they did not show significant difference in the final results. The code size after passing the value of 5 did not reduce the value of lossiness between the input and the output in more than 0.01% and started to negatively influence the classifier performance. The number of hidden layers falls off in the same category with values above 6 also not interfering with the final loss results in a meaningful way.

The other hyperparameters, including, the loss function, activation function and optimizer were chosen after consulting the available ones in the Tensorflow framework, and excluding the ones with functionalities more suitable for other types of data. At least two values for each hyperparameter were chosen so it can be possible to compare various baselines for further inspection.

The number of epochs will be stipulated as 50 since in the early tests all the models stopped before that because of one feature of the Tensorflow library that allow to stop the run when the decrease of the loss does not change the output in a significant way, which for the purpose of this dissertation is 0.01%.

Hyperparameter	Subset of Values
Code Size	2, ..., 5
Number of Hidden Layers	2, ..., 6
Loss Function	Mean Squared Error, Poisson
Activation Function	Elu, Exponential
Optimizer	Adadelta, Stochastic Gradient Descent

Table 3.2: Hyperparameter List

To train the autoencoder, only the correct contacts from the 100 complexes mentioned in section 3.3 are used since the purpose of the autoencoder is to extract meaningful features from the data. If the incorrect contacts are used to train the autoencoder there is a possibility that the algorithm will adjust to the wrong data which can decrease the performance of the results which is a less favorable scenario to test in the latter steps.

Each run of the autoencoder is cross validated where the type of cross validation is the k-fold which has a value k of 10, which is a common value for this types of algorithms. This cross validation step helps to choose the best configuration of hyperparameters for the models while trying to reduce the ovefitting of the data. The model is trained and has it performance checked by analyzing the loss with the cross validation set.

The machine used to run all of these test is a Intel Core i7-4790 with 3.60 GHz, 16GB of DDR3 1600MHz Ram and a NVIDIA GForce 980Ti with 6GB of dedicated memory. Using the *CUDA* drivers available for the *NVIDIA* graphic cards, one can use the GPU for increase the time performance of the job in a significant way with the Tensorflow framework.

After training and validating the models with cross-validation, the next step is to reduce the features from the 46 complexes that are used to train the classifier as well the 10 complexes that are used to train and validate the classifier, respectively. Both of them are saved in 2 separate *.csv* files. These results are showed and discussed in chapter 4.

To add another baseline algorithm for comparison with the autoencoder, a PCA algorithm is tested with the datasets. The algorithm was not be implemented but it is going to be imported from the *scikit* library that has a well documented code for easiness of development. As explained in section 2.2.2 the number of components that is adjusted in this algorithm is in function of the percentage of variance that are chosen to be preserved so it can be better compared with the results from the autoencoder and the use of the covarince matrix and correlation matrix.

In practice using the correlation matrix is the same as standardizing the data, so the datasets will also be tested with the data standardized to a more complete test. The chosen variance to be preserved is tested with the values 0.7, 0.9 and 0.95 to achieve a well supported experiment.

As it happens with the autoencoder the training set for the PCA is the 100 complexes for the training which is cross-validated with k-fold, where k is equal to 10. After training, the 46 complexes for the classifier training and the 10 complexes for the classifier validation are reduced. Both of them are saved in 2 separate *.csv* files as it happens in the autoencoder case. The results of this algorithm will also be shown in chapter 4.

3.4.2 Classification

After the step involving the reduction of the datasets dimensionality, there is a need to test the resulting datasets. For this purpose it is used a classifier to analyze if the extracted features are a good representation of the contacts, which are implied if it can correctly classify the correct and incorrect contacts.

The one chosen is the Gaussian Naive Bayes which is used to extrapolate how good of a representation the generated datasets are.

The training set used are the 46 complexes mentioned in section 3.3, and after training the classifier with it, the models that had the best performance are used with the set of 10 complexes that had its features reduced as well is used as the validation set for the classifier. Unlike the training, the separation between the correct contacts and the incorrect ones uses a ratio of around 1:30 to better simulate what happens in real life where there is much more incorrect contacts than correct ones. For each test it is

presented a confusion matrix and the ROC curve graph to better analyze the classifier performance for the different tests using the set of 10 complexes that were left out.

Like the PCA algorithm this classifier will not be implemented but is imported from the *scikit* library. The results will be shown and discussed in chapter 4.

Chapter 4

Results

This section contains the results of the algorithms used in chapter 3. They will be presented the same way that were tested to show the logic behind them. After each test a discussion will also be presented.

4.1 Data Statistics

Before the results of the machine learning algorithms it is valuable to analyze certain statistics of the data utilized to feed them.

Table 4.1 shows the number of contacts by distance and also the number of true and false contacts. A quick analysis shows that the majority of contacts is around 6 and 8 angstroms. The number of false contacts are similar to the positives since for each complex only a near native where the true contacts are localized and a decoy where the false contacts are localized.

Figure 4.1 shows the different numbers of types of residues in the contacts by contact distance for the correct contacts. As it can be observed the percentage of residues by distance does not change in a significant way.

Contact Distance	True Contacts	False Contacts	Total
2	735	779	1514
4	4068	3969	8037
6	5642	5604	11246
8	10685	10656	21341

Table 4.1: Number of contacts

As it can be observed in table 4.1, the number of data points per class stayed in the vicinity of 50% for the correct and incorrect contacts. This is good news since an

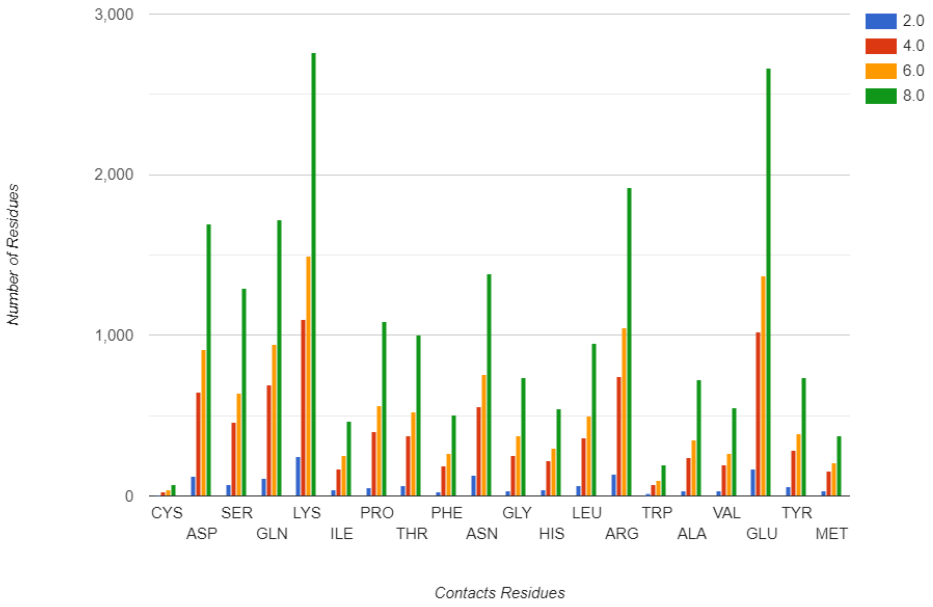


Figure 4.1: Number of different residues performing the contacts by distance of contact

unbalanced dataset can be harmful for the classifier training, leading to more incorrect predictions to the majority class, or the class with more data points.

4.2 Classification

The results of the classifier in relation to the datasets is presented in this section, divided by the datasets of the complexes without reduction, the datasets of the complexes that were reduced by the PCA algorithm and the datasets of the complexes reduced by the implemented autoencoder, for comparison and discussion.

This classification algorithm was used to distinguish between the correct and incorrect residue contacts that happen between residues of different chains.

In this classification part only 56 complexes are used, where the 46 complexes in chapter 3 that were the test set from the autoencoders are used as the training set of the classifier and the 10 complexes that were left out in the beginning of the dataset split are used as validation to select the best autoencoder. The results presented in this chapter, the confusion matrices and the ROC curves, are based on this validation set of 10 complexes.

4.2.1 Without Reduction

To compare the results of the reduced datasets it is a good idea to analyze first how well the classifier deals with the raw data first. A broad search was made first to see

how well the classification goes with the different parameters of contact distance and neighbourhood distance. The first ones to be compared are the datasets with 8 angstroms of distance between contacts. The results of the tests used in this dataset are present in table 4.2, table 4.3 and in figure 4.2.

Dataset	True Positive	True Negative
Predicted Positive	12 / 13	195 / 220
Predicted Negative	38 / 37	468 / 555

Table 4.2: Confusion Matrix results of the rescaling, the first value of each box being the not rescaled dataset and the second, the rescaled dataset

Dataset	Sensitivity	Specificity	Precision	Accuracy	F1 Score
Not Rescaled	0.2400	0.7059	**0.0580**	0.6732	**0.0934**
Rescaled	**0.2600**	**0.7161**	0.0558	**0.6885**	0.0919

Table 4.3: Test results of the rescaling

The results on table 4.2 and table 4.3 show no significant difference between the rescaled and not rescaled features of the different datasets.

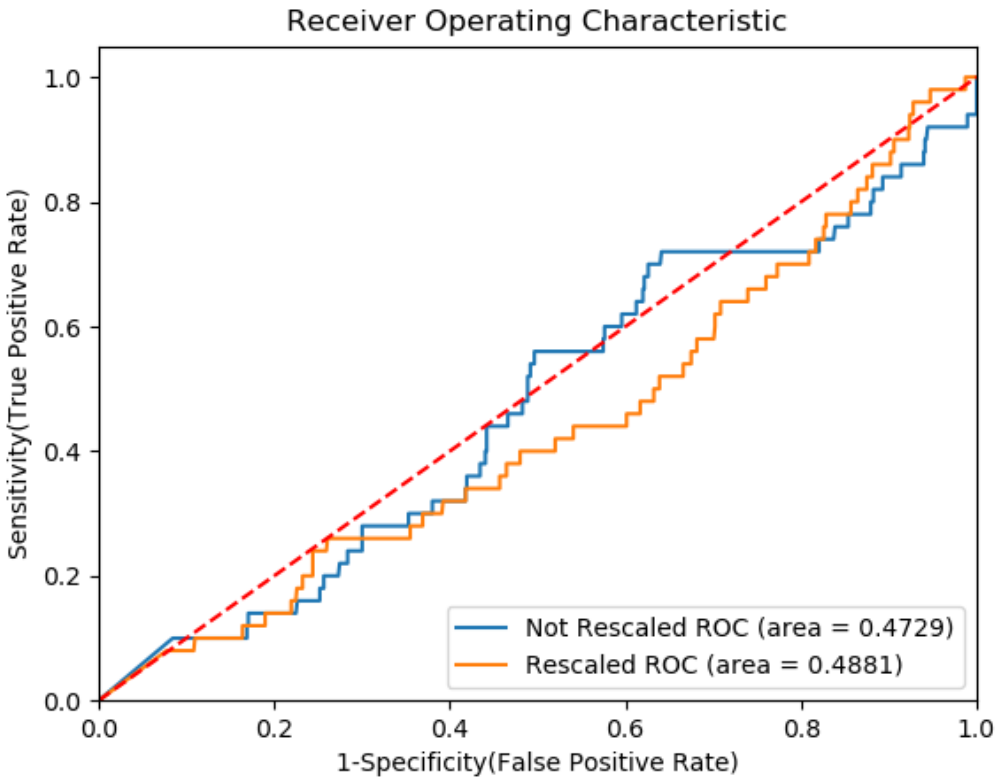


Figure 4.2: Roc curve and Auc score of the rescaling on the datasets without reduction

Both ROC curves showed in figure 4.2 are very close to the 45 degree line which

demonstrates that there is almost no predictive value from both tests. Both tests show that it is better guessing at random the correct contacts than using the dataset without any kind of dimensionality reduction.

The proposed hypothesis is that the classifier is overfitting derived of the large number of features it has. After comparing the train error and the test error from the not rescaled dataset there is reason to believe that this was the case since the metrics for the training were much better than those from the test. The training error metrics can be consulted in table 4.4.

Dataset	Sensitivity	Specificity	Precision	Accuracy	F1 Score
Training metrics	0.5172	0.7320	0.6343	0.6303	0.5698

Table 4.4: Test results of the training metrics of the not rescaled dataset

Next to test was the influence in using the neighbourhood features in the results. To inquiry this, a test with the datasets with 8 angstroms of contact distance and a neighbourhood radius of 2 and 6 angstroms, and another one where the neighbourhood features are not considered. The results can be consulted in table 4.5, table 4.2 and figure 4.3.

Dataset	True Positive	True Negative
Predicted Positive	30 / 43 / 14	399 / 543 / 225
Predicted Negative	20 / 7 / 36	237 / 151 / 744

Table 4.5: Confusion Matrix of the test results, each box has three values which are respectively no neighbourhood, neighbourhood distance of 2 and neighbourhood distance of 6

Dataset	Sensitivity	Specificity	Precision	Accuracy	F1 Score
Neig 0	0.6000	0.3726	0.0699	0.3892	0.1253
Neig 2	**0.8600**	0.2176	**0.0734**	0.2608	**0.1352**
Neig 6	0.2800	**0.7678**	0.0586	**0.7439**	0.0969

Table 4.6: Test results of the neighbourhood

As it can be observed in table 4.5 and table 4.6, the best metrics calculated are not in the dataset without data from the neighbourhood, which may imply the neighbourhood of the contacts are not so important for this. With low neighbourhood distance the classifier can find more correct contacts but as the distance increases it can find less and less. The opposite thing happens with the incorrect contacts.

The ROC curve from figure 4.3 shows that the data about the neighbourhood is included lifts up the classifier performance. The difference between the two distances of the neighbourhood does not seem to make significant difference, but they are going to be tested after the dimensionality reduction step to find if they will do more significant difference after that.

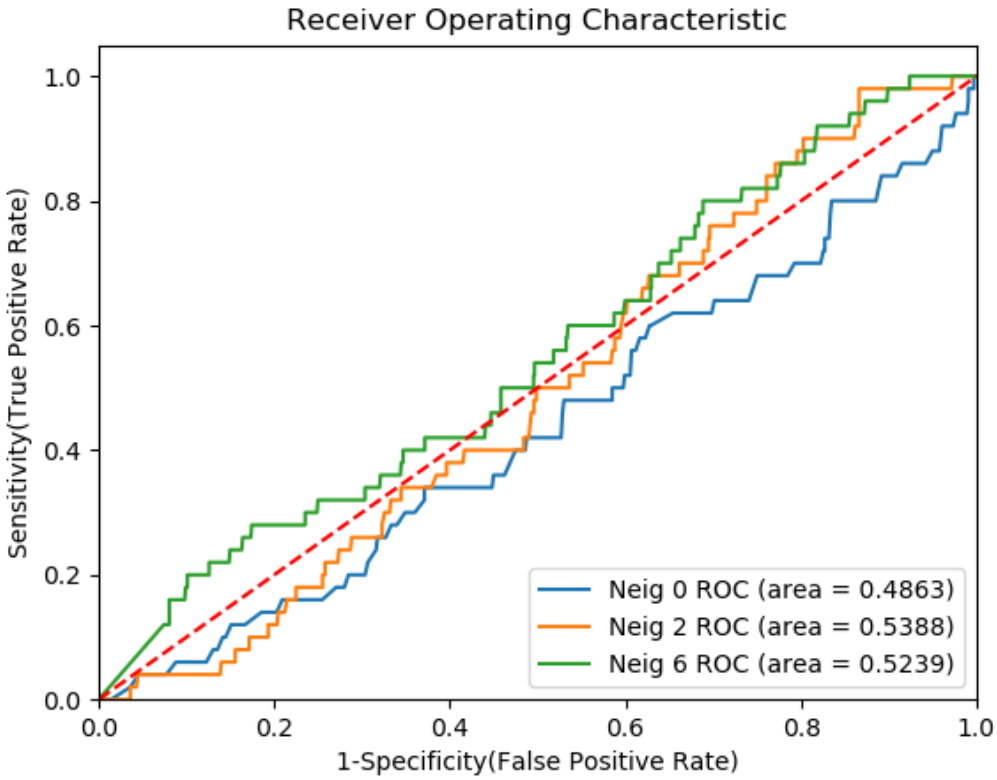


Figure 4.3: Roc curve and Auc score of the neighbourhood distances on the datasets without reduction

The next test made was to see if contacts with fewer distance between the residues may contribute for the predictability abilities of the classifier. This test used 2, 4, 6 and 8 angstroms as distance to test this. Table 4.7 and figure 4.4 shows the results of this test. Since this test has different number of total contacts to each contact distance, the confusion matrix table is not included.

Dataset	Sensitivity	Specificity	Precision	Accuracy	F1 Score
Contact 2	**0.7200**	0.3640	**0.1722**	0.4193	**0.2780**
Contact 4	0.5600	**0.3922**	0.1657	**0.4220**	0.2557
Contact 6	0.7000	0.2370	0.1452	0.3094	0.2405
Contact 8	0.6400	0.2587	0.0448	0.2784	0.0838

Table 4.7: Test results of the contact distance

Table 4.7 shows that increasing the contact distance considered makes the classifier performance go down. This implies that it is more difficult to judge contacts with large distance than it is to judge contacts with smaller distances.

Figure 4.4 confirms the results from table 4.7, showing that the contacts from bigger distances are more difficult to predict than the others.

In a general way the results of the classification of these datasets were not the most

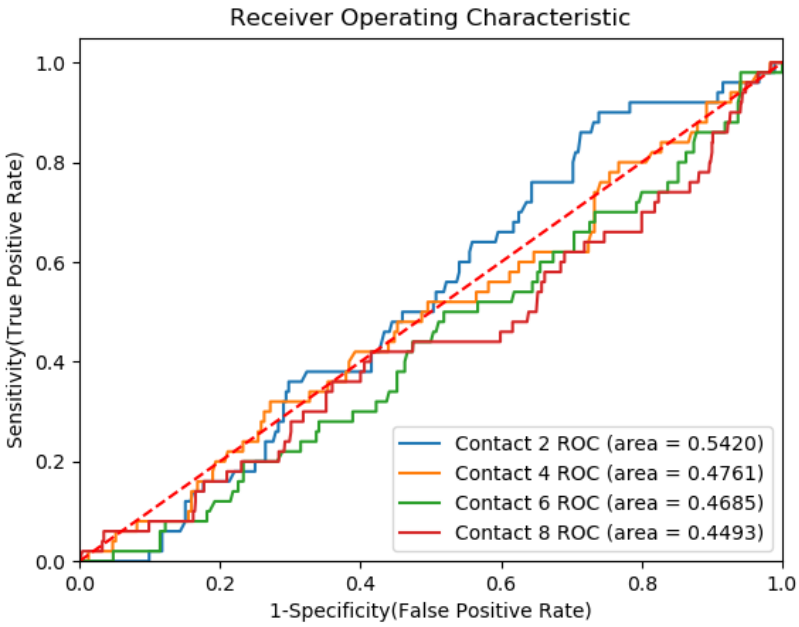


Figure 4.4: Roc curve and Auc score of the contact distances on the datasets without reduction

promising, but that was to be expected since so many features are involved which can influence the results of the classifier.

Having analyzed the results without reduction, they will be tested against the results from the dimensionality reduction algorithms to see how much of a improvement they can have on the datasets results.

4.2.2 PCA Reduction

The first results for constructing a baseline to compare to the autoencoder implemented are the ones from the PCA algorithm. First the results and comparisons between the tests of this algorithms are going to be shown and discussed and after that a comparison and discussion between these results and the results from the datasets without reduction.

The first test is to see if the difference between the distance of the contacts would affect the classification. This test used a distance of 2, 4, 6 and 8 angstroms to better evaluate the predictability power of the classifier. The results of this test are in table 4.8 and figure 4.5. Since this test has different number of total contacts to each contact distance, the confusion matrix table is not included.

Table 4.8 shows that the more distance that exists between contacts, in similarity with the results from the datasets without reduction in table 4.7, the lesser the performance of the classifier. Although the number of correct contacts does increase as the distance increases as well. Figure 4.5 supports the ROC curves and the AUC values from table 4.8

Dataset	Sensitivity	Specificity	Precision	Accuracy	F1 Score
Contact 2	0.6000	**0.5422**	**0.2083**	**0.5518**	**0.3093**
Contact 4	0.6400	0.3285	0.1468	0.3761	0.2388
Contact 6	0.7000	0.2702	0.1080	0.3184	0.1872
Contact 8	**0.8000**	0.2382	0.0738	0.2779	0.1351

Table 4.8: Test results of the contact distance

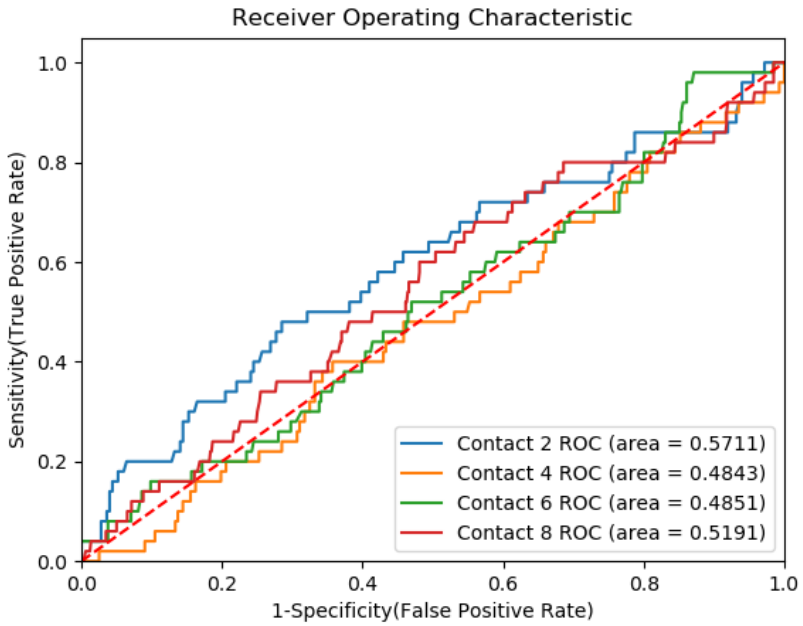


Figure 4.5: Roc curve and Auc score of the contact distances on the datasets reduced from PCA

The second test is to check the neighbourhood influence the results in any manner. To inquiry this, a test with the datasets with 8 angstroms of contact distance and a neighbourhood radius of 2 and 6 angstroms, and another one where the neighbourhood features are not considered, are presented in table 4.10, table 4.9 and figure 4.6.

Dataset	True Positive	True Negative
Predicted Positive	26 / 39 / 21	262 / 436 / 257
Predicted Negative	24 / 11 / 29	422 / 184 / 307

Table 4.9: Confusion Matrix of the test results. Each box has three values representing a neighbourhood distance of 0, 2 and 6 angstroms

As noticed in table 4.9 and table 4.10, the use of no data belonging to the neighbourhood is beneficial to the classification capabilities of the classifier where, as the results in table 4.6, the only metric that is increasing as the neighbourhood distance increases is the sensitivity.

Figure 4.6 shows similar results, where the ROC curve and AUC values have a

Dataset	Sensitivity	Specificity	Precision	Accuracy	F1 Score
Neig 0	0.5200	**0.6170**	**0.0903**	**0.6104**	**0.1538**
Neig 2	**0.7800**	0.2958	0.0818	0.3318	0.1480
Neig 6	0.4200	0.5443	0.0755	0.5342	0.1280

Table 4.10: Test results of the neighbourhood

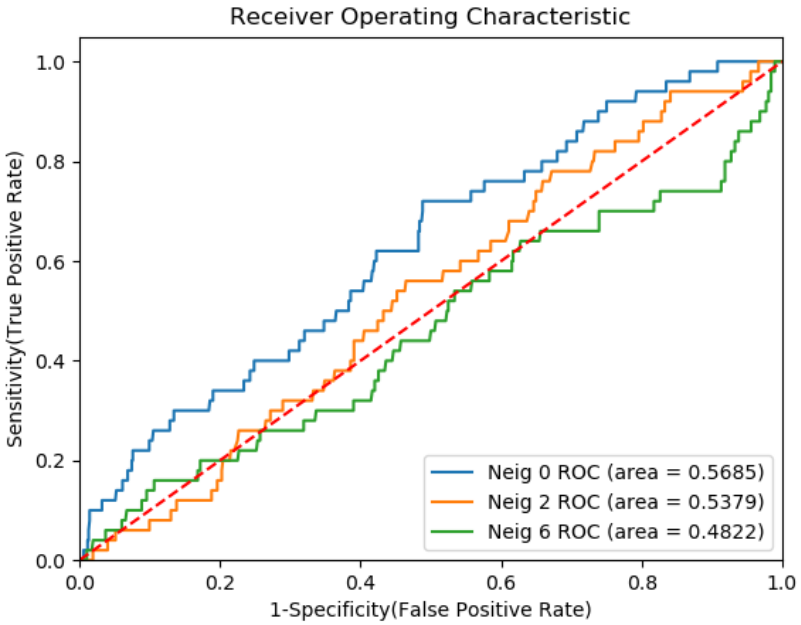


Figure 4.6: Roc curve and Auc score of the neighbourhood distances on the datasets reduced from PCA

decrease of value as the distance increases.

The third test is to review the influence of the variance retained by the PCA algorithm. To test this, a dataset with 8 angstroms of contact distance and 6 angstroms of neighbourhood distance will be tested with a preserved variance of 70%, 90% and 95%. The results of this test can be consulted in table 4.11, table 4.12 and figure 4.7.

Dataset	True Positive	True Negative
Predicted Positive	31 / 23 / 18	271 / 275 / 390
Predicted Negative	19 / 27 / 32	488 / 408 / 472

Table 4.11: Confusion Matrix of the test results. Each box has three values representing the PCA preserved variance of 0.7, 0.9 and 0.95

After reviewing the results in table 4.11 and table 4.12 it can be concluded that there is a increase in the results as the preserved variance decreases. Below 70% of preserved variance the values of the metrics begin to decrease as well.

Figure 4.7 shows a decrease of the AUC values as the preserved variance increases, as it happens with the results of table 4.12.

Dataset	Sensitivity	Specificity	Precision	Accuracy	F1 Score
0.7	**0.6200**	**0.6430**	**0.1026**	**0.6415**	**0.1761**
0.9	0.4600	0.5974	0.0772	0.5880	0.1322
0.95	0.3600	0.5476	0.0441	0.5373	0.0786

Table 4.12: Test results of the preserved variance

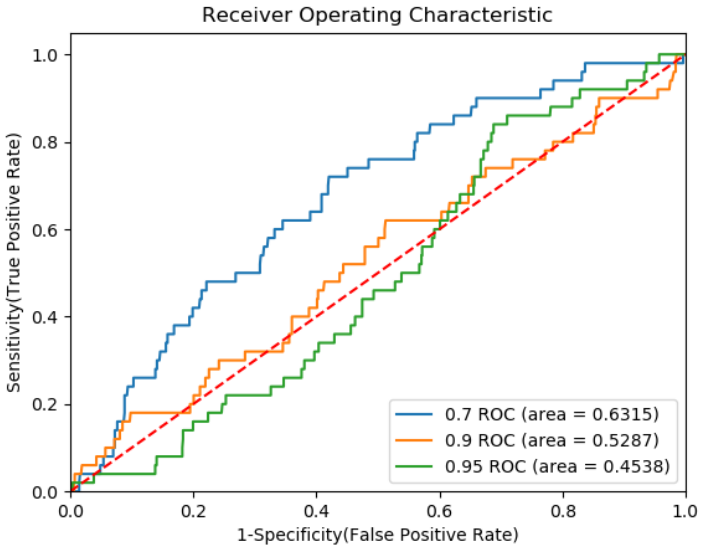


Figure 4.7: Roc curve and Auc score of the PCA preserved variance on the datasets reduced from PCA

Now that the comparisons between the reduced datasets are done, the last test is to compare the results of these datasets with the results from the datasets without reduction. To compare both, a test with contact distance of 8 angstroms and neighbourhood radius of 6 angstroms from both are presented in table 4.13, table 4.14 and figure 4.8.

Dataset	True Positive	True Negative
Predicted Positive	9 / 19	107 / 174
Predicted Negative	41 / 31	483 / 430

Table 4.13: Confusion Matrix of the test results. Each box has two values representing the dataset without reduction and the dataset from PCA, respectively

Dataset	Sensitivity	Specificity	Precision	Accuracy	F1 Score
Without Reduction	0.1800	**0.8186**	0.0776	**0.7688**	0.1084
PCA	**0.3800**	0.7119	**0.0984**	0.6865	**0.1564**

Table 4.14: Test results of the comparison between PCA and datasets without reduction

As it can be observed in table 4.13 and table 4.14 the results of applying PCA to the datasets have a positive effect in the classifier performance. The only downside is the

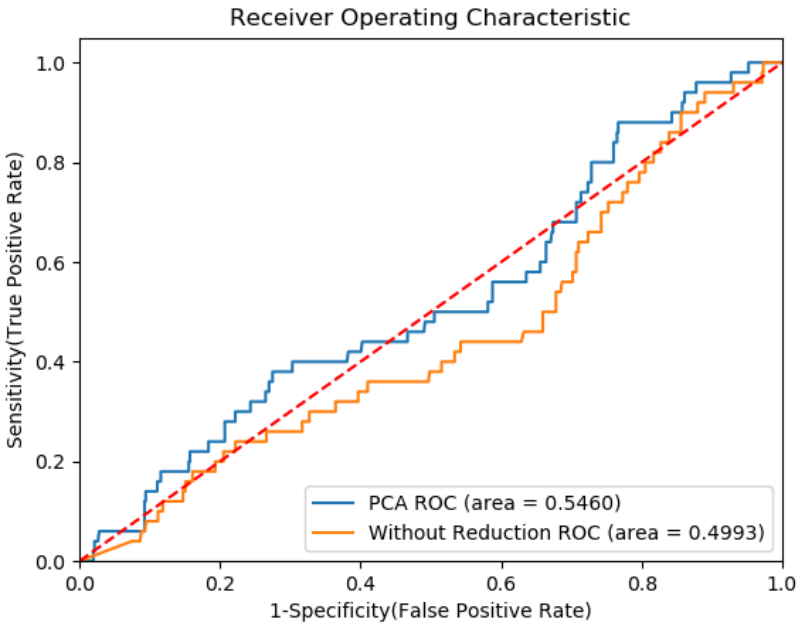


Figure 4.8: Roc curve and Auc score of the PCA dataset vs the dataset without reduction

specificity and the accuracy that decrease by around 10% each.

4.2.3 Autoencoder Reduction

With baselines, reduced and not reduced, already processed and analyzed, it is time to compare them to the implemented algorithm and the focus of this dissertation. First it is necessary to compare the models to observe how the different parameters and hyperparameters influence the results of the classifier. The initial test will be to verify if the distance between contacts are a factor to the predictability of the classifier. To assess that, a test containing datasets with distances of contact of 2, 4, 6 and 8 angstroms and a neighbourhood radius of 2 angstroms was made with the results present in table 4.15 and figure 4.9. Since this test has different number of total contacts to each contact distance, the confusion matrix table is not included.

Dataset	Sensitivity	Specificity	Precision	Accuracy	F1 Score
Contact 2	0.2400	0.7911	0.0759	0.7543	0.1154
Contact 4	0.1800	0.7783	0.0625	0.7329	0.0928
Contact 6	**0.3600**	0.6179	**0.0933**	0.5925	**0.1481**
Contact 8	0.2000	**0.8208**	0.0855	**0.7728**	0.1198

Table 4.15: Test results of the contact distance

As it can be observed in table 4.15, there is a slight increase of the classifier performance as more distant contacts are being considered. Figure 4.9 shows also this increase,

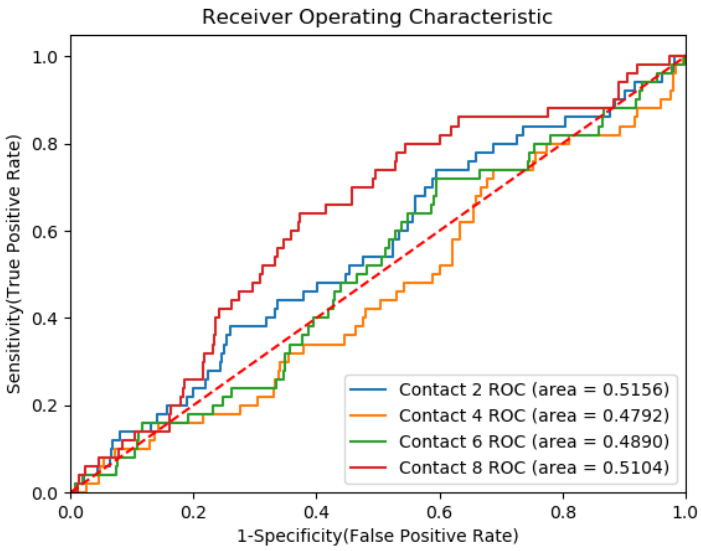


Figure 4.9: Roc curve and Auc score of the contact distance datasets on the datasets reduced from the autoencoder

except for the contacts at 2 angstroms of distance which are slightly better than the 4 and 6 angstroms contact distance datasets.

The next test was to evaluate how the neighbourhood features influenced the results. A test comprising of the datasets with contact distance of 8 angstroms and neighbourhood radius of 2 and 6 angstroms and a dataset without neighbourhood features was tested and the results can be observed in table 4.16, table 4.17 and figure 4.10.

Dataset	True Positive	True Negative
Predicted Positive	9 / 33 / 13	127 / 472 / 153
Predicted Negative	41 / 17 / 37	428 / 283 / 443

Table 4.16: Confusion Matrix of the test results. Each box has three values representing a neighbourhood distance of 0, 2 and 6 angstroms, respectively

Dataset	Sensitivity	Specificity	Precision	Accuracy	F1 Score
Neig 0	0.1800	**0.7712**	0.0662	**0.7223**	0.0968
Neig 2	**0.6600**	0.3748	0.0653	0.3925	0.1189
Neig 6	0.2600	0.7433	**0.0783**	0.7059	**0.1204**

Table 4.17: Test results of the neighbourhood

After analyzing table 4.17 the use of no data from the neighbourhood increases in about 2 to 3% the specificity and accuracy metrics when compared to the dataset that uses data from the 6 angstroms neighbourhood radius. The dataset with 2 angstroms of neighbourhood radius falls short on these metrics but has more sensitivity in relation to the other two datasets.

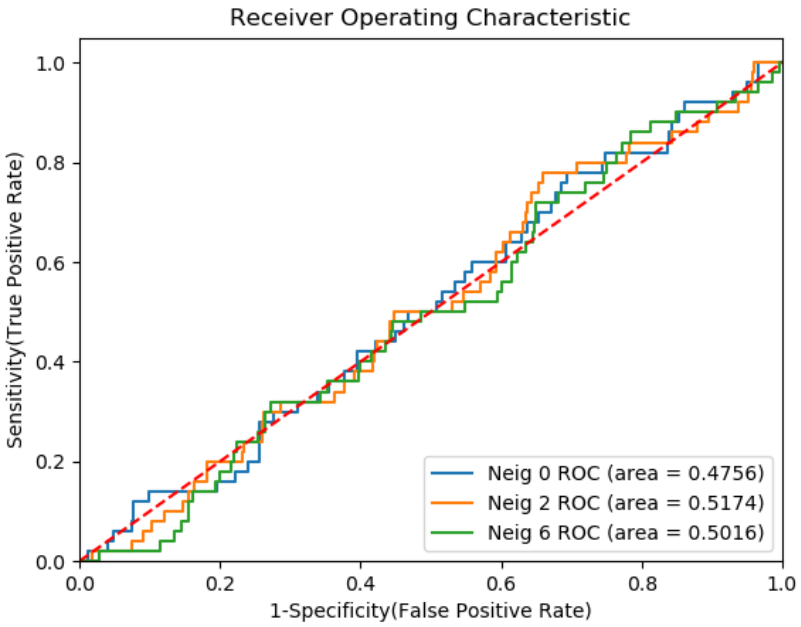


Figure 4.10: Roc curve and Auc score of the neighbourhood distance datasets on the datasets reduced from the autoencoder

The next test will be against the number of features, or code size, that the autoencoder has. To test this the dataset with 8 angstroms of contact distance and 6 angstroms of neighbourhood radius is going to be reduced to 2, 3, 4 and 5 features to evaluate how much of the results of the classifier will vary. Table 4.18, table 4.19 and figure 4.11 shows the results of this test.

Dataset	True Positive	True Negative
Predicted Positive	18 / 15 / 9 / 9	147 / 114 / 68 / 131
Predicted Negative	32 / 35 / 41 / 41	502 / 846 / 623 / 730

Table 4.18: Confusion Matrix of the test results. Each box has four values representing 2, 3, 4, and 5 features dataset

Dataset	Sensitivity	Specificity	Precision	Accuracy	F1 Score
2 features	**0.3600**	0.7735	0.1091	0.7439	0.1674
3 features	0.3000	0.8813	0.1163	0.8525	**0.1676**
4 features	0.1800	**0.9016**	**0.1169**	**0.8529**	0.1417
5 features	0.1800	0.8479	0.0643	0.8112	0.0947

Table 4.19: Test results of the number of features extracted

As it is shown in table 4.19, there is an increase of the metrics shared between the the datasets with three and four features. As the other datasets number of features increases the metrics decrease in a general manner. Figure 4.11 supports those mentioned metrics

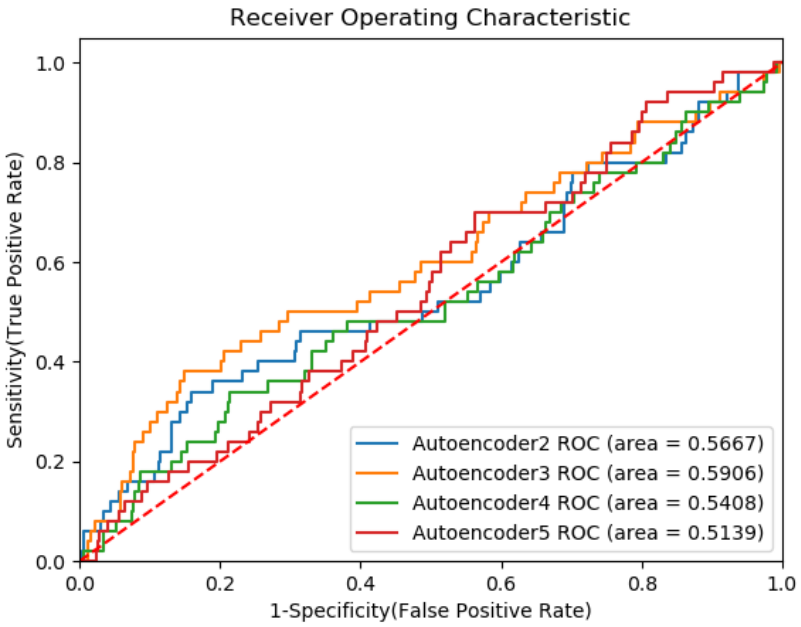


Figure 4.11: Roc curve and Auc score of the number of features datasets on the datasets reduced from the autoencoder

with the roc curve and auc values being most elevated at the three and four features datasets.

Now that the results from the autoencoder different parameters were experimented on, the comparison between the different baselines of the non reduced datasets and the PCA reduced datasets is the next to be analyzed. The parameters used to best analyze the different baselines were the 8 angstroms contact distance and the 6 angstroms neighbourhood distance from each of the three baselines. The compared result can be observed in table 4.20, table 4.21 and figure 4.12.

Dataset	True Positive	True Negative
Predicted Positive	8 / 17 / 19	185 / 280 / 122
Predicted Negative	42 / 33 / 31	392 / 536 / 447

Table 4.20: Confusion Matrix of the test results. Each box has three values representing the without reduction dataset, PCA dataset and Autoencoder dataset respectively

Dataset	Sensitivity	Specificity	Precision	Accuracy	F1 Score
Without Reduction	0.1600	0.6794	0.0415	0.6380	0.0658
PCA	0.3400	0.6569	0.0572	0.6386	0.0980
Autoencoder	**0.3800**	**0.7856**	**0.1348**	**0.7528**	**0.1990**

Table 4.21: Test results of the comparison between PCA, autoencoder and datasets without reduction

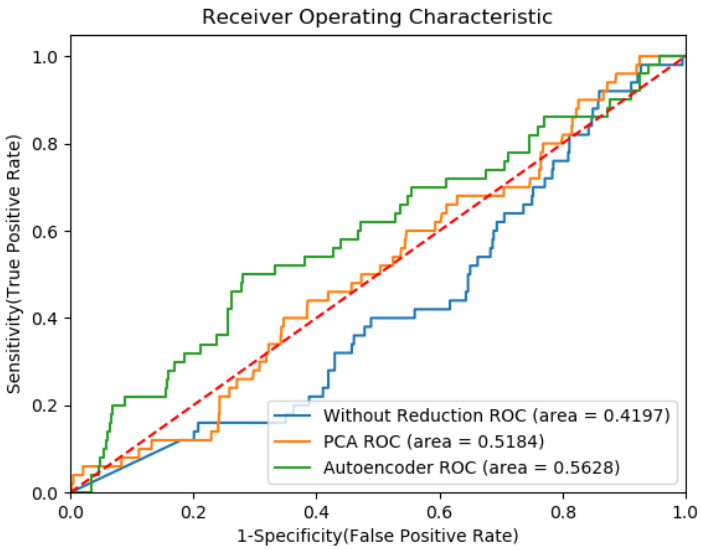


Figure 4.12: Roc curve and Auc score of the datasets without reduction, PCA and autoencoder

As noticed in table 4.20, table 4.21 and figure 4.12, the results show that the datasets reduced achieved better results than the dataset without reduction. The best of the dimensionality reduction algorithms was the autoencoder which had the best metrics all around and highest value of AUC value. It is probably related with PCA having the restriction of only having linear transformations that can be applied and the autoencoders having the ability to use non-linear activation functions, which in some cases can better grasp a better representation of the data.

Chapter 5

Conclusion and Future Work

5.1 Conclusion and Limitations

With the results presented and discussed, the only thing that is left is to make conclusions about them and the work done about this dissertation.

First and foremost, the results from the test of the autoencoder compared to the PCA and the datasets that had not its features reduced are promising. Autoencoders can learn data projections that are more interesting than PCA or other basic techniques. It also provides a more accurate output when compared to PCA and the not reduced datasets. This is useful information since this area of proteins interactions is still a hot topic in the current literature. Autoencoders can help the investigation of these contacts that participate in these interactions and help perceiving at what level these interactions occur and what are the most contributing factors involved.

Now it is time to address the limitations. The first limitation is the results of the number of contacts extracted from the datasets used which, using the maximum contact distance of 8 angstroms amounts to a total of 21341 contacts including the correct and false ones. This is an inevitable limitation on every machine learning problem where it is impossible to precisely estimate the minimum amount of data required for this types of project. This could had an impact in the results where the dataset does not show enough variance to construct a very high performance model.

The principal limitation against the results of this dissertation is the method on which the true contacts were picked, since it was an implemented method based only on distance of the residues who make them, which, is a simplification of how true contacts are determined. Not all the contacts chosen as correct were actually correct and it may have influence the results for the classifier which contributes with human bias since the labels asserted as correct could not be actually correct. This limitation does not influence

the false contacts though since all contacts extracted from the decoy models were actually false.

Nevertheless, the method at which they were picked are approximated with the methods at which they are considered in the literature, differing in some subtleties, one example being certain maximum contact distance being 7 angstroms for residues with some restrictions. The lack of tests with the real contacts from different datasets which have true contacts in them is a faulty aspect of this dissertation that was not concluded in time for the delivery.

Another aspect that this dissertation does not address is which features are the most important for the formation of contacts between the residues. Since the autoencoder is a black box algorithm in which the lack of transparency difficult the human interpretability, it is difficult to evaluate why and how the different features were used. With these problems analyzed and discussed, the results of the autoencoder, while being certainly a bit too optimistic, are very promising in these types of problems since they can adjust well to different type and size of data.

Finally in the state of the art there are more dimensionality reduction algorithms that those that were used in the experimental work. The Isomap, t-SNE, MDS and the LTSA algorithms were not considered mostly because of lack of time which makes the testing under complete.

The author of this dissertation comes from a purely computer science background, which implies a lack of knowledge in certain aspects of the biochemistry part of this dissertation. Thus, the complete implementation code will be available online for easiness of use and review of the community where are experts that can further take the results of this dissertation.

5.2 Future Work

One of the the limitation presented in the conclusion was the lack of test using true contacts properly tested with the literature manners. For this, there is a need to implement algorithms, which, fed by the *.pdb* files, could pick more accurately the true contacts from the complexes.

The next work that could be done is the choice of features that were used for this project, which include the ones used and the ones left out from the AAIndex. Furthermore, features from other sources could also be used to better enrich the datasets.

Since the results from chapter 4 are from the validation set, an improve to this work is to make a better use of the cross-validation to achieve a better representation of the contacts and after that creating a test set to evaluate the improvment of that representation which was not possible in this dissertation.

Also, more algorithms for feature extraction mentioned in chapter 2, which have non linear transformations, should be implemented to create more baselines for comparison with the implemented autoencoder to understand better the latent information about

the contacts.

The last limitation is the lack of tests with real world benchmarks to determine how much of a improvement or not the extracted features from the autoencoder are comparing with those benchmarks.

Bibliography

[1] F. Haurowitz and D. E.Koshland. *Protein*. 2018. URL: https://www.britannica.com/science/protein.

[2] J. Berg, J. Tymoczko, and L Stryer. *Biochemistry*. 5th ed. W H Freeman, 2002.

[3] *Amino Acids*. http://www.nutrientsreview.com/proteins/amino-acids. Accessed: 2019-01-30.

[4] E. Levy. "A Simple Definition of Structural Regions in Proteins and Its Use in Analyzing Interface Evolution." In: *Journal of Molecular Biology* (2010).

[5] A. Bogan and K. Thorn. "Anatomy of Hot Spots in Protein Interfaces." In: *Journal of Molecular Biology* (1998).

[6] I. Moreira, P. Fernando, and M. Ramos. "Hot-spots - A review of the protein-protein interface determinant amino-acid residues." In: *Wiley InterScience - Proteins* (2002).

[7] C. Chothia. "The nature of the accessible and buried surfaces in proteins." In: *Journal of Molecular Biology* 105.1 (1976), pp. 1 –12. ISSN: 0022-2836. DOI: https://doi.org/10.1016/0022-2836(76)90191-1. URL: http://www.sciencedirect.com/science/article/pii/0022283676901911.

[8] L Young, R Jernigan, and D Covell. "A role for surface hydrophobicity in protein-protein recognition." In: *Protein Science* (1994).

[9] A. F. Brito and J. W. Pinney. "Protein–Protein Interactions in Virus–Host Systems." In: *Frontiers in Microbiology* 8 (2017), p. 1557. ISSN: 1664-302X. DOI: 10.3389/fmicb.2017.01557. URL: https://www.frontiersin.org/article/10.3389/fmicb.2017.01557.

[10] P. Chakrabarti and J. Janin. "Dissecting Protein-Protein Recognition Sites." In: *Proteins: Structure, Function and Genetics* (2002).

[11] J. Morrow and S. Zhang. "Computational Prediction of Hot Spot Residues." In: *National Institute of Health Public Access* (2013).

[12] Y. Ofran and B. Rost. "Analysing Six Types of Protein–Protein Interfaces." In: *Journal of Molecular Biology* (2003).

[13] T Clackson and J. Wells. "A hot spot of binding energy in a hormone-receptor interface." In: *Science* (1995).

[14] L. H. Hartwell, J. J. Hopfield, S. Leibler, and A. W. Murray. "From molecular to modular cell biology." In: *Nature* 402 (1999), C47 EP –. URL: https://doi.org/10.1038/35011540.

[15] J. Drews. "Drug Discovery: A Historical Perspective." In: *Science* 287.5460 (2000), pp. 1960–1964. ISSN: 0036-8075. DOI: 10.1126/science.287.5460.1960. eprint: http://science.sciencemag.org/content/287/5460/1960.full.pdf. URL: http://science.sciencemag.org/content/287/5460/1960.

[16] C. Yan, F. Wu, R. Jernigan, D. Dobbs, and V. Honavar. "Characterization of Protein-Protein Interfaces." In: *The protein journal* (2008).

[17] K.-i. Cho, D. Kim, and D. Lee. "A feature-based approach to modeling protein-protein interaction hot spots." In: *Nucleic Acids Research* (2009).

[18] W. DeLano. "Unraveling hot spots in binding interfaces: progress and challenges." In: *Current Opinion in Structural Biology* (2002).

[19] A. Krogh, M. Brown, S. Mian, K. Sjolander, and D. Haussler. "Hidden Markov Models in Computational Biology: Applications to Protein Modeling." In: *Journal of Molecular Biology* (1994).

[20] H. M. Berman, G. J. Kleywegt, H. Nakamura, and J. L. Markley. "The Protein Data Bank at 40: Reflecting on the Past to Prepare for the Future." In: *Structure* 20.3 (2012), pp. 391 –396. ISSN: 0969-2126. DOI: https://doi.org/10.1016/j.str.2012.01.010. URL: http://www.sciencedirect.com/science/article/pii/S0969212612000184.

[21] K. PJ, A. I, D. T, K. I, M. D, C. MM, and V. IA. "Dockground: A comprehensive data resource for modeling of protein complexes." In: *Protein Science* (2018).

[22] A. Kolinski, M. Pokarowska, M. Kanehisa, P. Pokarowski, T. Katayama, and S. Kawashima. "AAindex: amino acid index database, progress report 2008." In: *Nucleic Acids Research* 36.suppl$_1$ (2007), pp. D202–D205. ISSN: 0305-1048. DOI: 10.1093/nar/gkm998. eprint: http://oup.prod.sis.lan/nar/article-pdf/36/suppl_1/D202/7637341/gkm998.pdf. URL: https://dx.doi.org/10.1093/nar/gkm998.

[23] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. http://www.deeplearningbook.org. MIT Press, 2016.

[24] D. H. Wolpert and W. G. Macready. "No free lunch theorems for optimization." In: *IEEE transactions on evolutionary computation* 1.1 (1997), pp. 67–82.

[25] Y. Ho and D. Pepyne. "Simple Explanation of the No-Free-Lunch Theorem and Its Implications." In: *Journal of Optimization Theory and Applications* 115.3 (2002), pp. 549–570. DOI: 10.1023/A:1021251113462. URL: https://doi.org/10.1023/A:1021251113462.

[26] R. H. Wilcox. "Adaptive control processes—A guided tour, by Richard Bellman, Princeton University Press, Princeton, New Jersey, 1961, 255 pp., $6.50." In: *Naval Research Logistics Quarterly* 8.3 (1961), pp. 315–316. DOI: 10.1002/nav.3800080314. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/nav.3800080314. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/nav.3800080314.

[27] *Dimensionality Reduction Techniques*. http://www.turingfinance.com/artificial-intelligence-and-statistics-principal-component-analysis-and-self-organizing-maps/. Accessed: 2019-01-30.

[28] N. Wiener and I. Extrapolation. *Smoothing of stationary time series*. 1949.

[29] B. M. Shahshahani and D. A. Landgrebe. "The effect of unlabeled samples in reducing the small sample size problem and mitigating the Hughes phenomenon." In: *IEEE Transactions on Geoscience and remote sensing* 32.5 (1994), pp. 1087–1095.

[30] M. Verleysen and D. François. "The curse of dimensionality in data mining and time series prediction." In: *International Work-Conference on Artificial Neural Networks*. Springer. 2005, pp. 758–770.

[31] S. T. Roweis and L. K. Saul. "Nonlinear dimensionality reduction by locally linear embedding." In: *science* 290.5500 (2000), pp. 2323–2326.

[32] I. Guyon and A. Elisseeff. "An introduction to variable and feature selection." In: *Journal of machine learning research* 3.Mar (2003), pp. 1157–1182.

[33] H. Liu and H. Motoda. *Feature extraction, construction and selection: A data mining perspective*. Vol. 453. Springer Science & Business Media, 1998.

[34] L. Cayton. "Algorithms for manifold learning." In: *Univ. of California at San Diego Tech. Rep* 12.1-17 (2005), p. 1.

[35] H. Narayanan and S. Mitter. "Sample complexity of testing the manifold hypothesis." In: *Advances in Neural Information Processing Systems*. 2010, pp. 1786–1794.

[36] M. W. Davis. "Groups generated by reflections and aspherical manifolds not covered by Euclidean space." In: *Annals of Mathematics* (1983), pp. 293–324.

[37] *2-manifold*. http://www.map.mpim-bonn.mpg.de/2-manifolds. Accessed: 2019-01-30.

[38] L. Cayton. "Algorithms for manifold learning." In: 2005.

[39] J. Shlens. "A Tutorial on Principal Component Analysis." In: *CoRR* abs/1404.1100 (2014). arXiv: 1404.1100. URL: http://arxiv.org/abs/1404.1100.

[40] C Croux and G Haesbroeck. "Principal component analysis based on robust estimators of the covariance or correlation matrix: influence functions and efficiencies." In: *Biometrika* 87.3 (Sept. 2000), pp. 603–618. ISSN: 0006-3444. DOI: 10.1093/biomet/87.3.603. eprint: http://oup.prod.sis.lan/biomet/article-pdf/87/3/603/830678/870603.pdf. URL: https://dx.doi.org/10.1093/biomet/87.3.603.

[41] *Principal Component Analisys*. https://galaxydatatech.com/2018/07/12/principal-component-analysis/. Accessed: 2019-01-30.

[42] J. B. Tenenbaum, V. De Silva, and J. C. Langford. "A global geometric framework for nonlinear dimensionality reduction." In: *science* 290.5500 (2000), pp. 2319–2323.

[43] *Dimension Reduction - IsoMap*. https://blog.paperspace.com/dimension-reduction-with-isomap/. Accessed: 2019-01-30.

[44] L. v. d. Maaten and G. Hinton. "Visualizing data using t-SNE." In: *Journal of machine learning research* 9.Nov (2008), pp. 2579–2605.

[45] J. B. Kruskal. "Nonmetric multidimensional scaling: A numerical method." In: *Psychometrika* 29.2 (1964), pp. 115–129. ISSN: 1860-0980. DOI: 10.1007/BF02289694. URL: https://doi.org/10.1007/BF02289694.

[46] Z. Zhang and H. Zha. "Principal Manifolds and Nonlinear Dimension Reduction via Local Tangent Space Alignment." In: *SIAM Journal of Scientific Computing* 26 (2002), pp. 313–338.

[47] *An Introduction to Neural Networks and Autoencoders*. https://www.alanzucconi.com/2018/03/14/an-introduction-to-autoencoders/. Accessed: 2019-01-30.

[48] L. Van Der Maaten, E. Postma, and J. Van den Herik. "Dimensionality reduction: a comparative." In: *J Mach Learn Res* 10 (2009), pp. 66–71.

[49] L. Krippahl and P. Barahona. "Protein docking with predicted constraints." In: *Algorithms Mol Biol* 10 (2015), pp. 9–9. ISSN: 1748-7188. DOI: 10.1186/s13015-015-0036-6. URL: https://www.ncbi.nlm.nih.gov/pubmed/25722738.

[50] T. Feng, F. Chen, Y. Kang, H. Sun, H. Liu, D. Li, F. Zhu, and T. Hou. "HawkRank: a new scoring function for protein–protein docking based on weighted energy terms." In: *Journal of cheminformatics* 9.1 (2017), p. 66.

[51] H. Hwang, T. Vreven, J. Janin, and Z. Weng. "Protein–protein docking benchmark version 4.0." In: *Proteins: Structure, Function, and Bioinformatics* 78.15 (2010), pp. 3111–3114.

[52] J. Bergstra, O. Breuleux, F. Bastien, P. Lamblin, R. Pascanu, G. Desjardins, J. Turian, D. Warde-Farley, and Y. Bengio. "Theano: A CPU and GPU Math Compiler in Python." In: *SCIPY 2010* (2010).

[53] G. R. Team. *TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems*. https://www.tensorflow.org/. 2015.

[54] *Why AI and machine learning researchers are beginning to embrace PyTorch*. https://www.oreilly.com/ideas/why-ai-and-machine-learning-researchers-are-beginning-to-embrace-pytorch. Accessed: 2019-01-30.

[55] F. Chollet et al. *Keras*. https://keras.io. 2015.

[56] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. "Scikit-learn: Machine Learning in Python." In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.

[57] E. Jones, T. Oliphant, P. Peterson, et al. *SciPy: Open source scientific tools for Python*. [Online; accessed <today>]. 2001–. URL: http://www.scipy.org/.

[58] J. D. Hunter. "Matplotlib: A 2D graphics environment." In: *Computing In Science & Engineering* 9.3 (2007), pp. 90–95. DOI: 10.1109/MCSE.2007.55.

[59] W. Mckinney. *pandas: powerful Python data analysis toolkit*. 2018. URL: https://pandas.pydata.org/.

[60] N. M. O'Boyle, M. Banck, C. A. James, C. Morley, T. Vandermeersch, and G. R. Hutchison. "Open Babel: An open chemical toolbox." In: *Journal of Cheminformatics* 3.1 (2011), p. 33. ISSN: 1758-2946. DOI: 10.1186/1758-2946-3-33. URL: https://doi.org/10.1186/1758-2946-3-33.

[61] P. J. A. Cock, T. Antao, J. T. Chang, B. A. Chapman, C. J. Cox, A. Dalke, I. Friedberg, T. Hamelryck, F. Kauff, B. Wilczynski, and M. J. L. de Hoon. "Biopython: freely available Python tools for computational molecular biology and bioinformatics." In: *Bioinformatics* 25.11 (2009), pp. 1422–1423. DOI: 10.1093/bioinformatics/btp163. URL: http://dx.doi.org/10.1093/bioinformatics/btp163.

[62] Schrödinger, LLC. "The PyMOL Molecular Graphics System, Version 1.8." 2015.

[63] B. Adhikari and J. Cheng. "Protein residue contacts and prediction methods." In: *Data Mining Techniques for the Life Sciences*. Springer, 2016, pp. 463–476.