UNIVERSITAT POLITÈCNICA DE VALÈNCIA

# UNIVERSITAT POLITÈCNICA DE VALÈNCIA

## School of Industrial Engineering

# Development of a predictive system for the generation of biosensor libraries with application to the dynamic regulation of bioproduction pathways

End of Degree Project

Bachelor's Degree in Biomedical Engineering

AUTHOR: Moreno López, Raúl

Tutor: Carbonell Cortés, Pablo Jorge

Experimental director: TELLECHEA LUZARDO, JONATHAN ALEXANDER

ACADEMIC YEAR: 2021/2022

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

ESCUELA TÉCNICA SUPERIOR INGENIERÍA INDUSTRIAL VALENCIA

**TRABAJO FIN DE GRADO EN INGENIERÍA BIOMÉDICA**

# DEVELOPMENT OF A PREDICTIVE SYSTEM FOR THE GENERATION OF BIOSENSOR LIBRARIES WITH APPLICATION TO THE DYNAMIC REGULATION OF BIOPRODUCTION PATHWAYS


AUTHOR: Raúl Moreno López

SUPERVISOR: Pablo Jorge Carbonell Cortés

Jonathan Tellechea Luzardo


**Academic year: 2021-22**

# Acknowledgements

*A Pablo y Jonathan, por todo lo que me habéis enseñado.*

*A mis padres, mi hermano y mi tía, por el apoyo y la confianza.*

*A los tertulianos y compañía, por brillar con luz propia.*

*Y a la que se ha convertido en mi familia aquí en Valencia.*
*Por elevarme, por escucharme, por cuidarme. Eternamente agradecido.*

# Abstract

The aim of the Final Degree Project is the development and implementation of optimisation and machine learning techniques for the development of biosensors based on transcription factors. The Project will focus on the development of machine learning tools from a database of transcription factors responsive to molecules and metabolites.

The objectives of the Project are the following:

1. Collection of experimental data from biosensors based on transcription factors.

2. Use of machine learning algorithms for the development of predictive systems using tools such as Keras with Tensorflow in Python to train learning sets from chemical, biological and process data.

3. Development of an open source library based on the use of machine learning methods.

**Keywords: biomanufacturing; automation; robotics; simulation; dynamic process; optimization; metabolic engineering.**

# Resumen

El objetivo del TFG es el desarrollo y la aplicación de estrategias de optimización y técnicas de machine learning para el desarrollo de biosensores basados en factores de transcripción. El TFG se centrará en el desarrollo de herramientas de aprendizaje automático a partir de una base de datos de factores de transcripción responsivos a moléculas y metabolitos.

Los objetivos del TFG son los siguientes:

1. Recopilación de datos experimentales de biosensores basados en factores de transcripción.

2. Empleo de algoritmos de machine learning para el desarrollo de sistemas predictivos empleando herramientas como Keras con Tensorflow en Python para entrenar conjuntos de aprendizaje a partir de datos químicos, biológicos y de proceso.

3. Desarrollo de una librería en código abierto basada en el empleo de métodos de machine learning.

**Palabras clave: biomanufactura; automatización; robótica; simulación; proceso dinámico; optimización; ingeniería metabólica.**

# Resum

L'objectiu del TFG és el desenvolupament i l'aplicació d'estratègies d'optimització i tècniques de machine learning per al desenvolupament de biosensors basats en factors de transcripció. El TFG es centrarà en el desenvolupament de ferramentes d'aprenentatge automàtic a partir d'una base de dades de factors de transcripció responsius a molècules i metabòlits.

Els objectius del TFG són els següents:

1. Recopilació de dades experimentals de biosensors basats en factors de transcripció.

2. Utilització d'algoritmes de machine learning per al desenvolupament de sistemes predictius fent ús de ferramentes com Keras amb Tensorflow en Python per entrenar conjunts d'aprenentatge a partir de dades químiques, biològiques i de procés.

3. Desenvolupament d'una llibreria en codi obert basada en la utilització de mètodes de machine learning.

**Paraules clau: biomaufactura; automatització; robòtica; simulació; procés dinàmic; optimització; enginyeria metabòlica.**

# Contents

# List of Figures

# List of Tables

**Document I**

# Project Report

# Chapter 1

# Introduction

*In this chapter, the main objectives of this Project will be described, as an introduction of the steps to be followed to accomplish the development of the proposed biosensors predictive model.*

In order to achieve the development of a biosensor library based on a predictive system, several steps must be followed. These will be guided through different main targets, which will structure this Project and set the path to reach its final objective.

**1. Collection of experimental data from biosensors based on transcription factors.**

For the development of this Project, a database containing information about molecules and transcription factors (TF from this time forward) will be used as starting point (built by the experimental director of this Project, Jonathan Tellechea). Here, some molecule data is collected together with TFs (and its sequences) that can be, theoretically, used as biosensors to sense the molecule they are associated with.

On the basis of this database, a data augmentation will be performed, as well as an information search of the given molecules and TFs, in order to find data that might be useful for the development of the predictive system.

The activities carried out for this purpose will be described in Chapter 3, but they include a search of homologous of the available sequences, the clustering of homologous found and obtaining the motifs for each sequence.

Moreover, a bibliography revision will be performed in Chapter 2 with the aim of understanding how TF based biosensors work and which features are currently being used in the field of protein function prediction.

**2. Use of machine learning algorithms for the development of predictive systems.**

As machine learning algorithms are the order of the day, it has been considered appropriate to use them to develop the predictive model that will allow to build the biosensor library.

Among all the available techniques (which will be reviewed in Chapter 2) it has been decided to select those based on deep learning. The main reason is that it is believed that neural networks will be able to find those features that best describe the relationships between the TFs and the molecules they are intended to sense.

For this purpose, different strategies (described in Chapter 3) will be followed, concerning each of the aspects to be taken into account to build a deep learning model. First of all, the network architecture will be decided, as it is needed to plan the following steps. It will be based on Long Short-Term Memory layers, which are suitable for sequential data as the ones in question.

Secondly, different data codification processes will be carried out. On the one hand, for exploring which data representations lead to a better model performance. On the other hand, for adapting the codification to each type of data to be used. These include the TF sequences, the molecules and the labels that will establish the relationship between both, since the technique used involves a supervised learning.

Then, data will be prepared for training the model. Here, how its performance will be evaluated must be considered, as it will condition how data is split. The K-Fold Cross-Validation technique will be carried out, so the observations will be divided in training data and test data, and then the firsts ones will be split again to obtain the validation groups.

Last but not least, it is necessary to decide on different types of parameters in order to build the model and ensure a good performance. On the one hand, parameters concerning the layers of the model are of importance for the purpose of having a good acceptance of the input data. On the other hand, the hyperparameters control how well the training will perform.

**3. Development of an open source library based on the use of machine learning methods.**

Once the model is built and working properly, it is of great interest to develop an open source library based on the model that allows new predictions on the affinity between TFs and molecules.

To this effect, an attempt to implement the model in a web server will be carried out. The platform in question is Sensbio, an online server for biosensor design developed for the Dynamic BioDesign Lab (DBDL), belonging to the supervisor of this Project, Pablo Carbonell (available at *https://sensbio.carbonelllab.org*).

Associated with this platform there is a database with different molecules and TFs. The objective is to use the predictive model to find relationships between new molecules or proteins introduced by the user and those already available in the database.

# Chapter 2

# State of the art

*In this chapter, bibliography will be inspected to establish the current state of biosensors based on TF and those protein function prediction techniques based on machine learning. Also, some key concepts will be described for better understanding of this Project.*

## 2.1 Transcription factor-based biosensors

As described in Section 1, the main target of this Project is to obtain biosensor libraries from a predictive model. Therefore, it is convenient to describe what biosensors are, how they work and which applications they have nowadays, among other matters.

### *2.1.1 Biosensors description*

Biosensors can be found naturally in cells, measuring its state and responding accordingly to it, for instance, with pathway regulation. However, as synthetic biology and bioproduction are increasingly the order of the day, the engineering of biosensors for these purposes has become essential (Michener et al. 2012). So, in the context of this Project, a biosensor is described as a system that can detect the presence or absence of a determined metabolite and generate a response which is used to regulate the bioproduction pathways.

Biosensors are sorted into two main groups: electrochemical, where the output is an electrical signal, and optical, where the output is based on the fluorescence, the luminescence or the absorbance (Wan, Marsafari, and Xu 2019).

In metabolic engineering, several types of biosensors can be found (Mehrotra 2016). In first place, fluorescence resonance energy transfer (FRET) effect is used to sense the presence of a determined ligand. FRET is the response generated when an interaction between two fluorophores (a donor and an acceptor) occurs (Kikuchi, Takakusa, and Nagano 2004). When a ligand-binding peptide is used together the fluorophores, and the target ligand binds, the FRET effect generates the output.

Secondly, riboswitches have been used since the beginnings of synthetic biology. They consist in domains found in the non-coding portions of some mRNAs. Therefore, when a metabolite binds, changes in processes of gene expression can be observed (Mandal and Breaker 2004). In this way, it is possible to develop, for example, RNA-based fluorescent biosensors.

Last but not least, transcription factors are proteins that regulate the transcription rate. These will be explained in more detail below, as well as its utility as biosensors.

### *2.1.2 Biosensors based on TFs*

As just mentioned, a transcription factor (TF) is a protein that is able to regulate the transcription rate of a given gene by binding to a DNA sequence (Latchman 1997). The transcription is regulated due to the fact that bound TFs recruit or block the RNA polymerase responsible for the transcription.

TFs can act as repressors or activators of the transcription process (Wan, Marsafari, and Xu 2019). In the first case, RNA polymerase is recruited when the protein is bound to the transcription factor binding site on the DNA sequence, while in the second case, polymerase is recruited in the absence of the TF. The binding or not of the TF to the DNA depends on the presence or absence of a metabolite, which binds with the protein to create the complete repressor or activator complex. These processes are represented in Figure 2.1. In this way, if an output could be generated from the union of the metabolite and the TF, these could be used as biosensors.

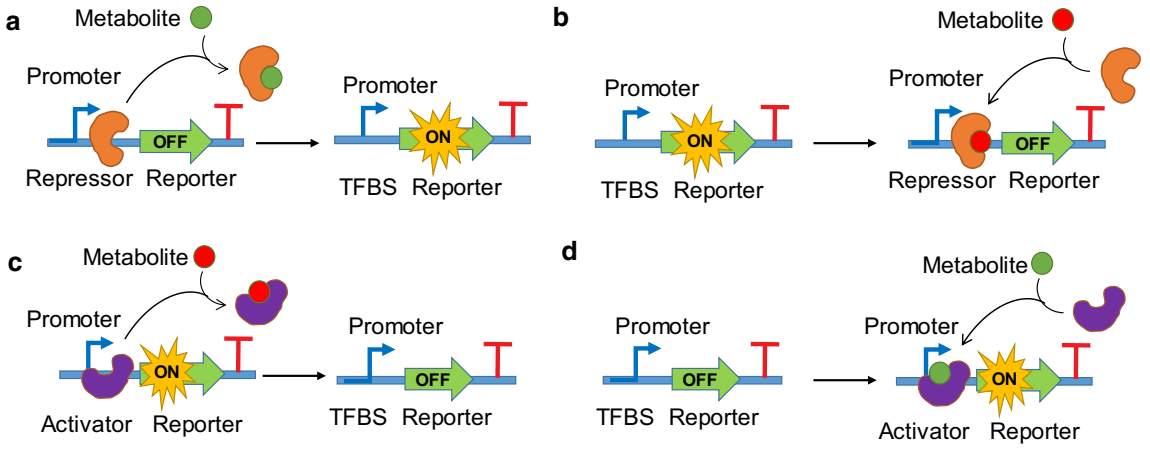


**Figure 2.1:** **a** TF as repressor. Transcription blocked in the absence of the metabolite. **b** TF as repressor. Transcription blocked in the presence of the metabolite. **c** TF as activator. Transcription promoted in the absence of the metabolite. **d** TF as activator. Transcription promoted in the presence of the metabolite. Figure extracted from *Wan, Marsafari, and Xu 2019*.

It is in the basis of this principle that TF-based biosensors are generated. As mentioned in Section 2.1.1, biosensors with an optical output are commonly used in metabolic engineering, and thus it is the case for TFs. This is possible thanks to the use of an actuator

in the architecture of the biosensor (Mahr and Frunzke 2016). Here, a fluorescent protein as the superfolder GFP gene can be used to generate a response (Ding, Zhou, and Deng 2021). The architecture of this type of biosensors and how the output is obtained are illustrated in Figure 2.2.

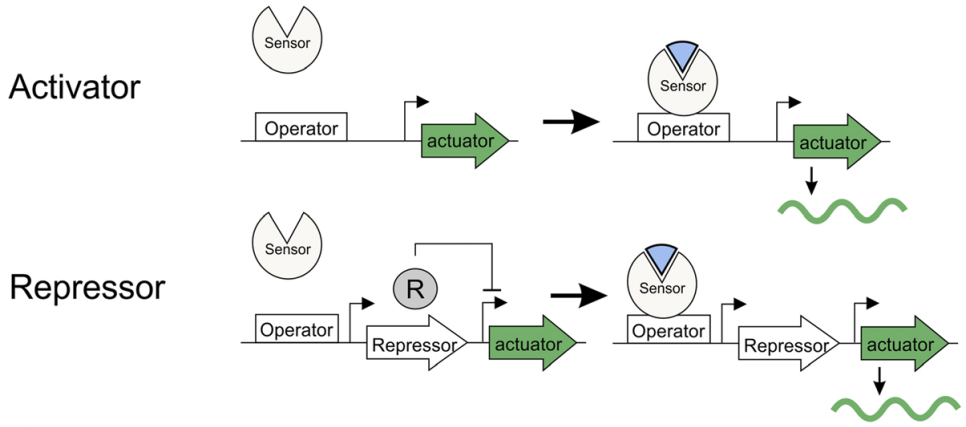


**Figure 2.2:** TF-based biosensor architecture. When the TF (sensor) is bound to the DNA sequence together with the metabolite (blue triangle), the fluorescence response is activated. Figure extracted from *Mahr and Frunzke 2016*.

In relation with the applications of TF biosensors, several options can be found nowadays. Among these are:

- The measurement of metabolite concentrations (Baumann et al. 2018) (Chen et al. 2018),
- High-throughput screening (L. Li et al. 2019) (Binder et al. 2012), which allows to screen through large chemical libraries to accelerate drug discoveries (Mayr and Bojanic 2009),
- Biosensor-mediated adaptive evolution for increasing metabolite production (Stella et al. 2019) (S.-D. Liu et al. 2017),
- and Dynamic pathway regulation (Xu et al. 2014), since metabolic fluxes can be detected and regulated by TFs during bioproduction.

### *2.1.3 Relevance of biosensor libraries*

One of the main problems found in TF-based biosensors is the limitation in the number of known metabolites that can be sensed using such proteins (Mahr and Frunzke 2016). There are some options to solve this matter. For instance, if an undetectable compound is wanted to be sensed, this can be transformed into a detectable metabolite through enzymatic pathways (Libis, Delépine, and Faulon 2016) (Delépine et al. 2016).

However, these pathways require to be tailor-made, which entails large costs in terms of investigation. This is where the importance of the use of predictive models lies. Gen-

erating biosensors libraries from predictions enables to reduce times of investigation, as this can be directly focused towards the guesses made by the model.

## 2.2 Machine learning for protein function prediction from sequences

In this section, a review of different machine learning methods will be performed, paying special attention to how they have been used for protein function prediction. How the data, which in this Project are sequences, are commonly prepared will also be discussed.

### *2.2.1 Why are machine and deep learning techniques needed in this field?*

To understand why machine learning techniques are increasingly required for predicting protein function, it is fundamental to define what "function" means. Traditionally, a protein function was a single and local action on a molecule or state to transform it into another. However, a post-genomic view of this term arose (Eisenberg et al. 2000). Currently, the context in which the protein is involved is considered, and its function is defined as how the protein relates with other proteins, taking into account each of the interactions generated. Therefore, "each protein plays a role in an extended network of interacting molecules" (Bonetta and Valentino 2020).

To define completely this new meaning of the function, Gene Ontology terms (Ashburner et al. 2000) are used. In this way, three interdependent levels of the protein function are defined:

- Molecular function, which refers to the traditional meaning of function and describes the protein activity at a molecular level. It is commonly predicted using computational methods, for instance, homologous search.
- Biological process, which defines a set of molecular functions. A metabolic pathway is an example of biological process. Genomic inference methods are used.
- Cellular component, which describes where a protein carries out its function in the cell. It is important to predict the protein function as it can help to identify the drug targets.

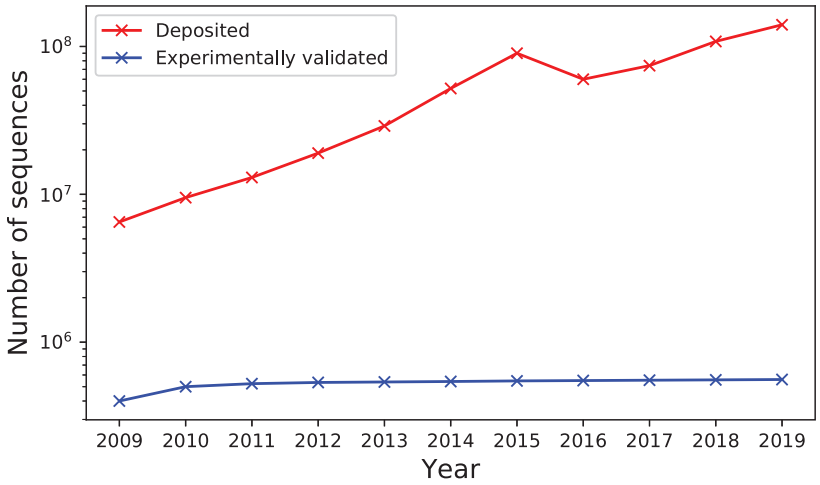


**Figure 2.3:** Number of sequences discovered (red) and with known function (blue) in UniProtKB over the last years. Figure extracted from *Bonetta and Valentino 2020*.

Therefore, it has become a great challenge to define the function of a new protein, as it is involved in a large number of biological processes. In fact, as observed in Figure 2.3, the number of sequences deposited in databases keeps growing each year, whereas the number of proteins whose function has been experimentally validated remains almost unchanged.

Consequently, a huge gap between the number of proteins with known and unknown functions has been generated. This is where machine learning can play a role. By predicting protein functions, experiments to validate them can be directly focused towards the solution.

Also, as interaction with all of the cellular processes could be highly complex, deep learning techniques are of great relevance. These models are created with a large number of relationships between their components governed by a series of weights that are updated during training to learn how the data relates to each other. Thus, they have the ability to learn characteristics and mechanisms that are even poorly understood by humans.

### *2.2.2 Machine learning techniques*

As seen above, machine learning models can be of great help for protein function prediction. Over the last years, several methods and algorithms have been used to solve the classification problem that involves the association of a protein sequence to a determined action. Some of the most popular machine learning methods are presented here, as well as concrete examples of their usefulness in this field.

- SVM algorithm (Boser, Guyon, and Vapnik 1992). It is a supervised model (that is to say, it requires output labels or ground truth) that aims to classify a set of observations in two groups, maximizing the distance between them and establishing

a maximum margin. It was first used when machine learning started to be considered for protein function prediction (Cai et al. 2003) (Lanckriet et al. 2003).

- kNN algorithm (Altman 1992). The k-nearest neighbours algorithm classifies an observation based on the classes to which the nearest observations pertain, using the majority vote to decide the group. Some attempts to predict protein function through kNN algorithm have been carried out (Hu et al. 2011) (De Ferrari and Mitchell 2014).
- Logistic regression. It is also a supervised model, which uses a sigmoid function to try to predict the class of a given observation depending on the values of other independent variables (Kleinbaum et al. 2002). Due to its simplicity, it has been used in some specific studies, as for protein interaction networks (Lee et al. 2006) or protein function prediction based on protein-protein interaction (Ni et al. 2009).
- Naive Bayes. This type of classifiers can be trained by supervised learning and are based on Bayes' theorem. It assumes that each variable is independent from the others when calculating the probability of belonging to a certain group (Rish et al. 2001). An extended version of this algorithm has been developed to predict protein and gene functions (Campos Merschmann and Freitas 2013).
- Ensemble methods. This term refers to those methods that use more than one algorithm together to perform predictions. There are mainly two types of ensemble methods that have been used for protein function prediction.
  - Bagging (Breiman 1996). It is a classification technique that tries to reduce the error variance. It consists in training several classifiers, each with a slightly different dataset, and then averaging the results. One of the most used methods is the random forest technique (Breiman 2001), which combines several decision trees. This has been used to predict glycosylation sites (F. Li et al. 2015) and for enzyme function classification (Kumar, G. Li, and Choudhary 2009).
  - Boosting. It is similar to the bagging technique, but this time trainings are carried out in series, not in parallel. In this way, the result of a classifier is used to train the next classifier. The most popular method is AdaBoost (Freund, Schapire, et al. 1996). This gives more importance to those observations that were misclassified in the previous iteration, trying to correct the errors in the next classifiers. It has been used, for example, to predict protein structural class (Niu et al. 2006).

### *2.2.3 Deep learning techniques*

Deep learning is a subtype of machine learning that uses a combination of algorithms to learn about more abstract characteristics and data representations. As explained in Section 2.2.1, this ability can be exploited in protein function prediction in order to achieve models that take into account features and relationships that are beyond the reach of humans.

The above-mentioned combination of algorithms allows to simulate non-linear classifiers, which distinguishes them from the other methods explained in Section 2.2.2. Nevertheless, the main difference between these two types of techniques lies in the feature engineering. In machine learning methods, a previous process of feature extraction must be done to find those variables that best explain the variability of the data in order to build and train the model successfully. Meanwhile, although it is recommended sometimes, this process could be skipped in deep learning techniques as the model is able to do the feature engineering on its own. Otherwise, the level of abstraction that defines deep learning could not be achieved.

Another fact that characterises deep learning is its architecture. Usually, these models are built simulating a neural network, that is to say, they are made up of a series of layers composed of a number of "neurons" that are interconnected with each other. A representation of this concept is given in Figure 2.4. The term hidden layers refers to the fact that, although the number of layers and some aspects of them can be decided, they form a black box for the user in terms of the operations carried out in their neurons.

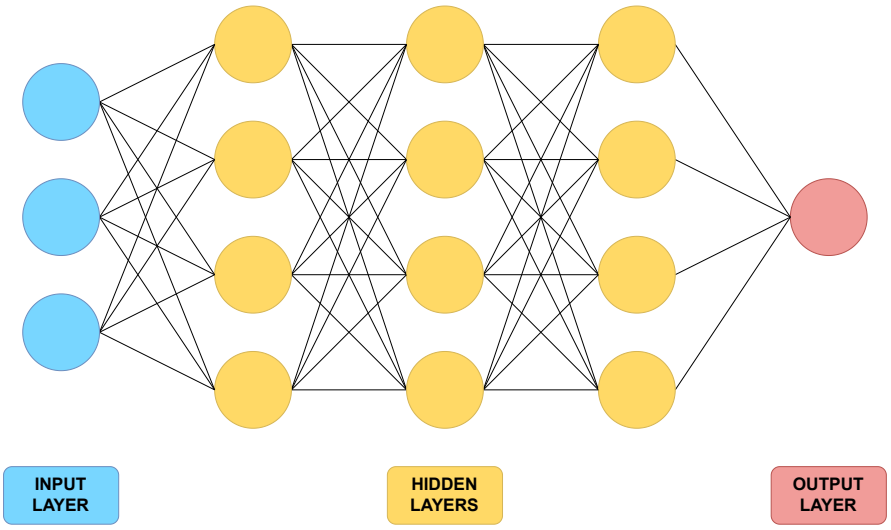


**Figure 2.4:** Representation of a neural network example

As a representative example of how deep learning can contribute to the matter in question, the software DeepGO (Kulmanov, Khan, and Hoehndorf 2017) is able to predict protein function taking into account the sequence and protein-protein interaction data.

For this purpose, it consists of three different models, one for each of the function levels described by the Gene Ontology (explained in Section 2.2.1).

Moreover, two of the most common neural networks are CNNs and RNNs. These are described here, as well as some models examples that can be of great use for protein function prediction.

- Convolutional Neural Network (CNN). This consists in a specialized network where some of its hidden layers perform an operation known as convolution (Albawi, Mohammed, and Al-Zawi 2017). Convolution connects local regions of the data to the neurons, so the total number of connections is reduced and the method becomes more efficient. CNNs are commonly used in problems where the data are images. These, and more specifically immunohistochemistry images, are used in cellular component prediction (Shao, M. Liu, and Zhang 2016), one of the function levels described in Section 2.2.1. The most popular CNN used is AlexNet (Krizhevsky, Sutskever, and Hinton 2012), which has proved to have low error levels. It has been used, for instance, to predict protein subcellular localisation (Su et al. 2021).
- Recurrent Neural Network (RNN). This type of networks have the ability to process temporal information (Medsker and Jain 1999). That is to say, they are able to storage information about data that has already been seen in previous states to perform predictions. It is for this reason that RNNs are widely use for temporal data and sequences, such as the protein amino acid sequences. Therefore, they are useful for predicting protein function (X. Liu 2017). The best known type of RNN is the one that includes LSTM layers (Hochreiter and Schmidhuber 1997), which can remember previous states over a long period. So, they are useful with long sequences as the amino acids ones.

### *2.2.4 Working with sequences*

As seen through the different examples given in Sections 2.2.2 and 2.2.3, when machine learning techniques were applied they required different types of input data, so the way in which proteins were represented was different each time. The following question therefore arises: which are the best features to represent proteins in order to get better models?

There is not a right answer, as each feature provides different advantages and disadvantages in relation with the information that the model can use to perform its predictions. A summary of these characteristic is shown in Table 2.1. The features considered are the protein physicochemical properties, the amino acids sequence, the protein-protein interaction and text based features extracted from biomedical literature. Also, representation learning is taken into account as it has been used in several studies (Gligorijević, Barot, and Bonneau 2018) (Kulmanov, Khan, and Hoehndorf 2017).

**Table 2.1:** Advantages and disadvantages of different types of features used to represent proteins for protein function prediction. Adapted from *Bonetta and Valentino 2020*.

| FEATURE | ADVANTAGES | DISADVANTAGES |
|---|---|---|
| Physicochemical properties | Simple and numeric | Insufficient information about the protein |
| Sequence-based | Capture plenty of information | Conversion process to numeric data required |
| PPI networks | Sharing functions with neighbouring proteins | Reliability of PPI data depends on the experimental source |
| Biomedical text | Rich source of information | Results strongly affected by how informative the selected terms are |
| Representation learning | No need for manual feature engineering and selection | More computational power and larger datasets needed |

It is worth paying attention to the representation with sequences, as it is the way in which most of the information about the protein is provided. This is due to the close relationship between the amino acids sequence and the protein function. Moreover, only a data conversion is required with the aim of preparing the sequence for use in a model.

For this purpose, several techniques are available. One of the most popular is the one-hot encoding. It consists in a vector of zeros where only a determined bit is a one. In this way, for amino acids sequences the bit with a one would be the bit indicating which amino acid is at a certain position in the sequence.

# Chapter 3

# Methodology

*In this chapter, the methods used to accomplish the objectives specified previously will be described. These methods are grouped into the two major steps that comprise the development of this Project: the preparation and study of the data and the building of the neural network which will be used to create the biosensor library.*

## 3.1 Data preparation

In order to make the available data appropriate to be used to train the predictive model, several steps must be taken. Those include an analysis of the database used, a data augmentation, if more observations are required to train the model, and a search of possibly useful information to improve the model performance.

### *3.1.1 Database initial analysis*

The database used consists of a list of molecules which are identified by their common name, their International Chemical Identifier (InChI) and their SMILES (Simplified Molecular Input Line Entry Specification). Other information about the organism of each molecule is given. In addition, each molecule has associated a TF. These are identified by their common name, the NCBI accession number and the amino acids sequence.

The most relevant information here are the molecule SMILES and the amino acid sequences, as they can be used to train the predictive model (how this data is prepared in order to be used as input for the neural network will be discussed in Section 3.2).

Also, these variables will provide the necessary information to apply the following techniques.

### *3.1.2 Search of homologous*

An important step to be taken in order to train a deep neural network is to make sure that the amount of available data is enough to perform a successful training. Bearing this in mind, a data augmentation must be carried out.

In this way, a homologous search of the TFs from the database is performed. As homologous, it is logical to expect that their functions will be similar, so they can be associated with the same molecule that their corresponding TF has been associated with. Therefore, the pair molecule - sequence needed for the data augmentation will be completed.

This search is performed using the BLAST algorithm (Altschul et al. 1990) through the NCBI platform. BLAST (Basic Local Alignment Search Tool) is an algorithm which allows to compare sequences rapidly, so computing times are reduced enough to make it possible to look for a homologous sequence through a huge sequence database.

First of all, conditions to execute BLAST have to be set up. As the aim is to obtain a large number of homologous to train the model, the number of hits decided for each BLAST execution is the maximum that the NCBI server allows. That is 5000. Then, since the inputs for the BLAST are proteins and so do the outputs, the type of algorithm used will be *blastp*. Finally, NCBI's databases where to search have to be decided. Looking for a balance between the database size and its quality, two databases where chosen: Reference proteins (*refseq_protein*) and UniProtKB/Swiss-Prot (*swissprot*).

With all the conditions established, several options are considered in order to execute BLAST trough the 3498 different TFs found in the database. The first one and more encouraging is based on using the Biopython library on Python (Chapman and Chang 2000), which allows to send querys directly to the NCBI site to avoid installing the BLAST algorithm and its databases.

Another option considered is parallel computing, in order to make different searches simultaneously to reduce even more times, as the list of sequences to search is extensive. In order to do so, the Galaxy Europe platform is used, which implements a variety of bioinformatic tools and allows to send multiple jobs which are executed in parallel (Jalili et al. 2020). The workflow established to obtain the results would be:

1. Uploading a *.txt* file with the Accession Numbers of the TF extracted from the database.
2. Obtaining the FASTA files from each Accession Number looking for them in GenBank and RedSeq databases. The FASTA format stems from the program of the same name (Pearson and Lipman 1988), and it is used to represent sequences simply and with a header which describes them.
3. Sending in parallel the BLAST queries giving as input the FASTA files obtained.
4. Obtaining the BLAST results from the two databases selected.

5. Obtaining the FASTA files from the BLAST hits for future tasks.

### *3.1.3 Clustering of sequences*

For the purpose of organising the vast number of sequences obtained from the BLAST search (5000 hits were looked for from 3498 sequences), a clustering of the found homologous is carried out. To this effect, the CD-HIT program is used (W. Li and Godzik 2006). This algorithm allows to cluster a huge number of sequences quickly, comparing and grouping them according to their similarity.

To avoid downloading and installing the program, the web server CD-HIT Suite is used, which also provides better accuracy, scalability and flexibility (Huang et al. 2010). The regular version of the algorithm is run. This, clusters the sequences depending on a threshold of similarity established by the user, and returns a FASTA file with the groups generated, each of which has a representative sequence. This main sequence is meant to be added to the data set to be trained along with the molecule associated to the TF of which it is a homologous.

To study different possibilities, a total of seven searches are carried out, each of these with a different similarity threshold. The cut-offs decided range between 0.6 and 0.9 (60% and 90% of similarity, respectively).

### *3.1.4 Search of motifs*

With the intention of guaranteeing a good performance of the model, a try to get useful information about the sequences is done. In this way, motifs of the original TFs of the database are looked for. A motif is a short sub-sequence of a protein which usually serves to differentiate which family the protein belongs to (Bork and Koonin 1996). Therefore, motifs are closely related to the protein's function, so the model learning about which part of the TF sequence contains the information about functionality might be useful.

The search of motifs is carried out using the PROSITE database, which collects a large number of "biologically meaningful signatures" (Hulo et al. 2006), that is to say, motifs. In order to compare the sequences with the PROSITE database to locate the motifs, the software ScanProsite has been downloaded and run via Python (to iterate over all sequences). This tool stands out from the rest as it implements totally the syntax established by PROSITE and complies with all the pattern scanning rules (Gattiker, Gasteiger, and Bairoch 2002).

Diagram in Figure 3.1 represents the whole process of data preparation prior to model building.

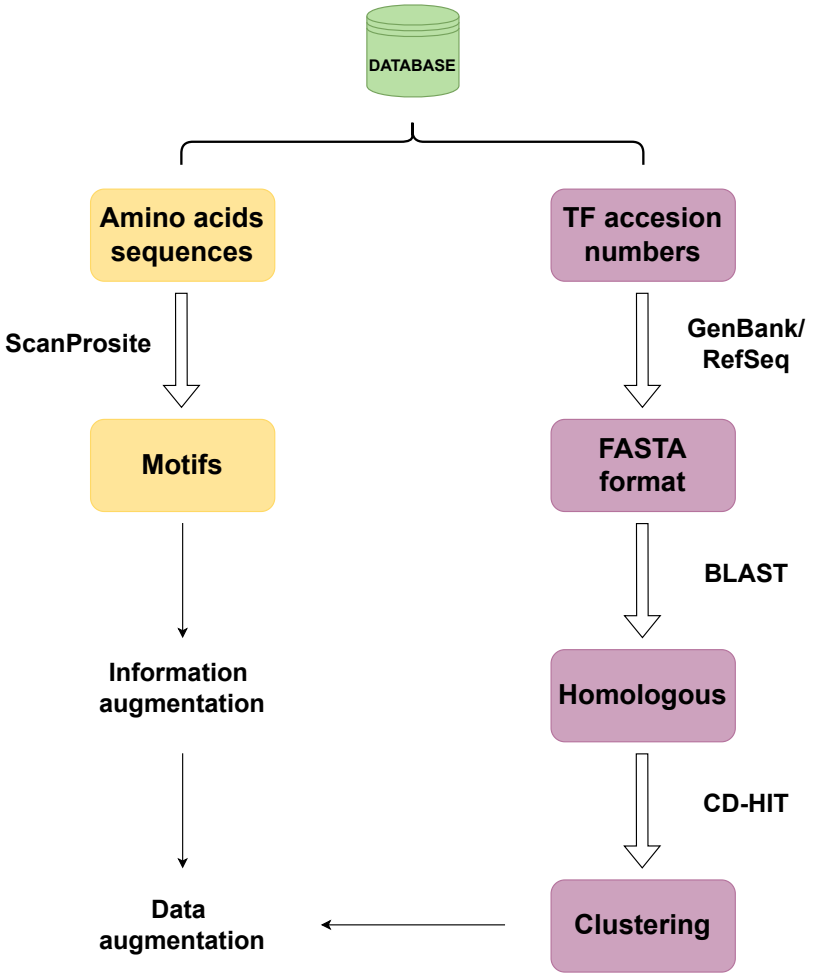


**Figure 3.1:** Complete data preparation diagram

## 3.2 Neural network: artificial intelligence model

Once all data is collected, it is time to prepare the artificial intelligence model. It is developed using machine learning techniques, in particular, deep learning ones. In this way, a neural network based on different layers has been established.

Moreover, different Python libraries have been used in order to develop the model. On the one side, Tensorflow, and more specifically its API Keras, allows to build and train the model giving a friendly and flexible interface. On the other side, Scikit-learn has been used to perform the model validation.

The different aspects to be taken into account are the architecture decided for the neural network, how the data is prepared to be acceptable as inputs for the net, the way the validation will be developed and the parameter selection for an acceptable performance of the model.

### *3.2.1 Network architecture*

When talking about the network architecture, it refers to which layers have been used and how they have been set up. A diagram of the architecture selected can be found in Figure 3.2. In this case, it has to be considered that two types of data will be given as inputs: the amino acids sequences of the TFs and the molecules associated to each one.

For the first one, an LSTM layer has been applied. LSTMs (Long Short-Term Memory) are a type of recurrent neural network, which have feedback connections, so they can learn from earlier states (Hochreiter and Schmidhuber 1997). For this reason, they are ideal for sequential data such as the ones in question.

For molecules, a simple dense layer is used. The term dense makes reference to a fully-connected layer, which means that all neurons of the layer are connected with the previous ones.

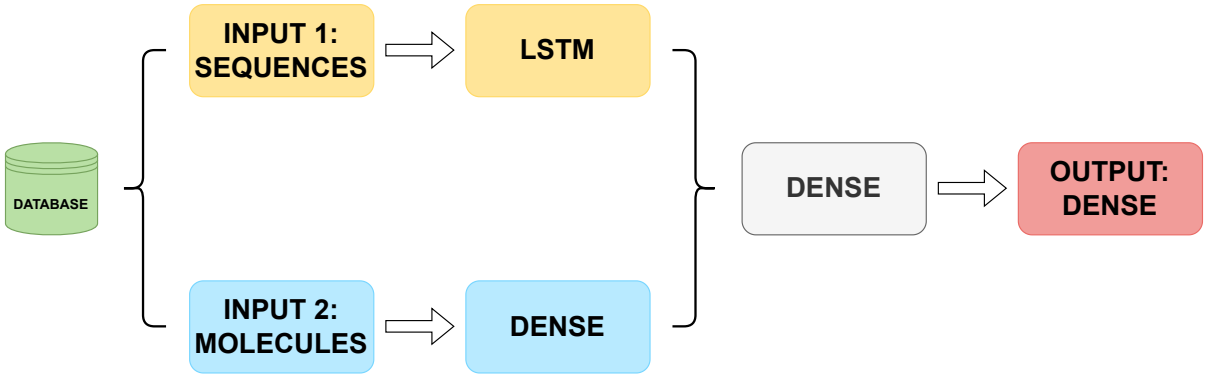


**Figure 3.2:** Network architecture diagram

After that, both inputs are concatenated and the model learns the relations between them for the first time in another dense layer. Finally, the output of the network will be given in the form of a last dense layer. This is expected to provide how much the sequence is related to the molecule, establishing if the TF can be used as a biosensor or not.

Actually, more dense layers could be used, but in order to keep the architecture as simple as possible, only the necessary ones have been set up. In this way, it will be easier to the model to learn about the characteristics of each input and how to relate them.

### *3.2.2 Input 1: sequences*

As mentioned, the model has two inputs which require a preparation for introduction into the network. The first of them concerns the amino acid sequences of the TFs. Those are composed of an alphabet of 20 characters, one for each amino acid. For the purpose of simplifying the input information, a data conversion is carried out.

Instead of using 20 different characters, the amino acids are represented in a matrix of ones and zeros using the one-hot encoding technique. In its columns, the position of

each amino acid is represented, so the matrix will have as many columns as the sequence length. Meanwhile, each row represents an amino acid, so there will be 20 rows. For each column, that is to mean, each position, a one will be placed in the row corresponding to the amino acid found in such position.

For understanding, an example is given. If a sequence starts with the amino acids MRRNK... the first columns of the matrix will look as in Figure 3.3. Note that just a few rows are represented.

| | 0 | 1 | 2 | 3 | 4 |
|---|---|---|---|---|---|
| K | 0 | 0 | 0 | 0 | 1 |
| M | 1 | 0 | 0 | 0 | 0 |
| N | 0 | 0 | 0 | 1 | 0 |
| R | 0 | 1 | 1 | 0 | 0 |

**Figure 3.3:** Amino acids matrix example for the sequence MRRNK

In order to simplify even more the input, a reduction of the dimensions of the matrix is considered. To achieve this, the amino acids can be grouped according to their characteristics, even if this means a loss of information. The groups taken are nonpolar aliphatic, nonpolar aromatic, polar uncharged, polar positively charged and polar negatively charged amino acids.

Another aspect to keep in mind is that not all sequences have the same length. If each matrix is built independently, dimensions will vary and this will lead to inconsistencies for the model. To avoid that situation, zero padding is apply to each matrix, giving them as many columns as positions has the largest sequence.

Furthermore, the information about motifs can be included in these matrices. There have been considered several options in order to do so:

1. A row for each motif. That is to say, an extra row is added for each motif found as explained in Section 3.1. In this way, there will be as many extra rows as different motifs found. Then, a one is placed in the row corresponding to the motif of the current sequence in the position where it begins.
2. Giving importance to the motifs' amino acids. For instance, placing a two instead of a one in positions where a motif is found in the sequence. This would mean matrices are not longer binary.

3. Not introducing the motifs. It could be possible that motifs do not give any relevant information during the training of the model.
4. Introducing just motifs. As motifs are supposed to have the information on the protein function, they might be enough to give the model the necessary knowledge to relate them with the molecule.

### *3.2.3 Input 2: molecules*

The second input to be introduced in the network refers to molecules. As commented in Section 3.1, SMILES of each molecule will be used. However, it is necessary to carry out a data transformation in order to make molecules understandable for the model.

Therefore, SMILES are converted into fingerprints, which are a way of representation of molecules by bits. Each bit is meant to represent the presence or absence of an specific characteristic (Bajorath 2001), so in this way the molecules are fully described. An example of how the fingerprint of a single molecule would look like is given in Figure 3.4 Moreover, as bits are represented by ones and zeros, data is given to the network in its simplest form.

| Characteristics | 1 | 2 | 3 | 4 | 5 | 6 | ··· | 512 |
|---|---|---|---|---|---|---|---|---|
| Fingerprint | 0 | 1 | 0 | 0 | 1 | 1 | ··· | 1 |

**Figure 3.4:** Example of fingerprint for a single molecule. Each bit indicates the presence or absence of a characteristic

For obtaining these fingerprints, the Python RDKit library is used, which is an open-source software for chemical informatics. A parameter to be described when using this toolkit is the fingerprint size. Two different sizes will be tested in the model looking for which one provides better performance: the default size (2048 bits) and a shorter size (512 bits), the latter to simplify the input.

Diagram in Figure 3.5 represents how both types of inputs are codified to be used in the model.

### *3.2.4 Labels: ground truth*

As machine learning is a type of supervised learning, a set of labels is required to tell the model which is the ground truth and to guide training based on these labels.

Bearing this in mind, a binary array is created to classify each pair sequence - molecule. If the TF is associated as a biosensor to the molecule, a one is placed in the labels set. Otherwise, a zero is used as label.

So far, as described in Section 3.1, only positive sequence - molecule pairs have been obtained. Though, negative labels are needed to train the model. Moreover, the number of positive and negative cases must be balanced to ensure a good model performance. For these reasons, each sequence of the data set is duplicated and associated to a selected molecule, with which it will not be useful as a biosensor. There have been considered two options to do this association: either the negative molecules are selected randomly or on the basis of the Tanimoto index (Bajusz, Rácz, and Héberger 2015). The Tanimoto index represents the similarity between fingerprints, so unalike molecules can be selected.

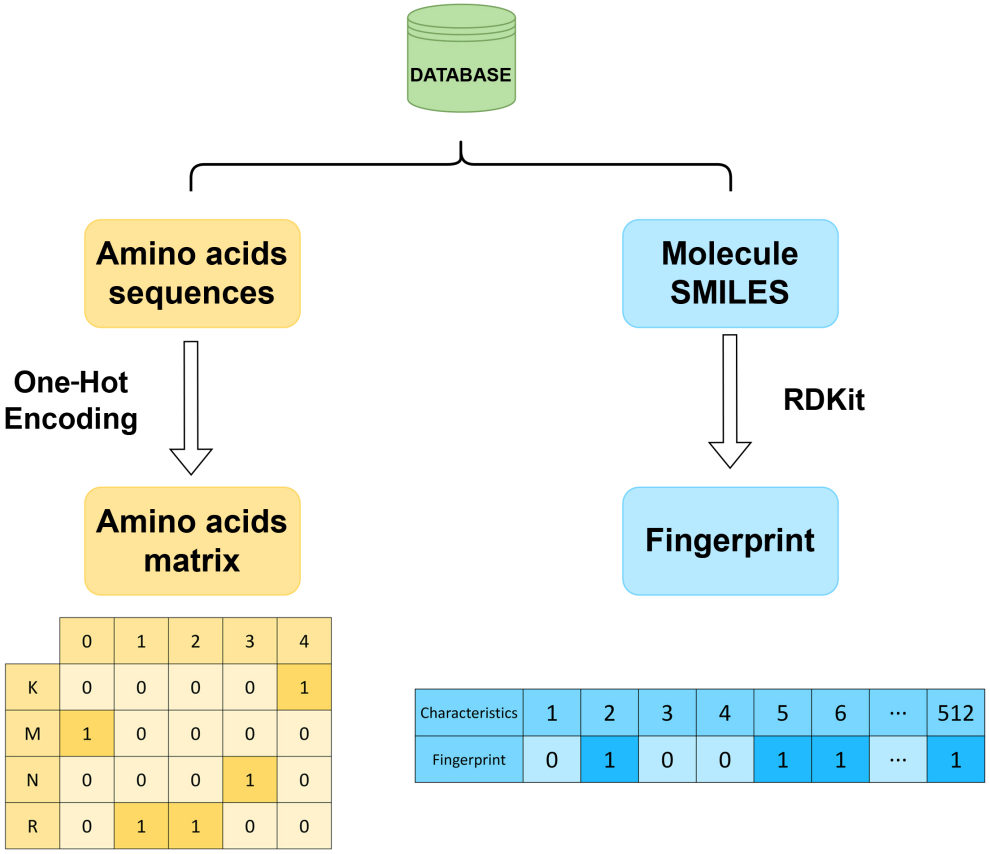


**Figure 3.5:** Data codification diagram

### *3.2.5 Training, test and validation*

When developing a predictive model such as the one in question here, the data set must be divided into two main groups: training data and test data. The first one will be used by the model to learn the weights associated to each neuron in order to make a good prediction. Meanwhile, the second one will be run through the trained model to obtain how accurate the prediction is. In this case, 20% of data will be used for testing.

However, before using the test data, several different models are tried. In order to select which one is the best, a validation process is carried out. This consists in using some of

the training data to make priory tests, the results of which are used to select the definitive model. Once the model is selected, the test data is used to obtain the actual accuracy of predictions.

In this case, a K-Fold Cross-Validation technique is performed (Refaeilzadeh, Tang, and H. Liu 2009). The training data is divided into $k$ folds of the same size. Then, $k$ model trainings are carried out. For each one, $k$ - 1 segments are used to train and the one left is used to test the performance. The fold selected to test the model in each iteration is different, so a different accuracy is returned each time. Once the $k$ accuracies are obtained, the mean of these is used as the model validation score. The Figure 3.6 illustrates this process.

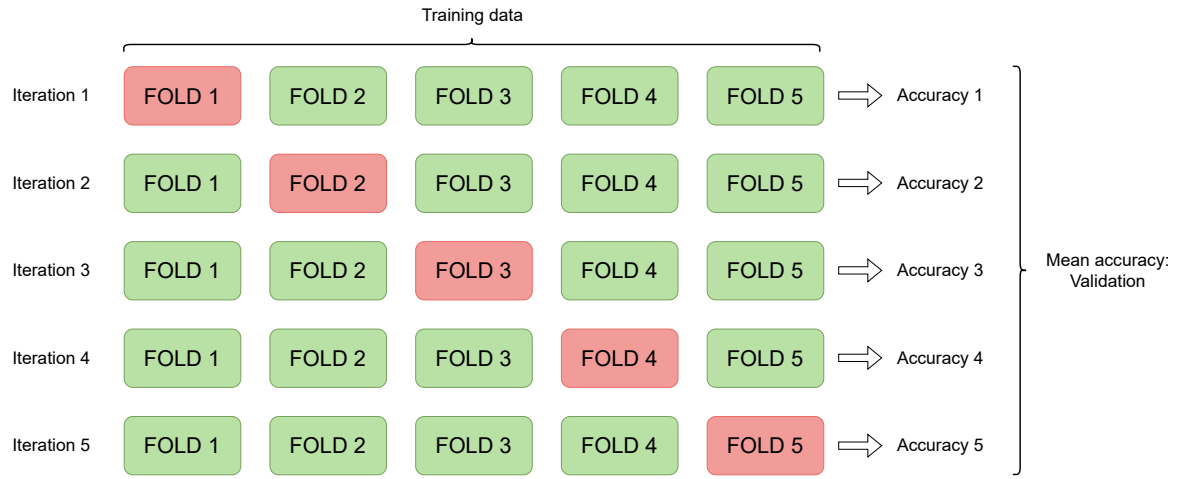


**Figure 3.6:** K-Fold Cross-Validation technique. For each iteration, green folds are used for training and red folds for testing

### *3.2.6 Parameter selection*

As explained above, the validation process aims to test different models. This includes testing the parameters that will define it. There have been considered two types of parameters: the number of units that each layer will have and the hyperparameters of the model.

Regarding the units' parameters, in the network structure there are four layers for which the number of neurons has to be decided. First of all, the layers receiving the inputs are found. The unique restriction taken into consideration here is that the number of units in the molecule branch (Dense layer) has to be less than the units in the sequence branch (LSTM layer). This is because it has been decided to give more emphasis on learning about sequences, where most of the information is contained. Moreover, a dropout has been applied to the LSTM layer in order to prevent overfitting (Srivastava et al. 2014).

Then, a Dense layer after the concatenation is found, but there are not restrictions on the number of units here. And lastly, there is the output layer, which has only a unit

as the result expected to be returned by the model is a single score about the affinity between the sequence and the molecule.

About the hyperparameters, they are used to control how the model learns. The hyperparameters selected to modify in this Project are the learning rate (step size of the model optimisation algorithm), the batch size (number of samples that the algorithm takes for each iteration) and the number of epochs (how many times the optimisation algorithm is run through all the data).

The range in which both, units' parameters and hyperparameters, move is defined first using a bayesian optimisation algorithm (Nogueira 2014), an then refined manually until the best parameters are found (though avoiding adjusting the parameters too much to prevent overfitting).

Finally, another two aspects to consider are the network optimisation algorithm and the activation function. On the one hand, the optimiser tries to reduce the loss function, which measures the differences between the predictions and the labels. The Adam algorithm (Kingma and Ba 2014) has been used, a stochastic gradient descent method that requires little memory, it is not computationally expensive and works well with large data. On the other hand, the function activation generates the output of each neuron based on its input. The ReLU (Rectified Linear Unit) function has been applied, as it avoids the optimisation algorithm to get trapped in local minima and accelerates the convergence (Hara, Saito, and Shouno 2015).

### *3.2.7 Model possibilities*

Although all of the considerations about the model made in this section are focused on a network where a sequence and a molecule are introduced as inputs and an affinity score (depending on the accuracy) is returned as output, other possibilities were considered. These are listed in Table 3.1.

**Table 3.1:** Model possibilities considered. The first one (darker blue) is the main model developed in this Project. Seq: sequence, Mol: molecule.

| INPUT | OUTPUT |
|---|---|
| Seq, Mol | Score |
| Seq | Mol (score) |
| Mol | Seq (score) |
| Seq | Generate Mol |
| Mol | Generate Seq |
| Seq, Mol | Mutations required |

The other possibilities include: entering a sequence or a molecule and obtaining its corresponding match from a database and its score, entering a sequence or a molecule and generating a likely corresponding match, or entering a sequence and a molecule and receiving the mutations required to be a match.

## 3.3 Model implementation

The last aspect to consider is the implementation of the model to generate the biosensor library. For this purpose, the web server Sensbio will be used, as explained in Section 1. This consists on a database where new relationships between TFs and molecules can be found by introducing new inputs.

The platform is divided into three main sections:

- The first one just provides a visualization of the database.
- The second one enables to search similarity between molecules. The user can introduce a new molecule in SMILES format and the Tanimoto index for each of the molecules in the database is calculated. Then, it returns the most similar molecules. The implementation of the model here consists on calculating the affinity of the molecule introduced with each of the TFs in the database.
- In the third section the user introduces the amino acids sequence of a protein and then the BLAST algorithm is run to obtain a score for similar proteins in the database, that is to say, likely homologous. In the same way, the prediction model developed will look for molecules in the database with high affinity with the sequence introduced.

Since a prediction score for each of the observations in the database is calculated each time, this leads to maybe expensive computing times. To avoid this situation, an option to run the model just through a selected number of observations will be included in the web server. The observations selected will depend on the most similar molecules according to the Tanimoto index in the second section, or on the most similar sequences according to the BLAST score in the third section.

It should be noted that a successful implementation of the model on the web server would accomplish two further possibilities for the use of the model, which were described in Table 3.1. These would be the second and the third options.

# Chapter 4

# Results

*In this chapter, results for every step taken to develop this Project will be presented. These will also be discussed in order to relate which of the methodologies described in Chapter 3 gives the best outcomes.*

## 4.1 Data analysis

A particularly important aspect when planning to train a predictive model is the available data and its quality. Therefore, it is essential to ensure that results from homologous and motifs searches are consistent, as well as that the initial database provides sufficient information.

### *4.1.1 Analysis of initial database*

Although a data augmentation is performed in case more observations are needed to train the network, mainly the initial data from the database will be used. After its analysis, there are a total of 5438 pairs molecule - TF available. These will turn into 10876 observations to be used in the training process, after the negative cases are created as described in Section 3.2.4.

### *4.1.2 Data augmentation: homologous*

As described in Section 3.1.2, two different methods were considered in order to find the homologous sequences via the BLAST algorithm. First of all, Biopython library was used. This made it possible to iterate through the sequences and send them, one by one, to the NCBI site where searches were executed. However, as this algorithm is widely used by the bioinformatic community, this process was extremely slow, although there was no problem with the results obtained. Finally, this option was discarded.

Secondly, bearing in mind the time issue, parallel computing was used. As Galaxy platform allowed to establish a work flow for obtaining the FASTA files needed and sending the BLAST queries, this method proved to be much faster.

The quality of the hits (homologous found) done by BLAST is measured with the E-value, which represents how many similar hits could have been found by chance. So, the lower the E-value is, the better. In Figure 4.1, the histogram of the maximum E-value obtained for each BLAST search is represented.

It is noted that the area with the highest density in the histogram is between the values -7 and -75 of the E-value logarithm, which corresponds with the E-values $10^{-3}$ and $3 \cdot 10^{-33}$, respectively. Considering these are just the maximum values of each query, it can be concluded that such low values represent a well performed search.

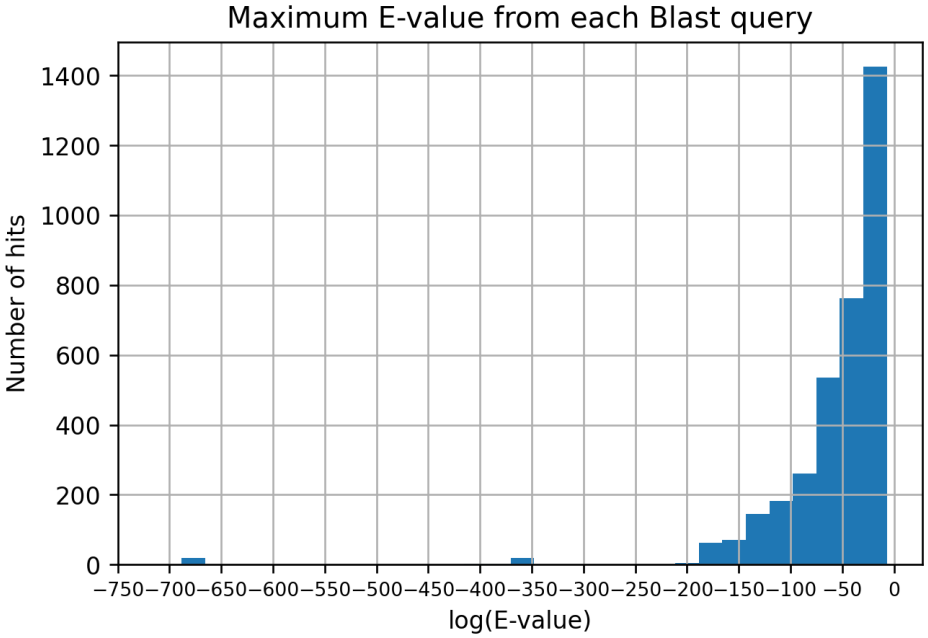


**Figure 4.1:** Maximum E-value histogram obtained for each BLAST search

In relation to the number of homologous found, there were more than 16 million hits found across all searches. However, the number of hits that are repeated in several searches is quite high, so the total number of hits obtained without repeating each other was almost 1.5 million.

With these results, another step taken was the homologous clustering. As described in Section 3.1.3, this was performed with different similarity thresholds. The number of clusters obtained for each cut-off is represented in Figure 4.2. As expected, when the threshold is more restrictive, less sequences are grouped together so more cluster are generated.

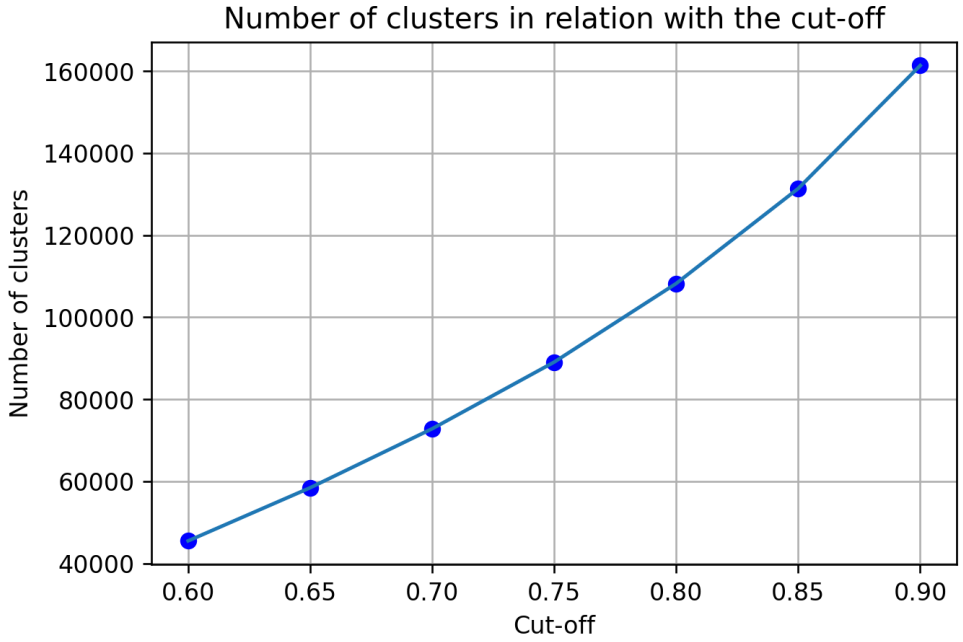


**Figure 4.2:** Number of clusters obtained depending on the threshold selected

A reference sequence for each cluster is selected to be used as training data, as explained in Section 3.1.3, so the number of clusters represents the number of observations which will be given as inputs for the neural network. Therefore, depending on the number of observations required, a different cut-off will be selected.

Whether the data augmentation is finally necessary or not will be discussed in Section 4.3, depending on the results of the training processes.

### *4.1.3 Database extension: motifs*

As explained in Section 3.1.4, motifs from the original sequences were extracted. Among these, a total of 5205 motifs were found, although they are repeated and just 49 are different. The frequency of occurrence of each one can be found in Figure 4.3. It can be observed that 29 of them have been found less than 50 times, while the rest have a higher frequency of occurrence. The motifs PS50949 and PS50932 stand out from the rest.

It should be noted that there are some sequences with more than one motif, whereas there are others with no motifs. Therefore, while training, the neural network will not have the same amount of information for each sequence.

Whether the information provided by motifs is useful or not in order to obtain a better accuracy in the model will be discussed in Section 4.3. There, the different options considered in Section 3.2.2 to introduce the motifs information as inputs will be evaluated depending on the model performance.

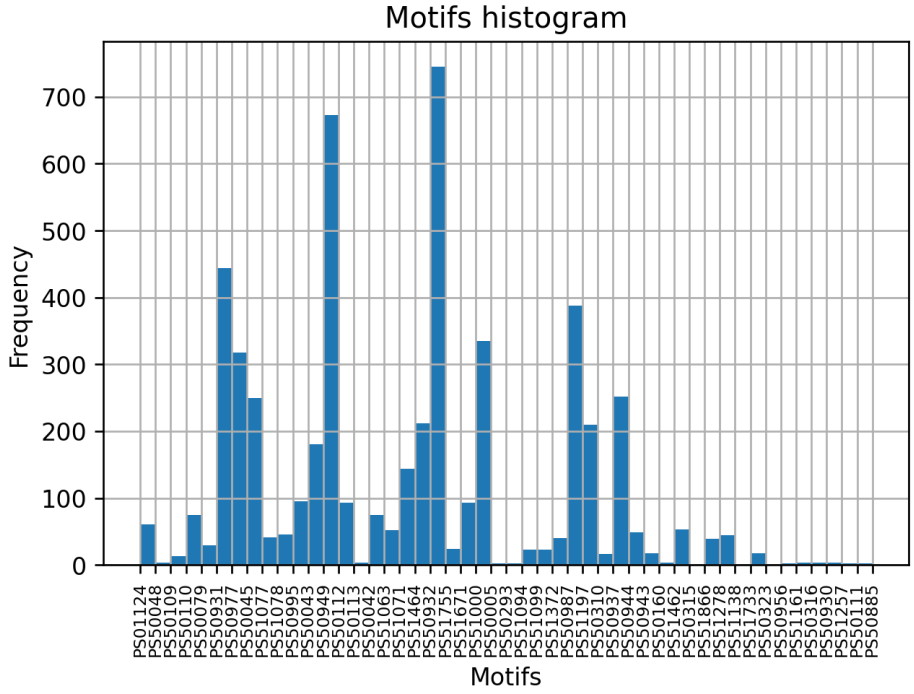


**Figure 4.3:** Frequency of occurrence of each motif found among the initial sequences in the database

## 4.2 Data preparation as inputs for the neural network

On the one hand, sequences' matrices were constructed as described in Section 3.2.2. Since different options were considered, the generated inputs have different sizes. These are represented in Table 4.1.

To understand them, it is necessary to take into account several aspects. Regarding the number of observations, as initially just sequences from the original database will be used to train the model, 5438 sequences are available. Since negative cases were included as described in Section 3.2.4, the number of observations is doubled. About the number of columns, zero padding was applied to make all sequences of the same length and the largest sequence was made of 978 amino acids. The latter does not apply to the last case.

**Table 4.1:** Matrices shapes for each of the inputs considered. Shapes are represented as number of (observations, rows, columns).

| TYPE OF INPUT | SHAPE |
|---|---|
| All amino acids (no motifs) | (10876, 20, 978) |
| Amino acids grouped (no motifs) | (10876, 5, 978) |
| Each motif in a row (all aa) | (10876, 69, 978) |
| Motifs in sequence (all aa) | (10876, 20, 978) |
| Just motifs (all aa) | (10876, 20, 255) |

It is in the number of rows where differences appear. The first case is the base one, where motifs are not included and all the information about the amino acids is represented. As there are 20 amino acids, this is the number of rows for the matrices. When amino acids are clustered as described in Section 3.2.2, 5 groups are formed in the second case.

Then, the addition of motifs is considered. As seen above in Section 4.1.3, 49 different motifs were found. When a row is added for each motif, the number increases to 69 rows. Meanwhile, if motifs information is added in the sequence itself, there are not shape modifications. Lastly, if just motifs are introduced, it is the length of the sequences that is modified, having as many columns as the number of amino acids of the longest motif.

Note that this input is 3-dimensional, since the matrices for each sequence are stored together.

On the other hand, in Section 3.2.3 was described how molecules have to be converted into inputs. The fingerprints generated are also placed in a matrix, where columns represent the bits with the information about the characteristics of the molecules and in rows each molecule fingerprint is introduced. In this way, there are as many rows as observations (10876), while the number of columns depends on the number of bits decided to create the fingerprints (2048 and 512 bits were considered).

Finally, labels indicating whether the TF is valid to sense the molecule or not are set in a single array of 10876 observations.

Bearing in mind the amount of observations and the fact that inputs have up to three dimensions, it is to be expected that computing times during the training will not be short. Taking into consideration that Cross-Validation is performed too, a training lasts up to one hour.

## 4.3 Development of the predictive model

In this section, exploration through the different possibilities the model provides will be described. These encompasses the different types of inputs seen, the content of each layer of the network and the value of the hyperparameters.

To check the performance of the the model in each possibility, a Cross-Validation process (described in Section 3.2.5) is carried out. The number of folds selected ($k$) is 5, so each model is validated five times.

The metrics used during the validation are the loss function and the accuracy. The loss function selected is the binary cross entropy, as the labels represent a binary classification: whether the pair sequence - molecule is valid or not.

### *4.3.1 Best inputs*

The first input tried was the one with all amino acids represented and the motifs placed each in a row (option 1 in Section 3.2.2). The reason why it was first used is that this represents all the possible information, as all amino acids are set and motifs are differentiated between them.

Later, in an attempt to reduce the matrix size to make things easier to the model, the motifs were marked in each sequence (option 2 in Section 3.2.2), losing its differentiation but winning importance about its place in the sequence. However, no significant improvement was observed.

At this point, the quality of information given by motifs was questioned, so they were removed from the matrix (option 3 in Section 3.2.2), leaving just the raw sequences. Again, no changes were perceived in the model accuracy. Therefore, it could be concluded that motifs do not give any relevant information to the model that could be used to ensure a better performance.

Despite these findings, introducing just the motifs sub-sequences was tried. In this way, matrix size is reduced and the model can be focused on the information about sequences function. Nevertheless, the model performance was even worse, so it was conclude that it is necessary to have the entire sequence and the irrelevance of motifs role was confirmed.

Another aspect considered when constructing the sequences matrix was whether to use all amino acids or to group them as explained in Section 3.2.2. Both options were tried with no significant changes, so it was decided to conserve all amino acids since the information remains complete.

In relation with the molecules input, two decisions had to be taken. The first one refers to the number of bits used to obtain the fingerprints and it is described in Section 3.2.3. Initially, this was set to 2048 bits as it was expected to provide more information about molecules characteristics. However, most of the fingerprints were plenty of zeros. So, to simplify the inputs, fingerprints were rearranged to 512 bits. This was a determinant decision, since the model accuracy went from poor levels around 56% to better results of around 73% of accuracy.

The second issue to be discussed is described in Section 3.2.4 and it is related to the selection of molecules for the negative cases. When chosen randomly, the results were not bad at all, as accuracy levels remained unchanged. Nevertheless, when the Tanimoto index was used to select particularly different molecules, performance increased up to 84%.

Note that the percentages given are not the final metrics which describe the model performance, because the discussion on parameter values has yet to be carried out (Sections 4.3.2 and 4.3.3).

To sum up, the inputs finally used are:

- Input 1: sequence matrices with a row for each of the 20 amino acids and the information about motifs not included.
- Input 2: fingerprints matrix with 512 bits and negative cases selected by little similarity.

### *4.3.2 Units of the layers*

When describing the parameter selection in Section 3.2.6, several restrictions were made about the number of units that each layer of the network architecture should have. Using the bayesian optimisation algorithm mentioned in the same Section and through several validation process, the units that best fit the model were decided.

In Figure 4.4, the units selected for each layer are represented. The term "output shape" makes reference to the fact that each unit generates a weight to be used by the neural network during the training, so the number of units represents the number of weights generated at the output of that layer.

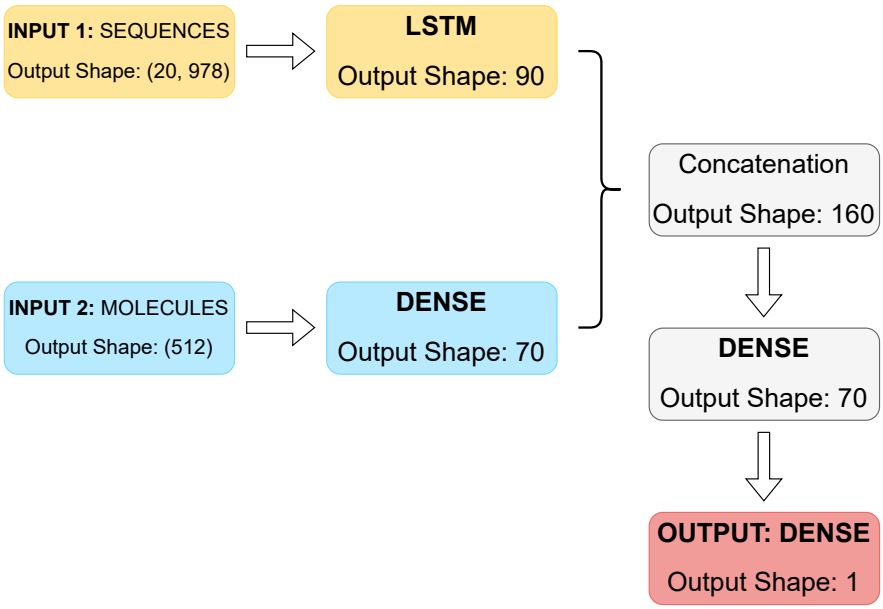


**Figure 4.4:** Network architecture. The number of units selected for each layer is represented as the shape of the output of the layer

One of the most relevant aspects here is the units of the first Dense layer, in the molecules branch (blue). It was determinant for a better model performance to select a lower number than the selected for the sequences branch (yellow). In that way, it was ensured that the network focused on learning about the sequential naturalness of the first input.

### *4.3.3 Network hyperparameters*

In Section 3.2.6 was established that three hyperparameters would be modified in order to obtain a good model performance. The final values selected for each one are here indicated, as well as the reasoning that has led to their election.

- Learning rate: 0.0001

  This is indicated to the optimisation algorithm. As explained in Section 3.2.6, Adam was used. The learning rate indicates the magnitude in which the gradient descent method corrects the network weights, so when greater values were used the model converged too quickly and the optimal solution was not achieved. On the contrary, when the learning rate was too small, the method got stuck in a local minimum and results did not change over iterations.

  The value selected finds a balance between this two situations and achieves an optimal result.

- Batch size: 50

  As it refers to the number of observations taken in each iteration, the batch size value compromises the speed and accuracy of the training. When lower, more iterations are required, so performance is better and the training time is longer. Oppositely, if it is larger it takes less iterations, so it is faster to train the model but it is more difficult to obtain an optimal accuracy.

  After several validation processes testing values, it resulted that batches of 50 samples balanced the training speed and the model accuracy.

- Epochs: 30

  It is important to select an appropriate number of epochs, not only for the computing time, but for the model performance. A low value might lead to a sub-optimal solution, while a very large value can result in over-training. The latter situation refers to the case where the model has seen the training data so many times that it overfits them.

  30 epochs have been enough to obtain acceptable accuracies while avoiding overfitting.

### *4.3.4 Performance evolution*

At the beginning of this section it has been indicated that loss and accuracy are the metrics used to evaluate the models. Both values have been studied closely during all the possibilities of the model tried. The last results obtained during the validation process, which describe the model performance when the training is carried out taking into consideration the decisions made in Section 4.2 and previously in this one, are shown in Table 4.2.

**Table 4.2:** Loss and accuracy scores obtained during the definitive validation process of the model. Folds make reference to the cross validation technique carried out

| FOLD NUMBER | LOSS | ACCURACY |
|---|---|---|
| Fold 1 | 0.460 | 89.31 % |
| Fold 2 | 0.331 | 89.43 % |
| Fold 3 | 0.505 | 83.97 % |
| Fold 4 | 0.399 | 88.56 % |
| Fold 5 | 0.297 | 89.77 % |
| Average | 0.399 | 88.21 % |

Not only its final value at the end of each attempt carried out, but also the accuracy and loss evaluation during the training process has been observed. Its visual analysis provides information about how the network is performing the learning, so undesirable events as overfitting or artifacts can be detected.

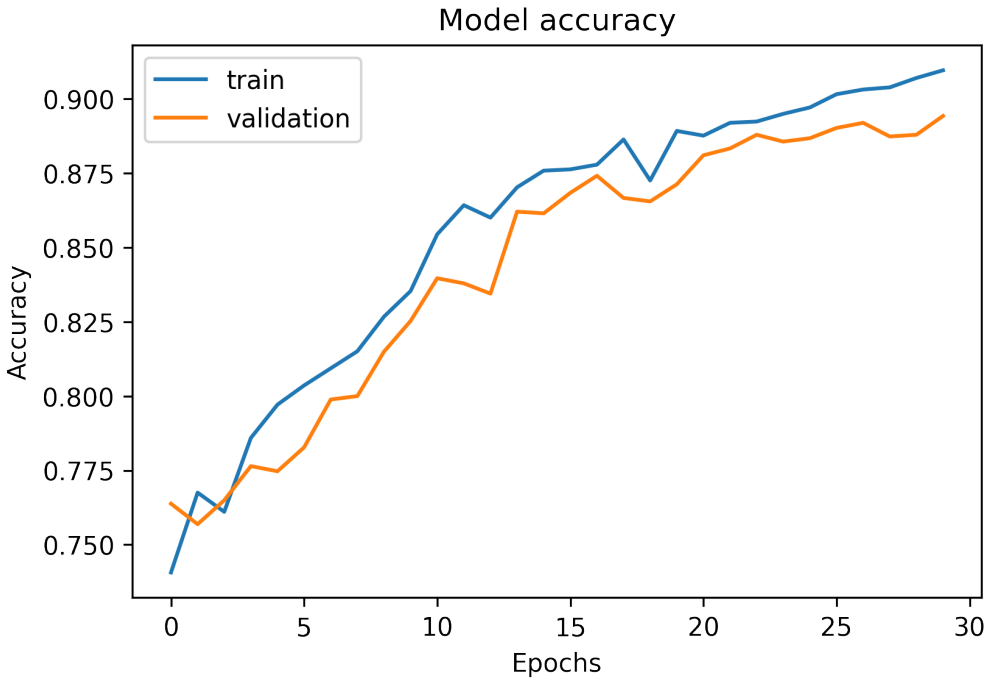


**Figure 4.5:** Example of accuracy curve for one of the validation processes of the final model

In Figure 4.5 a representative example (taken from one of the folds during the cross validation process) of the evolution of the accuracy curve is shown. As expected, a growth of the curve can be observed as the number of epochs increases, and then it stabilises. This effect demonstrates that the number of epochs is enough and, since the stabilisation is not long, it also proves that it is not too large.

Moreover, the validation curve is under the train one. This is a good sign because an overfitted model would not produce this effect as clearly.

Lastly, in Figure 4.6 the loss curve is represented. The fact that the validation curve goes over the train one could be disturbing as it could mean a case of overfitting. However, taking into account the discussion made above about the accuracy curve and the fact that loss curves stabilise (an actual case of over-training would produce a late rise in curves), overfitting can be discarded.

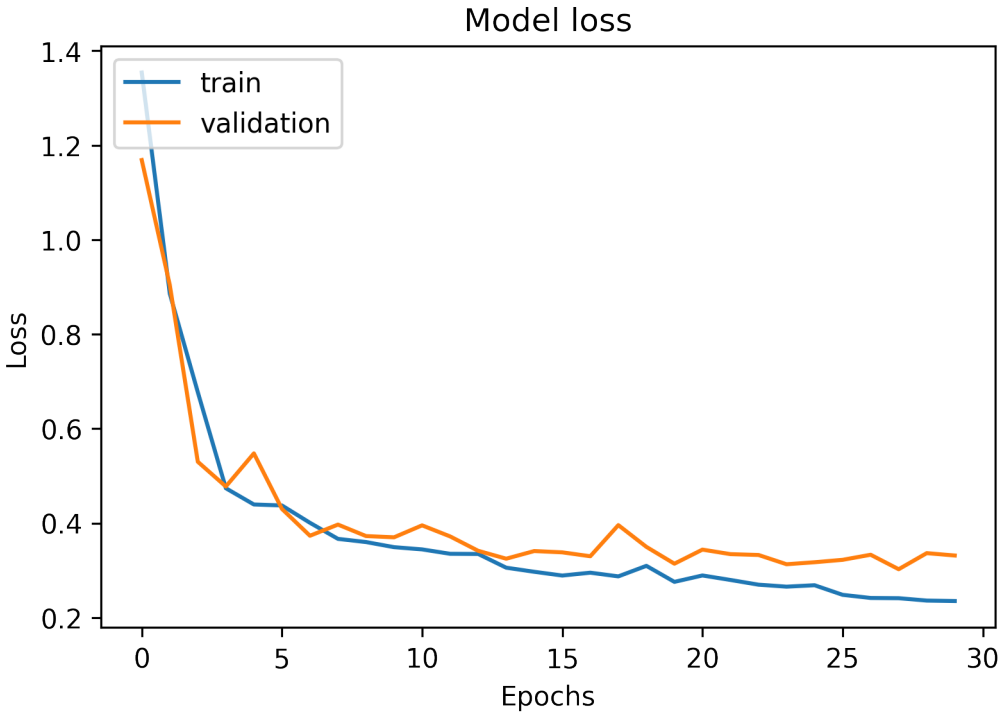


**Figure 4.6:** Example of loss curve for one of the validation processes of the final model

Therefore, bearing in mind the scores obtained by each of the validation process and the performance evolution, which demonstrates there is no overfitting and neither apparent artefacts, it can be concluded that the model validation has been carried out successfully.

## 4.4 Model testing

Now that the best parameters for the model have been selected and properly validated, the actual training is carried out. It is important to distinguish between the validation process and the training process. While the former is used to train several model possibilities and to explore which parameters best fit the problem, the latter is used to train the model when all its aspects have been decided. The difference lies in the fact that the train data is not split in folds the second time: the entire set is used for training.

Once the model is fully developed and trained, it is time to test its actual performance. As it was explained in Section 3.2.5, data was split in order to preserve some observations that have never been seen by the model. Consequently, there is no way the model could return its predictions based on a previous learning of that concrete data, so these results are a reliable representation of how the model would actually work with new data.

### *4.4.1 Accuracy and loss*

The scores obtained when predictions were made with the test data have been **0.323 of loss and 89.84% of accuracy**. As expected, these values are around the average validation scores (Table 4.2). This leaves an error of slightly more than 10% of the cases, which is considered to be a desirable performance.

### *4.4.2 The ROC curve and the AUC value*

Another aspect considered to evaluate the model performance has been the Receiver Operating Characteristic (ROC) curve. This is used to evaluate a binary classification such as the one in question (whether there is affinity between the TF and the molecule or not). To understand its meaning, some statistical terms must be defined first.

- Recall: it is the model's ability to find all positive cases. That is to say, the recall represents how many positive cases have been correctly labeled among all the positive cases. It is calculated by the formula TP/(TP+FN). It is also known as the true positive rate.
- Specificity: it is the model's ability to find all negative cases. That is to say, the specificity represents how many negative cases have been correctly labeled among all the negative cases. It is calculated by the formula TN/(TN+FP). It is also known as the true negative rate.

It is also important to bear in mind what the model is intended to do with its training. The main objective is to distinguish as best as possible between two populations of data, in this case TF - molecule affinity or not. If the model was totally inefficient, the two populations would be completely superimposed. On the contrary, if the model was ideal, there would be no overlap between them.

When trying to classify a new observation, a threshold has to be defined to decide to which population it belongs in case it is in the overlapping section. Depending on where the threshold is located, the recall and specificity will change its values when testing the model.

The relation between the values of this two metrics according to the threshold location is that the ROC curve represents. However, instead of using specificity, it uses its complementary value. In this way, the ROC curve is a graph that plots the true positive rate against the false positive rate.

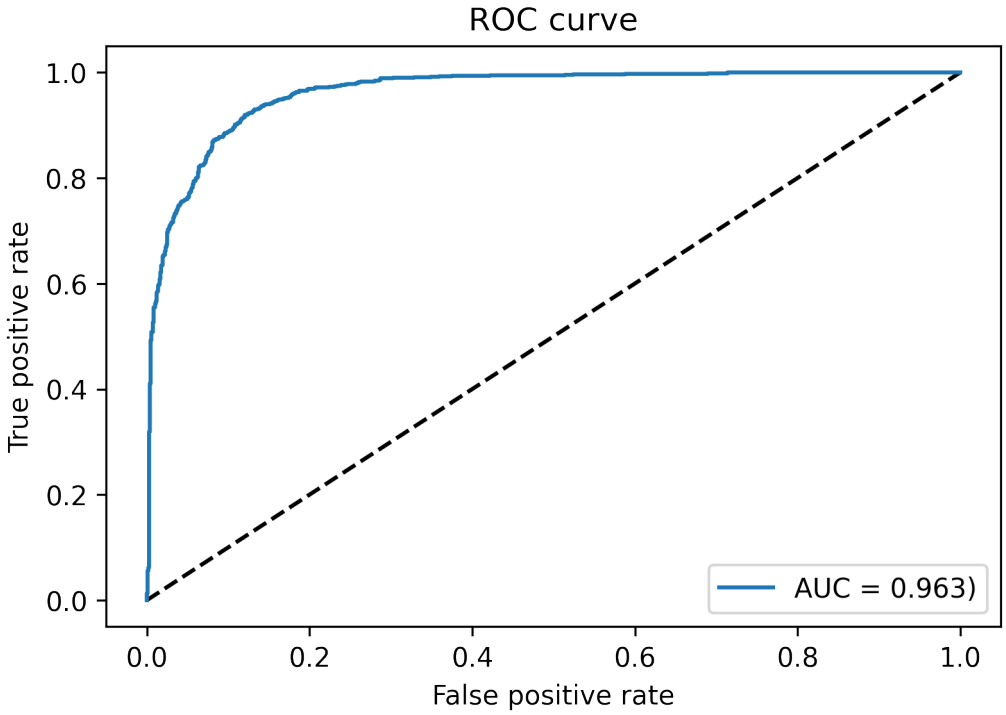


**Figure 4.7:** ROC curve resulting from test and its AUC value

In the worst of cases, the ROC curve would look as the striped line in Figure 4.7, whereas if the model performs well, the curve departs from the diagonal. The actual ROC curve obtained when using the test data to make predictions is shown in the same figure. As it can be observed, the recall increases rapidly, so the model is able to distinguish between the two groups.

The AUC value (area under the curve) is useful too. It ranges from 0.5 to 1 and the higher it is the better the performance of the model. In this case, its value is 0.963, which confirms that the model has been properly trained.

### *4.4.3 Group comparison*

Finally, other metrics have been evaluated to explore how well the model performs concretely with each of the groups. It is valuable information in order to find biases between classes. The metrics extracted have been:

- Precision: it is the model's ability to not label positive cases as negative. That is to say, the precision represents how many positive cases have been correctly labeled among all the observations predicted as positive. It is calculated by the formula TP/(TP+FP).
- Recall: already explained in Section 4.4.2.
- F1-score: it is is the harmonic mean of precision and recall.

The values obtained are shown in Table 4.3. As it can be observed, all of them are quite similar for both groups, specially the F1-score. This fact leads to the conclusion that the model is well balanced for predicting either the affinity between a TF and a molecule or the impossibility of using the TF to sense the molecule.

**Table 4.3:** Precision, recall and F1-score for prediction with the test data. The terms "no match" and "match" refer to whether the pair TF - molecule was supposed to have a negative label (0) or a positive one (1)

| TF - molecule | Precision | Recall | F1-score |
|---|---|---|---|
| No match | 0.92 | 0.87 | 0.89 |
| Match | 0.88 | 0.93 | 0.9 |

# Chapter 5

# Conclusions

*In this last chapter, several conclusions found during the development of this Project will be presented. In addition, the accomplishment of the objectives explained in Section 1 will be reviewed. Finally, future lines of work will be considered.*

## 5.1 Convenience of using deep learning techniques

The choice of using deep learning methods instead of other machine learning techniques has proved to be a success. Here, some aspects of the processes carried out in relation with the model development will be discussed.

### *5.1.1 Deep learning abilities*

One of the most important difficulties found during this Project was the fact of working with sequences. Whether with the use of bioinformatic tools for the data augmentation or with the preparation of inputs for the model, the long sequences and the large number of them were something difficult to work with.

Along the same lines, the information the TFs have in order to relate them with the metabolites that they can sense is highly abstract. Therefore, if feature characterisation had been necessary, it would have been an added difficulty.

Thanks to deep learning techniques, this process was not required. The neural network ability to extract the characteristics that better fit the model objective was of great help in order to develop a well-performing model. Also, it has been able to find data relationships which are perhaps currently unattainable for human understanding.

It should be noted that this was possible thanks to the fact that enough data observations were available.

### *5.1.2 Importance of simplicity in the architecture*

When the model architecture was described in Section 3.2.1, it was mentioned the fact that only the necessary layers were used to keep it as simple as possible. This has resulted to be vitally important.

Thanks to the structure simplicity, the model was able to learn the required features avoiding to learn too much of the data, so there was not overfitting. Furthermore, it has led to relatively short computing times. Although each validation process needed almost an hour to be complete, the final model training lasted no more than fifteen minutes. Consequently, predictions with new data are carried out in reasonable periods of time.

Actually, other architectures with more layers were tested. However, there was not a significant improvement in the performance and it took more time to train the models.

### *5.1.3 Importance of the validation process*

It has also been considered important to highlight how the validation processes carried out to find the best model were of great help in order to achieve a good performance and, most importantly, to avoid false results.

Thanks to the different tests carried out with different groups of data, it could be detected when a model configuration was acceptable for a concrete fold but not for the rest. Or vice versa, which is a more dangerous situation if some results are presumed as correct. When the average accuracy seemed to be proper, the validation process enabled to check if there was any fold with poor performance, which would have meant that the model was wrongly trained.

## 5.2 Importance of data codification

The most relevant aspect in order to achieve an acceptable performance of the model was the way in which the data was prepared to be used as inputs for the network. Although the parameters selection and the network architecture were important to achieve better results, the main key was the data codification.

### *5.2.1 Simplicity in input codification*

The philosophy followed when preparing data as inputs was to keep them as simple as possible. For instance, this is the reason why one-hot encoding was used with sequences, since it can represent them in binary form.

However, it was the molecules codification that finally was of most importance. As explained in Sections 3.2.3 and 4.3.1, first the default size for fingerprints was used. When training models with fingerprints of 2048 bits, the performance results were incredibly poor. It was the size reduction that let the model to obtain better accuracies.

Until this fact was discovered, no change on the model parameters were useful to enable the model learn about the relationships between sequences and molecules. This is therefore a representative example of the importance of proper data preparation.

### *5.2.2 Relevance of motifs*

At the beginning of this Project it was thought that motifs would be important in order to facilitate the network's extraction of protein function characteristics. However, as seen in Section 4.3.1, this was not the case. Two possible explanations can be concluded from this fact:

- Motifs were not correctly codified. That is to say, the methodologies applied (which were described in Section 3.2.2) were not able to extract the motifs information from the sequences in a useful way for the neural network.
- Motifs are not vitally important in this field. Although they theoretically contain the information about the protein function, these sub-sequences may not be of great relevance for predicting the affinity between a protein and a molecule. In other words, it is more important to consider the entire TF sequence to find which molecules it can sense.

Anyway, not using the motifs information has not been an impediment to achieve a well-performing model.

### *5.2.3 Class balance*

Another aspect that often poses difficulties in deep learning classification problems is the class balance, that is to say, the balance between the number of observations of each group.

Luckily, as the negative cases were created manually (as explained in Section 3.2.4), it was possible to have exactly the same number of positive and negative observations. This turned out to be important to avoid biases between classes when new predictions are performed, as the model has no preferences when classifying the new observation. For example, if the available data had been unbalanced in favour of the positive cases, the model would tend to predict new cases as positive, resulting in false positives.

An evidence that this problem has been avoided can be observed in Table 4.3, where metrics for both classes are similar and acceptable.

## 5.3 Final results

Taking into account the different evaluations carried out about the model performance, it can be concluded that it has been developed successfully.

The performance evolution during the final validation process demonstrates a properly training. As seen in Section 4.3.4, the values of loss and accuracy in every fold were similar, so there are not any specific data to which the model is overfitted. Also, the discussion carried out about how these values changed over the epochs (Figures 4.5 and 4.6) leads to the conclusion that there are not artefacts that may lead to wrong predictions.

Moreover, the results from using the test data were also good enough. The accuracy and loss values from the testing process were similar to those obtained while the validation process. They are a little better, perhaps because of the fact that more observation were used when training (train data is not split to obtain validation data). The ROC curve and the AUC value proved good performance too when distinguishing observations of each class. Lastly, the group comparison demonstrated a balanced model, as mentioned above.

## 5.4 Objectives accomplishment

In this section, a review of the objectives set out at the beginning of this report will be performed, checking if they have been fulfilled successfully.

1. **Collection of data from TF based biosensors.** A series of bioinformatic tools have been used during the data augmentation and the search of sequence information. These were used with success, collecting experimental data that could have been useful. Even if neither the homologous found nor the motifs of the sequences have been used finally for this model, it has been an enriching and formative experience.
2. **Use of machine learning algorithms for predictive model development.** As the results shown in Section 4.4 are satisfactory, it could be concluded that this objective has been fulfilled successfully. The predictive model is ready to be used with new data.
3. **Biosensor open source library development.** With the implementation of the model in the Sensbio web server, this tool is ready to generate biosensor libraries that can be useful for dynamic regulation of bioproduction pathways. Moreover, the results obtained in the web server when introducing new proteins or molecules are apparently reasonable: the prediction scores range between 0 and 1 with different values for each comparison.

## 5.5 Future lines

There are some tasks that could be carried out in the future if this Project were to be continued. Some of them are exposed here:

- Use of the data augmentation performed. Although it was not necessary for obtaining acceptable performance values, the neural network could be trained with all the data collected from the homologous search and its clustering. In this way, the model would have a wider range of work and possibly the quality of its predictions would increase when using new data. Computing times and resources limitation were the reasons for not further exploring the use of this data set at present.
- Model updating. As new coincidences between TF and metabolites to be sensed are discovered continuously, keeping up to date the model is a necessary task. This would ensure the quality of predictions, avoiding leaving the model useless.
- Experimental testing. This comprises the evaluation of new TFs based biosensors predicted by the model. If predictions are tested in the laboratory, the model quality could be confirmed.

# Bibliography


Albawi, Saad, Tareq Abed Mohammed, and Saad Al-Zawi (2017). "Understanding of a convolutional neural network". In: *2017 international conference on engineering and technology (ICET)*. Ieee, pp. 1–6 (cit. on p. 14).

Altman, Naomi S (1992). "An introduction to kernel and nearest-neighbor nonparametric regression". In: *The American Statistician* 46.3, pp. 175–185 (cit. on p. 12).

Altschul, Stephen F. et al. (1990). "Basic local alignment search tool". In: *Journal of Molecular Biology* 215.3, pp. 403–410. ISSN: 0022-2836. DOI: https://doi.org/10.1016/S0022-2836(05)80360-2 (cit. on p. 18).

Ashburner, Michael et al. (2000). "Gene ontology: tool for the unification of biology". In: *Nature genetics* 25.1, pp. 25–29 (cit. on p. 10).

Bajorath, Jürgen (2001). "Selected concepts and investigations in compound classification, molecular descriptor analysis, and virtual screening". In: *Journal of chemical information and computer sciences* 41.2, pp. 233–245 (cit. on p. 23).

Bajusz, Dávid, Anita Rácz, and Károly Héberger (2015). "Why is Tanimoto index an appropriate choice for fingerprint-based similarity calculations?" In: *Journal of cheminformatics* 7.1, pp. 1–13 (cit. on p. 24).

Baumann, Leonie et al. (2018). "A yeast-based biosensor for screening of short-and medium-chain fatty acid production". In: *ACS synthetic biology* 7.11, pp. 2640–2646 (cit. on p. 9).

Binder, Stephan et al. (2012). "A high-throughput approach to identify genomic variants of bacterial metabolite producers at the single-cell level". In: *Genome biology* 13.5, pp. 1–12 (cit. on p. 9).

Bonetta, Rosalin and Gianluca Valentino (2020). "Machine learning techniques for protein function prediction". In: *Proteins: Structure, Function, and Bioinformatics* 88.3, pp. 397–413 (cit. on pp. 10, 11, 15).

Bork, Peer and Eugene V Koonin (1996). "Protein sequence motifs". In: *Current Opinion in Structural Biology* 6.3, pp. 366–376. ISSN: 0959-440X. DOI: https://doi.org/10.1016/S0959-440X(96)80057-1 (cit. on p. 19).

Boser, Bernhard E, Isabelle M Guyon, and Vladimir N Vapnik (1992). "A training algorithm for optimal margin classifiers". In: *Proceedings of the fifth annual workshop on Computational learning theory*, pp. 144–152 (cit. on p. 11).

Breiman, Leo (1996). "Bagging predictors". In: *Machine learning* 24.2, pp. 123–140 (cit. on p. 12).

— (2001). "Random forests". In: *Machine learning* 45.1, pp. 5–32 (cit. on p. 12).

Cai, CZ et al. (2003). "SVM-Prot: web-based support vector machine software for functional classification of a protein from its primary sequence". In: *Nucleic acids research* 31.13, pp. 3692–3697 (cit. on p. 12).

Campos Merschmann, Luiz Henrique de and Alex Alves Freitas (2013). "An extended local hierarchical classifier for prediction of protein and gene functions". In: *International Conference on Data Warehousing and Knowledge Discovery*. Springer, pp. 159–171 (cit. on p. 12).

Chapman, Brad and Jeffrey Chang (Aug. 2000). "Biopython Python Tools for Computational Biology". In: *SIGBIO Newsl.* 20.2, pp. 15–19. ISSN: 0163-5697. DOI: 10.1145360262.360268 (cit. on p. 18).

Chen, Xue-Feng et al. (2018). "Engineering tunable biosensors for monitoring putrescine in Escherichia coli". In: *Biotechnology and bioengineering* 115.4, pp. 1014–1027 (cit. on p. 9).

De Ferrari, Luna and John BO Mitchell (2014). "From sequence to enzyme mechanism using multi-label machine learning". In: *BMC bioinformatics* 15.1, pp. 1–13 (cit. on p. 12).

Delépine, Baudoin et al. (2016). "SensiPath: computer-aided design of sensing-enabling metabolic pathways". In: *Nucleic acids research* 44.W1, W226–W231 (cit. on p. 9).

Ding, Nana, Shenghu Zhou, and Yu Deng (2021). "Transcription-Factor-based Biosensor Engineering for Applications in Synthetic Biology". In: *ACS Synthetic Biology* 10.5. PMID: 33899477, pp. 911–922. DOI: 10.1021/acssynbio.0c00252. eprint: https://doi.org/10.1021/acssynbio.0c00252 (cit. on p. 9).

Eisenberg, David et al. (2000). "Protein function in the post-genomic era". In: *Nature* 405.6788, pp. 823–826 (cit. on p. 10).

Freund, Yoav, Robert E Schapire, et al. (1996). "Experiments with a new boosting algorithm". In: *icml*. Vol. 96. Citeseer, pp. 148–156 (cit. on p. 12).

Gattiker, Alexandre, Elisabeth Gasteiger, and Amos Marc Bairoch (2002). "ScanProsite: a reference implementation of a PROSITE scanning tool". In: *Applied bioinformatics* 1.2, pp. 107–8 (cit. on p. 19).

Gligorijević, Vladimir, Meet Barot, and Richard Bonneau (2018). "deepNF: deep network fusion for protein function prediction". In: *Bioinformatics* 34.22, pp. 3873–3881 (cit. on p. 14).

Hara, Kazuyuki, Daisuke Saito, and Hayaru Shouno (2015). "Analysis of function of rectified linear unit used in deep learning". In: *2015 international joint conference on neural networks (IJCNN)*. IEEE, pp. 1–8 (cit. on p. 26).

Hochreiter, Sepp and Jürgen Schmidhuber (1997). "Long Short-Term Memory". In: *Neural Computation* 9.8, pp. 1735–1780. DOI: 10.1162/neco.1997.9.8.1735 (cit. on pp. 14, 21).

Hu, Lele et al. (2011). "Predicting functions of proteins in mouse based on weighted protein-protein interaction network and protein hybrid properties". In: *PloS one* 6.1, e14556 (cit. on p. 12).

Huang, Ying et al. (Jan. 2010). "CD-HIT Suite: a web server for clustering and comparing biological sequences". In: *Bioinformatics* 26.5, pp. 680–682. ISSN: 1367-4803. DOI: 10.1093/bioinformatics/btq003. eprint: https://academic.oup.com/bioinformatics/article-pdf/26/5/680/28355559/btq003.pdf (cit. on p. 19).

Hulo, Nicolas et al. (Jan. 2006). "The PROSITE database". In: *Nucleic Acids Research* 34.suppl_1, pp. D227–D230. ISSN: 0305-1048. DOI: 10.1093/nar/gkj063. eprint: https://academic.oup.com/nar/article-pdf/34/suppl_1/D227/3924681/gkj063.pdf (cit. on p. 19).

Jalili, Vahid et al. (June 2020). "The Galaxy platform for accessible, reproducible and collaborative biomedical analyses: 2020 update". In: *Nucleic Acids Research* 48.W1, W395–W402. ISSN: 0305-1048. DOI: 10.1093/nar/gkaa434. eprint: https://academic.oup.com/nar/article-pdf/48/W1/W395/33446235/gkaa434.pdf (cit. on p. 18).

Kikuchi, Kazuya, Hideo Takakusa, and Tetsuo Nagano (2004). "Recent advances in the design of small molecule-based FRET sensors for cell biology". In: *TrAC Trends in Analytical Chemistry* 23.6, pp. 407–415. ISSN: 0165-9936. DOI: https://doi.org/10.1016/S0165-9936(04)00608-9 (cit. on p. 7).

Kingma, Diederik P and Jimmy Ba (2014). "Adam: A method for stochastic optimization". In: *arXiv preprint arXiv:1412.6980* (cit. on p. 26).

Kleinbaum, David G et al. (2002). *Logistic regression*. Springer (cit. on p. 12).

Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E Hinton (2012). "Imagenet classification with deep convolutional neural networks". In: *Advances in neural information processing systems* 25 (cit. on p. 14).

Kulmanov, Maxat, Mohammed Asif Khan, and Robert Hoehndorf (Oct. 2017). "DeepGO: predicting protein functions from sequence and interactions using a deep ontology-aware classifier". In: *Bioinformatics* 34.4, pp. 660–668. ISSN: 1367-4803. DOI: 10.1093/bioinformatics/btx624. eprint: https://academic.oup.com/bioinformatics/article-pdf/34/4/660/25117339/btx624.pdf (cit. on pp. 13, 14).

Kumar, Chetan, Gang Li, and Alok Choudhary (2009). "Enzyme function classification using protein sequence features and random forest". In: *2009 3rd International Conference on Bioinformatics and Biomedical Engineering*. IEEE, pp. 1–4 (cit. on p. 12).

Lanckriet, Gert RG et al. (2003). "Kernel-based data fusion and its application to protein function prediction in yeast". In: *Biocomputing 2004*. World Scientific, pp. 300–311 (cit. on p. 12).

Latchman, David S. (1997). "Transcription factors: An overview". In: *The International Journal of Biochemistry & Cell Biology* 29.12, pp. 1305–1312. ISSN: 1357-2725. DOI: https://doi.org/10.1016/S1357-2725(97)00085-X (cit. on p. 8).

Lee, Hyunju et al. (2006). "Diffusion kernel-based logistic regression models for protein function prediction". In: *Omics: a journal of integrative biology* 10.1, pp. 40–55 (cit. on p. 12).

Li, Fuyi et al. (2015). "GlycoMine: a machine learning-based approach for predicting N-, C-and O-linked glycosylation in the human proteome". In: *Bioinformatics* 31.9, pp. 1411–1419 (cit. on p. 12).

Li, Liangpo et al. (2019). "Development of a synthetic 3-dehydroshikimate biosensor in Escherichia coli for metabolite monitoring and genetic screening". In: *ACS Synthetic Biology* 8.2, pp. 297–306 (cit. on p. 9).

Li, Weizhong and Adam Godzik (May 2006). "Cd-hit: a fast program for clustering and comparing large sets of protein or nucleotide sequences". In: *Bioinformatics* 22.13, pp. 1658–1659. ISSN: 1367-4803. DOI: 10.1093/bioinformatics/btl158. eprint: https://academic.oup.com/bioinformatics/article-pdf/22/13/1658/484588/btl158.pdf (cit. on p. 19).

Libis, Vincent, Baudoin Delépine, and Jean-Loup Faulon (2016). "Expanding biosensing abilities through computer-aided design of metabolic pathways". In: *ACS synthetic biology* 5.10, pp. 1076–1085 (cit. on p. 9).

Liu, Shu-De et al. (2017). "Maltose utilization as a novel selection strategy for continuous evolution of microbes with enhanced metabolite production". In: *ACS synthetic biology* 6.12, pp. 2326–2338 (cit. on p. 9).

Liu, Xueliang (2017). "Deep recurrent neural network for protein function prediction from sequence". In: *arXiv preprint arXiv:1701.08318* (cit. on p. 14).

Mahr, Regina and Julia Frunzke (2016). "Transcription factor-based biosensors in biotechnology: current state and future prospects". In: *Applied microbiology and biotechnology* 100.1, pp. 79–90 (cit. on p. 9).

Mandal, Maumita and Ronald R Breaker (2004). "Gene regulation by riboswitches". In: *Nature reviews Molecular cell biology* 5.6, pp. 451–463 (cit. on p. 8).

Mayr, Lorenz M and Dejan Bojanic (2009). "Novel trends in high-throughput screening". In: *Current Opinion in Pharmacology* 9.5. Anti-infectives/New technologies, pp. 580–588. ISSN: 1471-4892. DOI: https://doi.org/10.1016/j.coph.2009.08.004 (cit. on p. 9).

Medsker, L.R and L.C Jain (1999). *Recurrent neural networks: design and applications*. Dutch. CRC Press (cit. on p. 14).

Mehrotra, Parikha (2016). "Biosensors and their applications–A review". In: *Journal of oral biology and craniofacial research* 6.2, pp. 153–159 (cit. on p. 7).

Michener, Joshua K. et al. (2012). "Applications of genetically-encoded biosensors for the construction and control of biosynthetic pathways". In: *Metabolic Engineering* 14.3. Synthetic Biology: New Methodologies and Applications for Metabolic Engineering, pp. 212–222. ISSN: 1096-7176. DOI: https://doi.org/10.1016/j.ymben.2011.09.004 (cit. on p. 7).

Ni, Qingshan et al. (2009). "Using logistic regression method to predict protein function from protein-protein interaction data". In: *2009 3rd International Conference on Bioinformatics and Biomedical Engineering*. IEEE, pp. 1–4 (cit. on p. 12).

Niu, Bing et al. (2006). "Predicting protein structural class with AdaBoost learner". In: *Protein and peptide letters* 13.5, pp. 489–492 (cit. on p. 12).

Nogueira, Fernando (2014). *Bayesian Optimization: Open source constrained global optimization tool for Python* (cit. on p. 26).

Pearson, W R and D J Lipman (1988). "Improved tools for biological sequence comparison." In: *Proceedings of the National Academy of Sciences* 85.8, pp. 2444–2448. DOI: 10.1073/pnas.85.8.2444. eprint: https://www.pnas.org/doi/pdf/10.1073/pnas.85.8.2444 (cit. on p. 18).

Refaeilzadeh, Payam, Lei Tang, and Huan Liu (2009). "Cross-validation." In: *Encyclopedia of database systems* 5, pp. 532–538 (cit. on p. 25).

Rish, Irina et al. (2001). "An empirical study of the naive Bayes classifier". In: *IJCAI 2001 workshop on empirical methods in artificial intelligence*. Vol. 3. 22, pp. 41–46 (cit. on p. 12).

Shao, Wei, Mingxia Liu, and Daoqiang Zhang (2016). "Human cell structure-driven model construction for predicting protein subcellular location from biological images". In: *Bioinformatics* 32.1, pp. 114–121 (cit. on p. 14).

Srivastava, Nitish et al. (2014). "Dropout: a simple way to prevent neural networks from overfitting". In: *The journal of machine learning research* 15.1, pp. 1929–1958 (cit. on p. 25).

Stella, Roberto G et al. (2019). "Evolutionary engineering of Corynebacterium glutamicum". In: *Biotechnology journal* 14.9, p. 1800444 (cit. on p. 9).

Su, Ran et al. (2021). "Protein subcellular localization based on deep image features and criterion learning strategy". In: *Briefings in Bioinformatics* 22.4, bbaa313 (cit. on p. 14).

Wan, Xia, Monireh Marsafari, and Peng Xu (2019). "Engineering metabolite-responsive transcriptional factors to sense small molecules in eukaryotes: current state and perspectives". In: *Microbial Cell Factories* 18.1, pp. 1–13 (cit. on pp. 7, 8).

Xu, Peng et al. (2014). "Improving fatty acids production by engineering dynamic pathway regulation and metabolic control". In: *Proceedings of the National Academy of Sciences* 111.31, pp. 11299–11304 (cit. on p. 9).

**Document II**

# Budget

# Chapter 1

# Budget breakdown

In this chapter, the different costs linked to the development of this Project are presented. For this purpose, the labour and materials costs are described first. Then, how they contribute to each development phase is explained.

## 1.1 Labour costs

In order to establish the salaries corresponding to the supervisor of this Project, the experimental director and the author (junior biomedical engineer), data shared by the UPV Human Resources Service have been used. The three roles have been considered to belong to the Teaching and Research Staff (PDI) of the UPV. Taking this into account and defining the most accurate profile for each one, an estimation of their annual salary was made. Then, the price per hour for each case was calculated considering that there are 1696 working hours per year.

The total cost from labour is presented in Table B.1.

**Table B.1:** Labour price table

| Labour table | | | | | |
|---|---|---|---|---|---|
| No. | Id. | Labour description | Price (€/h) | Hours (h) | Total (€) |
| 1 | L.01 | Supervisor labour | 20.39 | 26 | 530.14 |
| 2 | L.02 | Experimental director labour | 19.37 | 13 | 251.81 |
| 3 | L.03 | Junior biomedical engineer labour | 11.79 | 488 | 5753.52 |
| | | | | Total labour (€): | 6535.47 |

## 1.2 Materials costs

Here, the different materials that were necessary for the development of the Project are listed (Table B.2). even if many were free of charge, it has been considered useful to include them in the budget to describe how they have contributed to each phase.

**Table B.2:** Materials price table

| Material price table | | | | | |
|---|---|---|---|---|---|
| No. | Id. | Material description | Price (€/u) | Quantity (units) | Total (€) |
| 1 | MAT.01 | Microsoft 365 licence | 140.00 | 1 | 140.00 |
| 2 | MAT.02 | PC: Intel(R) Core(TM) i7, 8 GB RAM | 650.00 | 1 | 650.00 |
| 3 | MAT.03 | Anaconda (Python) licence | 0.00 | 1 | 0.00 |
| 4 | MAT.04 | Google Colab licence | 0.00 | 1 | 0.00 |
| 5 | MAT.05 | Overleaf (Latex) licence | 0.00 | 1 | 0.00 |
| | | | | Total materials (€): | 790.00 |

## 1.3 Unit costs

In Table B.3 the different phases carried out for the Project development are listed, including the steps followed. Each of these has an associated cost, which will be described in Section 1.4.

**Table B.3:** Unit prices table

| No. | Description | Price in numbers (€) | Price in words |
|---|---|---|---|
| 1 | Preliminary study | | |
| 1.1 | Project definition | 107.22 | One hundred and seven euros and twenty-two cents |
| 1.2 | Project planning | 214.45 | Two hundred and fourteen euros and forty-five cents |
| 1.3 | Investigation of the problem | 68.07 | Sixty-eight euros and seven cents |
| 2 | Data processing | | |
| 2.1 | Initial study of the database | 43.54 | Forty-three euros and fifty-four cents |
| 2.2 | Homologous search | 551.30 | Five hundred and fifty-one euros and thirty cents |
| 2.3 | Homologous clustering | 303.56 | Three hundred and three euros and fifty-six cents |
| 2.4 | Motifs search | 495.50 | Four hundred and ninety-five euros and fifty cents |
| 2.5 | Follow-up meeting | 99.34 | Ninety-nine euros and thirty-four cents |
| 3 | Model development | | |
| 3.1 | Programming language learning | 247.75 | Two hundred and forty-seven euros and seventy-five cents |
| 3.2 | Model planning | 68.07 | Sixty-eight euros and seven cents |
| 3.3 | Data codification | 439.69 | Four hundred and thirty-nine euros and sixty-nine cents |
| 3.4 | Model training | 1430.69 | One thousand four hundred and thirty euros and sixty-nine cents |
| 3.5 | Follow-up meeting | 33.47 | Thirty-three euros and forty-seven cents |
| 4 | Study of results | | |
| 4.1 | Models validation | 340.34 | Three hundred and forty euros and thirty-four cents |
| 4.2 | Model test | 55.81 | Fifty-five euros and eighty-one cents |
| 4.3 | Interpretation of results | 481.21 | Four hundred and eighty-one euros and twenty-one cents |
| 4.4 | Follow-up meeting | 37.70 | Thirty-seven euros and seventy cents |
| 5 | Model implementation | | |
| 5.1 | Web server implementation | 92.59 | Ninety-two euros and fifty-nine cents |
| 5.2 | Follow-up meeting | 88.21 | Eighty-eight euros and twenty-one cents |
| 6 | Documents writing | | |
| 6.1 | Project report writing | 1517.78 | One thousand five hundred seventeen euros and seventy-eight cents |
| 6.2 | Budget writing | 277.00 | Two hundred and seventy-seven euros |
| 6.3 | Follow-up meeting | 38.50 | Thirty-eight euros and fifty cents |

## 1.4 Decomposed prices

In this section, how each of the components listed in Tables B.1 (labour cost) and B.2 (materials cost) contribute to each of the project phases is described. This justifies the costs found in Table B.3 (unit costs). Tables from B.4 to B.9 decompose the prices of each phase.

It should be mentioned that the percentage considered for the indirect costs has been a 4%. Also, the use of each material has been divided proportionally in each phase.

**Table B.4:** Phase 1 price table

| No. | Id. | Unit | Description | Quantity | Price per unit of resource (€) | Total (€) |
|---|---|---|---|---|---|---|
| 1 | 1 | | **Preliminary study** | | | |
| 1.1 | 01.01 | U | Project definition | | | |
| | L.01 | h | Supervisor labour | 2 | 20.39 | 40.78 |
| | L.02 | h | Experimental director labour | 2 | 19.37 | 38.74 |
| | L.03 | h | Junior biomedical engineer labour | 2 | 11.79 | 23.58 |
| | | % | Indirect costs | 4 | 103.10 | 4.12 |
| | | | | | Total price per U (€): | 107.22 |
| 1.2 | 01.02 | U | Project planning | | | |
| | L.01 | h | Supervisor labour | 4 | 20.39 | 81.56 |
| | L.02 | h | Experimental director labour | 4 | 19.37 | 77.48 |
| | L.03 | h | Junior biomedical engineer labour | 4 | 11.79 | 47.16 |
| | | % | Indirect costs | 4 | 206.20 | 8.25 |
| | | | | | Total price per U (€): | 214.45 |
| 1.3 | 01.03 | U | Investigation of the problem | | | |
| | L.03 | h | Junior biomedical engineer labour | 5 | 11.79 | 58.95 |
| | MAT.02 | u | PC: Intel(R) Core(TM) i7, 8 GB RAM | 0.01 | 650.00 | 6.50 |
| | | % | Indirect costs | 4 | 65.45 | 2.62 |
| | | | | | Total price per U (€): | 68.07 |

**Table B.5:** Phase 2 price table

| No. | Id. | Unit | Description | Quantity | Price per unit of resource (€) | Total (€) |
|---|---|---|---|---|---|---|
| 2 | 2 | | **Data processing** | | | |
| 2.1 | 02.01 | U | Initial study of the database | | | |
| | L.03 | h | Junior biomedical engineer labour | 3 | 11.79 | 35.37 |
| | MAT.03 | u | Anaconda (Python) licence | 0.1 | 0.00 | 0.00 |
| | MAT.02 | u | PC: Intel(R) Core(TM) i7, 8 GB RAM | 0.01 | 650.00 | 6.50 |
| | | % | Indirect costs | 4 | 41.87 | 1.67 |
| | | | | | | |
| | | | | | Total price per U (€): | 43.54 |
| | | | | | | |
| 2.2 | 02.02 | U | Homologous search | | | |
| | L.03 | h | Junior biomedical engineer labour | 40 | 11.79 | 471.60 |
| | MAT.03 | u | Anaconda (Python) licence | 0.025 | 0.00 | 0.00 |
| | MAT.02 | u | PC: Intel(R) Core(TM) i7, 8 GB RAM | 0.09 | 650.00 | 58.50 |
| | | % | Indirect costs | 4 | 530.10 | 21.20 |
| | | | | | | |
| | | | | | Total price per U (€): | 551.30 |
| | | | | | | |
| 2.3 | 02.03 | U | Homologous clustering | | | |
| | L.03 | h | Junior biomedical engineer labour | 22 | 11.79 | 259.38 |
| | MAT.02 | u | PC: Intel(R) Core(TM) i7, 8 GB RAM | 0.05 | 650.00 | 32.50 |
| | | % | Indirect costs | 4 | 291.88 | 11.68 |
| | | | | | | |
| | | | | | Total price per U (€): | 303.56 |
| | | | | | | |
| 2.4 | 02.04 | U | Motifs search | | | |
| | L.03 | h | Junior biomedical engineer labour | 36 | 11.79 | 424.44 |
| | MAT.03 | u | Anaconda (Python) licence | 0.1 | 0.00 | 0.00 |
| | MAT.02 | u | PC: Intel(R) Core(TM) i7, 8 GB RAM | 0.08 | 650.00 | 52.00 |
| | | % | Indirect costs | 4 | 476.44 | 19.06 |
| | | | | | | |
| | | | | | Total price per U (€): | 495.50 |
| | | | | | | |
| 2.5 | 02.05 | U | Follow-up meeting | | | |
| | L.01 | h | Supervisor labour | 2 | 20.39 | 40.78 |
| | L.02 | h | Experimental director labour | 1 | 19.37 | 19.37 |
| | L.03 | h | Junior biomedical engineer labour | 3 | 11.79 | 35.37 |
| | | % | Indirect costs | 4 | 95.52 | 3.82 |
| | | | | | | |
| | | | | | Total price per U (€): | 99.34 |

**Table B.6:** Phase 3 price table

| No. | Id. | Unit | Description | Quantity | Price per unit of resource (€) | Total (€) |
|---|---|---|---|---|---|---|
| 3 | 3 | | **Model development** | | | |
| 3.1 | 03.01 | U | Programming language learning | | | |
| | L.03 | h | Junior biomedical engineer labour | 18 | 11.79 | 212.22 |
| | MAT.03 | u | Anaconda (Python) licence | 0.1 | 0.00 | 0.00 |
| | MAT.02 | u | PC: Intel(R) Core(TM) i7, 8 GB RAM | 0.04 | 650.00 | 26.00 |
| | | % | Indirect costs | 4 | 238.22 | 9.53 |
| | | | | | Total price per U (€): | 247.75 |
| 3.2 | 03.02 | U | Model planning | | | |
| | L.03 | h | Junior biomedical engineer labour | 5 | 11.79 | 58.95 |
| | MAT.03 | u | Anaconda (Python) licence | 0.1 | 0.00 | 0.00 |
| | MAT.02 | u | PC: Intel(R) Core(TM) i7, 8 GB RAM | 0.01 | 650.00 | 6.50 |
| | | % | Indirect costs | 4 | 65.45 | 2.62 |
| | | | | | Total price per U (€): | 68.07 |
| 3.3 | 03.03 | U | Data codification | | | |
| | L.03 | h | Junior biomedical engineer labour | 32 | 11.79 | 377.28 |
| | MAT.03 | u | Anaconda (Python) licence | 0.4 | 0.00 | 0.00 |
| | MAT.02 | u | PC: Intel(R) Core(TM) i7, 8 GB RAM | 0.07 | 650.00 | 45.50 |
| | | % | Indirect costs | 4 | 422.78 | 16.91 |
| | | | | | Total price per U (€): | 439.69 |
| 3.4 | 03.04 | U | Model training | | | |
| | L.03 | h | Junior biomedical engineer labour | 104 | 11.79 | 1226.16 |
| | MAT.04 | u | Google Colab licence | 0.7 | 0.00 | 0.00 |
| | MAT.02 | u | PC: Intel(R) Core(TM) i7, 8 GB RAM | 0.23 | 650.00 | 149.50 |
| | | % | Indirect costs | 4 | 1375.66 | 55.03 |
| | | | | | Total price per U (€): | 1430.69 |
| 3.5 | 03.05 | U | Follow-up meeting | | | |
| | L.01 | h | Supervisor labour | 1 | 20.39 | 20.39 |
| | L.03 | h | Junior biomedical engineer labour | 1 | 11.79 | 11.79 |
| | | % | Indirect costs | 4 | 32.18 | 1.29 |
| | | | | | Total price per U (€): | 33.47 |

**Table B.7:** Phase 4 price table

| No. | Id. | Unit | Description | Quantity | Price per unit of resource (€) | Total (€) |
|---|---|---|---|---|---|---|
| 4 | 4 | | **Study of results** | | | |
| 4.1 | 04.01 | U | Models validation | | | |
| | L.03 | h | Junior biomedical engineer labour | 25 | 11.79 | 294.75 |
| | MAT.03 | u | Anaconda (Python) licence | 0.1 | 0.00 | 0.00 |
| | MAT.04 | u | Google Colab licence | 0.2 | 0.00 | 0.00 |
| | MAT.02 | u | PC: Intel(R) Core(TM) i7, 8 GB RAM | 0.05 | 650.00 | 32.50 |
| | | % | Indirect costs | 4 | 327.25 | 13.09 |
| | | | | | | |
| | | | | | Total price per U (€): | 340.34 |
| | | | | | | |
| 4.2 | 04.02 | U | Model test | | | |
| | L.03 | h | Junior biomedical engineer labour | 4 | 11.79 | 47.16 |
| | MAT.03 | u | Anaconda (Python) licence | 0.05 | 0.00 | 0.00 |
| | MAT.04 | u | Google Colab licence | 0.1 | 0.00 | 0.00 |
| | MAT.02 | u | PC: Intel(R) Core(TM) i7, 8 GB RAM | 0.01 | 650.00 | 6.50 |
| | | % | Indirect costs | 4 | 53.66 | 2.15 |
| | | | | | | |
| | | | | | Total price per U (€): | 55.81 |
| | | | | | | |
| 4.3 | 04.03 | U | Interpretation of results | | | |
| | L.03 | h | Junior biomedical engineer labour | 30 | 11.79 | 353.70 |
| | MAT.01 | u | Microsoft 365 licence | 0.5 | 140.00 | 70.00 |
| | MAT.02 | u | PC: Intel(R) Core(TM) i7, 8 GB RAM | 0.06 | 650.00 | 39.00 |
| | | % | Indirect costs | 4 | 462.70 | 18.51 |
| | | | | | | |
| | | | | | Total price per U (€): | 481.21 |
| | | | | | | |
| 4.4 | 04.04 | U | Follow-up meeting | | | |
| | L.01 | h | Supervisor labour | 1 | 20.39 | 20.39 |
| | L.02 | h | Experimental director labour | 0.2 | 20.34 | 4.07 |
| | L.03 | h | Junior biomedical engineer labour | 1 | 11.79 | 11.79 |
| | | % | Indirect costs | 4 | 36.248 | 1.45 |
| | | | | | | |
| | | | | | Total price per U (€): | 37.70 |

**Table B.8:** Phase 5 price table

| No. | Id. | Unit | Description | Quantity | Price per unit of resource (€) | Total (€) |
|---|---|---|---|---|---|---|
| 5 | 5 | | **Model implementation** | | | |
| 5.1 | 05.01 | U | Web server implementation | | | |
| | L.03 | h | Junior biomedical engineer labour | 7 | 11.79 | 82.53 |
| | MAT.03 | u | Anaconda (Python) licence | 0.025 | 0.00 | 0.00 |
| | MAT.02 | u | PC: Intel(R) Core(TM) i7, 8 GB RAM | 0.01 | 650.00 | 6.50 |
| | | % | Indirect costs | 4 | 89.03 | 3.56 |
| | | | | | | |
| | | | | | Total price per U (€): | 92.59 |
| | | | | | | |
| 5.2 | 05.02 | U | Follow-up meeting | | | |
| | L.01 | h | Supervisor labour | 1 | 20.39 | 20.39 |
| | L.02 | h | Experimental director labour | 1.5 | 19.37 | 29.06 |
| | L.03 | h | Junior biomedical engineer labour | 3 | 11.79 | 35.37 |
| | | % | Indirect costs | 4 | 84.815 | 3.39 |
| | | | | | | |
| | | | | | Total price per U (€): | 88.21 |

**Table B.9:** Phase 6 price table

| No. | Id. | Unit | Description | Quantity | Price per unit of resource (€) | Total (€) |
|---|---|---|---|---|---|---|
| 6 | 6 | | **Documents writing** | | | |
| 6.1 | 06.01 | U | Project report writing | | | |
| | L.03 | h | Junior biomedical engineer labour | 110 | 11.79 | 1296.90 |
| | MAT.05 | u | Overleaf (Latex) licence | 0.7 | 0.00 | 0.00 |
| | MAT.02 | u | PC: Intel(R) Core(TM) i7, 8 GB RAM | 0.25 | 650.00 | 162.50 |
| | | % | Indirect costs | 4 | 1459.40 | 58.38 |
| | | | | | | |
| | | | | | Total price per U (€): | 1517.78 |
| | | | | | | |
| 6.2 | 06.02 | U | Budget writing | | | |
| | L.03 | h | Junior biomedical engineer labour | 15 | 11.79 | 176.85 |
| | MAT.01 | u | Microsoft 365 licence | 0.5 | 140.00 | 70.00 |
| | MAT.05 | u | Overleaf (Latex) licence | 0.3 | 0.00 | 0.00 |
| | MAT.02 | u | PC: Intel(R) Core(TM) i7, 8 GB RAM | 0.03 | 650.00 | 19.50 |
| | | % | Indirect costs | 4 | 266.35 | 10.65 |
| | | | | | | |
| | | | | | Total price per U (€): | 277.00 |
| | | | | | | |
| 6.3 | 06.03 | U | Follow-up meeting | | | |
| | L.01 | h | Supervisor labour | 1 | 20.39 | 20.39 |
| | L.02 | h | Experimental director labour | 0.25 | 19.37 | 4.84 |
| | L.03 | h | Junior biomedical engineer labour | 1 | 11.79 | 11.79 |
| | | % | Indirect costs | 4 | 37.0225 | 1.48 |
| | | | | | | |
| | | | | | Total price per U (€): | 38.50 |

## 1.5 Measurements

In Table B.10, the number of work units necessary to complete each of the steps described above is shown. This will condition the final price of each phase.

**Table B.10:** Measurements price table

| 1. Preliminary study | | | |
|---|---|---|---|
| No. | Unit | Description | Measurement (U) |
| 1.1 | U | Project definition | 1 |
| 1.2 | U | Project planning | 1 |
| 1.3 | U | Investigation of the problem | 1 |
| | | | |
| **2. Data processing** | | | |
| No. | Unit | Description | Measurement (U) |
| 2.1 | U | Initial study of the database | 1 |
| 2.2 | U | Homologous search | 1 |
| 2.3 | U | Homologous clustering | 1 |
| 2.4 | U | Motifs search | 1 |
| 2.5 | U | Follow-up meeting | 2 |
| | | | |
| **3. Model development** | | | |
| No. | Unit | Description | Measurement (U) |
| 3.1 | U | Programming language learning | 1 |
| 3.2 | U | Model planning | 1 |
| 3.3 | U | Data codification | 1 |
| 3.4 | U | Model training | 1 |
| 3.5 | U | Follow-up meeting | 5 |
| | | | |
| **4. Study of results** | | | |
| No. | Unit | Description | Measurement (U) |
| 4.1 | U | Models validation | 1 |
| 4.2 | U | Model test | 1 |
| 4.3 | U | Interpretation of results | 1 |
| 4.4 | U | Follow-up meeting | 5 |
| | | | |
| **5. Model implementation** | | | |
| No. | Unit | Description | Measurement (U) |
| 5.1 | U | Web server implementation | 1 |
| 5.2 | U | Follow-up meeting | 2 |
| | | | |
| **6. Documents writing** | | | |
| No. | Unit | Description | Measurement (U) |
| 6.1 | U | Project report writing | 1 |
| 6.2 | U | Budget writing | 1 |
| 6.3 | U | Follow-up meeting | 4 |

## 1.6 Partial budgets

To finish with the budget breakdown, the partial budget for each of the phases is calculated in Table B.11.

**Table B.11:** Partial budgets

| No. | Unit | Description | Measurement | Price (€) | Total (€) |
|---|---|---|---|---|---|
| 1.1 | U | Project definition | 1 | 107.22 | 107.22 |
| 1.2 | U | Project planning | 1 | 214.45 | 214.45 |
| 1.3 | U | Investigation of the problem | 1 | 68.07 | 68.07 |
| | | | | | |
| | | | | Total partial budget 1. Preliminary study (€): | 389.74 |

| No. | Unit | Description | Measurement | Price (€) | Total (€) |
|---|---|---|---|---|---|
| 2.1 | U | Initial study of the database | 1 | 43.54 | 43.54 |
| 2.2 | U | Homologous search | 1 | 551.30 | 551.30 |
| 2.3 | U | Homologous clustering | 1 | 303.56 | 303.56 |
| 2.4 | U | Motifs search | 1 | 495.50 | 495.50 |
| 2.5 | U | Follow-up meeting | 2 | 99.34 | 198.68 |
| | | | | | |
| | | | | Total partial budget 2. Data processing (€): | 1592.58 |

| No. | Unit | Description | Measurement | Price (€) | Total (€) |
|---|---|---|---|---|---|
| 3.1 | U | Programming language learning | 1 | 247.75 | 247.75 |
| 3.2 | U | Model planning | 1 | 68.07 | 68.07 |
| 3.3 | U | Data codification | 1 | 439.69 | 439.69 |
| 3.4 | U | Model training | 1 | 1430.69 | 1430.69 |
| 3.5 | U | Follow-up meeting | 5 | 33.47 | 167.35 |
| | | | | | |
| | | | | Total partial budget 3. Model development (€): | 2353.55 |

| No. | Unit | Description | Measurement | Price (€) | Total (€) |
|---|---|---|---|---|---|
| 4.1 | U | Models validation | 1 | 340.34 | 340.34 |
| 4.2 | U | Model test | 1 | 55.81 | 55.81 |
| 4.3 | U | Interpretation of results | 1 | 481.21 | 481.21 |
| 4.4 | U | Follow-up meeting | 5 | 37.70 | 188.5 |
| | | | | | |
| | | | | Total partial budget 4. Study of results (€): | 1065.86 |

| No. | Unit | Description | Measurement | Price (€) | Total (€) |
|---|---|---|---|---|---|
| 5.1 | U | Web server implementation | 1 | 92.59 | 92.59 |
| 5.2 | U | Follow-up meeting | 2 | 88.21 | 176.42 |
| | | | | | |
| | | | | Total partial budget 5. Model implementation (€): | 269.01 |

| No. | Unit | Description | Measurement | Price (€) | Total (€) |
|---|---|---|---|---|---|
| 6.1 | U | Project report writing | 1 | 1517.78 | 1517.78 |
| 6.2 | U | Budget writing | 1 | 277.00 | 277.00 |
| 6.3 | U | Follow-up meeting | 4 | 38.50 | 154.00 |
| | | | | | |
| | | | | Total partial budget 6. Documents writing (€): | 1948.78 |

# Chapter 2

# Contractual execution budget

Considering the cost for each of the phases obtained previously, the total cost of this Project is calculated. For this purpose, it has been considered a 13% of overheads and an industrial profit of 6%. Also, the value-added tax (IVA in Spanish) of 21% has been applied.

**Table B.12:** Summary price table

| Chapter | Price (€) |
|---|---|
| 1. Preliminary study | 389.74 |
| 2. Data processing | 1592.58 |
| 3. Model development | 2353.55 |
| 4. Study of results | 1065.86 |
| 5. Model implementation | 269.01 |
| 6. Documents writing | 1948.78 |
| | |
| Material execution budget | 7619.52 |
| 13 % of overheads | 990.54 |
| 6 % of industrial profit | 457.17 |
| | |
| Addition | 9067.23 |
| 21 % IVA | 1904.12 |
| | |
| Contractual execution budget | **10971.35** |

The total contractual execution budget is TEN THOUSAND NINE HUNDRED AND SEVENTY-ONE EUROS AND THIRTY-FIVE CENTS.