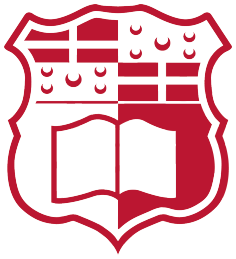


GO Term Predictions in CATH: a Machine Learning Approach

KENNETH PENZA

Supervised by Mr Joseph Bonello

Co-supervised by Dr Jean-Paul Ebejer

Department of Artificial Intelligence

Faculty of ICT

University of Malta

January, 2019

A dissertation submitted in partial fulfilment of the requirements for the degree of Master of Science in Artificial Intelligence.

WWW.UM.EDU.MT

First edition, May 20, 2019

L-Università ta' Malta

Declaration by Postgraduate Students

(a) Authenticity of Dissertation

I hereby declare that I am the legitimate author of this Dissertation and that it is my original work.
No portion of this work has been submitted in support of an application for another degree or qualification of this or any other university or institution of higher education.
I hold the University of Malta harmless against any third party claims with regard to copyright violation, breach of confidentiality, defamation and any other third party right infringement.

(b) Research Code of Practice and Ethics Review Procedures

I declare that I have abided by the University's Research Ethics Review Procedures.
As a Master's student, as per Regulation 58 of the General Regulations for University Postgraduate Awards, I accept that should my dissertation be awarded a Grade A, it will be made publicly available on the University of Malta Institutional Repository.

Faculty/Institute/Centre/School	Faculty of ICT
Degree	Master of Science in Artificial Intelligence
Title	GO Term Predictions in CATH: a Machine Learning Approach
Candidate (Id.)	Kenneth Penza (530479M)
Signature of Student	_____
Date	May 20, 2019

08.02.2018

To my father

Whose memories and words of wisdom will never fade.
Francis J. Penza (1938-2018)

Acknowledgements

I would like to thank my supervisor Mr Joseph Bonello and co-supervisor Dr Jean-Paul Ebejer who with great dedication, patience and professionalism, supported me throughout this journey. The lengthy discussions on ideas and issues helped me broaden my knowledge, hone my skills and tackle subject areas outside of my comfort zone. The initial discussions were fundamental to instil my curiosity in the research area.

The formulation of the project was assisted by Prof Christine Orengo and Dr Jon Lees from the Orengo Group at University College London. I would like to thank them for their availability for the various conference calls and the datasets provided.

This journey was a challenging one, but with the continuous patience and unconditional support of my family and close friends, I managed to reach this milestone.

Abstract

Proteins perform different tasks within an organism such as regulation and signalling. Protein function is characterised through laboratory experiments or predicted using computational methods. Protein function is described using Gene Ontology (GO) terms. Protein sequencing is the process of determining the amino acid sequence that makes up the protein. Technological improvements in sequencing technology is making the process more accessible, leading to an ever-increasing growth rate of protein databases. The low throughput of laboratory experiments and increasing rate of proteins deposited in protein databases has made protein function prediction (PFP) a central problem in computational biology. Domains are independent structural units that have their own structural and function. Structural protein databases categorise protein using structural properties. CATH is a structural database of protein domain using four levels of hierarchy.

This research applied machine learning (ML) techniques to improve PFP. The protein function aspect investigated was molecular function. This research uses a labelled ML dataset consisting for GO terms, features extracted from protein sequence and proportions computed from protein databases such as CATH and PFAM. The problem was tackled by defining five experiments that were executed on *Homo sapiens* and *E. coli* datasets. The model performance was measured using F_{max} computed as per Critical Assessment of Functional Annotation (CAFA) shared task methodology. The first experiment applied automatic feature selection using four different fitness methods based on Random Forest and Support Vector Machine. The second experiment applied different neural network architectures to the datasets. The third experiment applied cross validation to the automatic feature selection process to assess dataset sensitivity in the feature selection process. The fourth experiment investigated the amount of training data required by best performing ML model for each species identified in the first experiment. The fifth experiment investigated the application of the best performing ML model for each species identified in the first experiment to other species.

The methods selected in the first and second experiment were evaluated on the CAFA3 targets. The RF with Gini node splitting criterion outperforms the best CAFA2 methods by an F_{max} of 0.01 for *Homo sapiens* and an F_{max} of 0.16 for *E. coli*. The cross validation of the automatic feature selection shows that *E. coli* models were more sensitive to changes in the dataset with respect to *Homo sapiens* models. The smaller *E. coli* dataset explains the sensitivity observed. The training dataset size experiment shows that the models have similar performance levels with the same amount of training data. The experiment that applied species-specific models to different species confirms the intuition that models perform well on species of the same domain, and that performance decreases as evolutionary distance increases.

The results show that features based on protein structure and proportions from structural protein databases permit reliable PFP.

Contents

List of Figures

List of Tables

List of Abbreviations

CAFA Critical Assessment of Functional Annotation . . . 3
CART Classification And Regression Trees . . . 21
DAG Directed Acyclic Graph . . . 17
DL Deep Learning . . . 47
DRBM Deep Restricted Boltzmann Machines . . . 48
ECO Evidence Code Ontology . . . 11
GA Genetic Algorithm . . . 34
GO Gene Ontology . . . 11
HDF Hierarchical Data Format . . . 61
ID3 Iterative Dichotomiser 3 . . . 19
LSTM Long Short-Term Memory . . . 47
ML Machine Learning . . . 5
MSA Multiple Sequence Alignment . . . 13
NMR Nuclear Magnetic Resonance . . . 9
NN Neural Network . . . 6
OVO One vs One . . . 24
OVR One vs Rest . . . 24
PFP Protein Function Prediction . . . 5
PPI Protein-Protein Interaction . . . 42
RF Random Forest . . . 6
RL Reinforcement Learning . . . 19
RNN Recurrent Neural Network . . . 47
SVD Singular Value Decomposition . . . 47

SVM Support Vector Machine 6
UCL University College London 4
UniProtKB UniProt Knowledgebase 2

1

Introduction

1.1 | Problem Definition

Proteins perform a variety of activities and functions in an organism. These functions include acting as a catalyst to facilitate chemical reactions, enabling signalling, and performing regulation functions inside the organism (Lesk, 2013).

Amino acids are small molecules composed of chains of carbon (C), hydrogen (H), oxygen (O), nitrogen (N) and sulphur (S) atoms. There are twenty naturally occurring amino acids. Proteins are assembled using chains of amino acids. The number of amino acids in a protein varies from protein to protein. The sequence of the protein represents the linear combination of amino acids that make up the protein. Protein sequences are represented as strings of arbitrary length composed of characters from a twenty letter alphabet representing amino acids.

The function of a protein is determined through laboratory tests performed on the protein. These types of tests are referred to as "wet lab" or "laboratory curation". This process has low throughput and is both costly and laborious. Experiments follow a scientific rigour and the outcome is a high-quality laboratory verified functionality of the given protein.

The function of a protein can be predicted through computational approaches. These approaches utilise different techniques such as sequence similarity, protein ancestry relationship (homology) and interaction networks. These methods have higher throughput but are less reliable. This area is actively researched with scientists using different aspects to improve the performance of the predictor.

Annotations are used to attach functionality details to protein sequences in protein databases. Annotations include three components: the function description, evidence code and method details such as journal article. The function is described using a

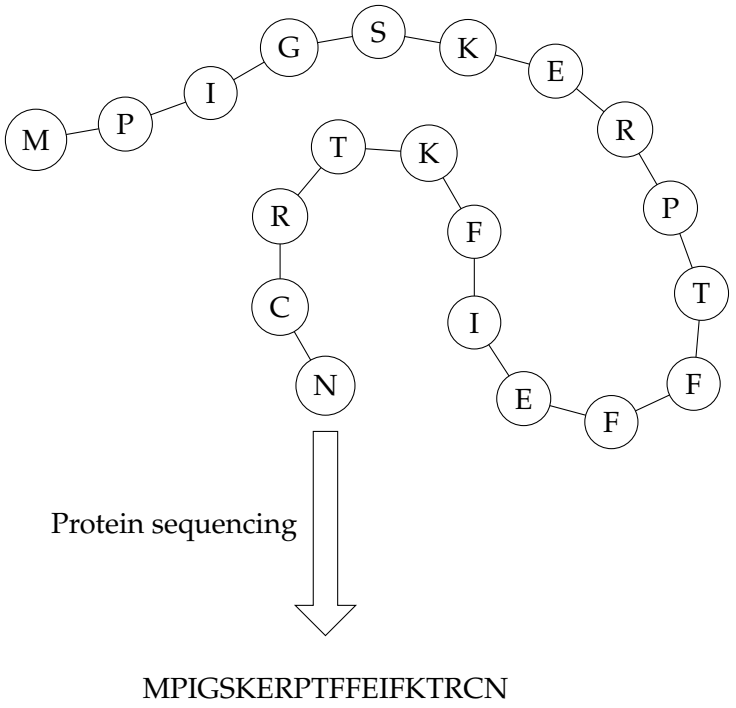


Figure 1.1: Protein sequencing is the process used to determine the amino acids making up the protein.

species agnostic vocabulary derived from a biological ontology such as Gene Ontology. The evidence code describes the class of the method used to determine the described functionality. Method details contain information related to evidence documentation such as peer reviewed research defining the experimental conditions and the respective outcomes.

Sequencing is the process of determining the amino acid sequence of a given protein as illustrated in Figure 1.1. Technological improvements are making this process more efficient and affordable. This gave rise to the number of protein sequences available in biological databases. The ever-increasing number of proteins without functional annotations is highlighting the need for a fast-computational protein function prediction method (Scaiewicz and Levitt, 2015).

UniProt Knowledgebase (UniProtKB) is a collection of biological databases that stores protein details and functionality information, which include the sequence of the protein and originating species. The functional information includes functionality description using GO terms and details on the process used to determine the functionality of the protein. UniProtKB database is divided into two categories, the manually annotated section “Swiss-Prot” and the automatically annotated category “TrEMBL”. Fig-

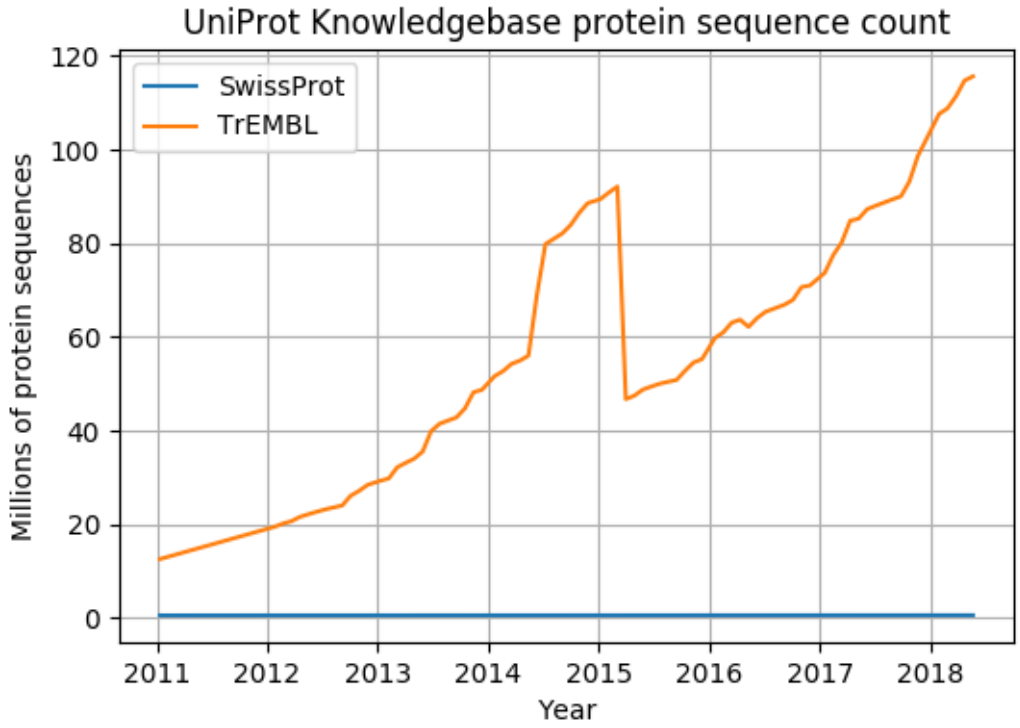


Figure 1.2: Number of protein sequences in UniProtKB databases. Figure compiled using UniProtKB release statistics, available from UniProt FTP site[1]. The drop in TrEMBL growth is linked to redundancy minimisation implemented on March 2015.

ure 1.2 shows the number of sequences in UniProtKB databases. From Figure 1.2, "SwissProt" growth looks stationary in comparison with the sheer growth experienced by "TrEMBL".

The Critical Assessment of Functional Annotation (CAFA) challenge is a time-based shared task with the purpose of assessing the state of protein function prediction methods. The CAFA shared task selects a set of proteins that lack functionality description and asks the community to submit their predictions. The outcome of the CAFA shared task is a rigorous performance evaluation of the different methods, on the proteins that acquired functionality description in the meantime.

[1] `ftp://ftp.uniprot.org/pub/databases/uniprot/previous_releases` (Accessed 2018-12-28)

1.2 | Motivation

The ability to map protein sequences to functionality is key to different areas in biology. This knowledge would provide better understanding of organisms. In drug design, it would enable medicinal to target specific proteins reducing side effects (Berger and Iyengar, 2009). Grant (2011) reports that detailed knowledge of protein function will permit precise interaction mapping of proteins and thus improve drug specificity.

Protein function prediction is a central problem within bioinformatics. To date, there is no protein function prediction technique that can replace high-quality experimental annotations (Koskinen et al., 2015). This research exploits bioinformatics computational methods and machine learning techniques to predict protein function. Generated predictions (annotations) will be utilised to hint researchers about the functionality of a given sequence.

Protein function prediction is a complex problem. Sequence similarity approaches determine ancestry relationship (homology) between proteins through conserved regions. This relation can be exploited to transfer functionality between proteins on the basis that structural similarity suggests functional similarity. Paralogs tend to functionally diverge from their ancestors (Baxevanis and Ouellette, 2004). Homologs of moonlighting proteins may have none, a subset or all functions of the ancestor protein (Jeffery, 2015). Homology based methods effectiveness is hindered in the case of paralogs and moonlighting proteins as the sequence relationship cannot be used to transfer protein function.

1.3 | Scope

This dissertation uses computational methods to predict the molecular function of proteins. The computational techniques will use a dataset of protein measurements and output Gene Ontology terms describing molecular function. The performance of the computational techniques will be assessed using CAFA metrics and compared against CAFA methods.

The dataset used in this research was provided by Dr Jon Lees a member of the Orengo Group at University College London (UCL). This dataset was generated by Dr Lees using readings obtained from different bioinformatics tools used to extract structural information for the protein sequences. Further information was extracted from biological databases and converted to different ratio values.

The scope of this dissertation is to identify putative features from the provided

dataset that enable reliable protein function prediction. For this purpose, a number of experiments with different Machine Learning (ML) techniques will be performed and evaluated.

1.4 | Aims and Objectives

The aim of this research is to apply ML techniques to improve protein function prediction. The defined aim will be achieved by fulfilling the following objectives:

1. Identify features correlated with protein functionality to enable reliable Protein Function Prediction (PFP).
2. Investigate the application of neural network for PFP.
3. Investigate the sensitivity of feature selection.
4. Investigate the amount of training data required to train the ML model.
5. Investigate if models are transferable across species.

1.5 | Our Solution

In the initial phases of the dissertation, high-level discussions on bioinformatics with two members of the Orengo Group at UCL[2], namely Prof Christine Orengo and Dr Jon Lees were held. Orengo Group is a research group that focuses on computational techniques to classify protein through evolutionary relationships.

Within the protein structure, there are structural units called domains that are structurally independent from the protein itself. Each domain has its own structure that can be associated with a given function.

Structural databases classify proteins using the structural characteristics of the protein. CATH is a structural database that classifies protein using a hierarchical classification. Proteins grouped at the same hierarchical level share common structural attributes.

Dr Jon Lees made available two datasets that were generated by the Orengo Group at UCL for the CAFA shared task. The provided datasets are labelled and contain several features for each protein. The features describe the protein in terms of attributes from different structural databases including CATH and sequence structural properties.

[2] `http://www.ucl.ac.uk/orengo-group` (Accessed 2018-12-28)

The training dataset contains ML features for the CAFA2 training dataset. Whilst the evaluation dataset contains ML features for the CAFA3 targets.

The proposed solution will use the training dataset to determine the dataset features that enable reliable protein function predictions. Species-specific datasets will be extracted from the training dataset to train species-specific ML models. Different ML techniques will be used to determine the most appropriate one for this data.

Four components will be developed: the pipeline component, the genetic algorithm, the machine learning driver and evaluation implementation. The different modules will be utilised to perform a number of experiments.

This work will utilise the two species-specific datasets to perform five experiments. The first experiment uses the two datasets to perform automatic feature selection using different ML techniques to evaluate features. The two ML techniques investigated will be Random Forest (RF) and Support Vector Machine (SVM). The second experiment will investigate the application of different Neural Network (NN) architectures to the two datasets. The third experiment applies cross validation on automatic feature selection. The fourth experiment determines the training dataset size required for the ML model. The fifth experiment is a control experiment that applies a species-specific model to predict functionality of protein of another species. Performance of the different ML models will be evaluated using CAFA metrics.

Automatic feature selection will randomly select features from the feature space and evaluate their performance using CAFA metrics. Through the ML wrapper part, the same code can be utilised for both SVM and RF techniques. The system will log execution details, the features selected and the respective CAFA performance to facilitate reporting.

The described architecture enables the required experimentation to cover the aims defined in the outset. The ML methods will be evaluated using the CAFA shared task methodology. The evaluation dataset has the ML features of the CAFA3 targets. At the time of writing the results of the CAFA3 are not available. For this purpose this work will evaluate the performance obtained on the CAFA3 targets against CAFA2 submissions.

1.6 | Document Structure

The rest of this dissertation is split into five sections.

Background and Literature Review The background section introduces the different areas of relevance for this research. The related work section tackles the work performed and the current state of this area of research. The discussion details the different approaches employed to tackle protein function prediction.

Methodology This chapter gives a detailed description on how the knowledge acquired was utilised to tackle the problem. The software architecture section details the aspects related to the solution itself. The discussion moves on to describe the provided dataset. Another section describes the processing carried out on the dataset and how it is utilised in the machine learning pipeline. The last section in this chapter describes the experiments performed.

Results and Discussion This chapter reports the results of each experiment set in the previous chapter. The results of each experiment are critically discussed with respect to the objectives set for this research project.

Evaluation This chapter discusses the performance of the proposed solution against CAFA benchmarks. The evaluation of the best methods, will be paper evaluated against CAFA submissions.

Conclusions The concluding chapter summarises the research performed and discusses how it is pertinent to the problem tackled. This chapter reports briefly on the achieved aims and objectives. Subsequent subsections summarise the critique of the solution and reports on further work to be performed.

1.7 | Summary

This chapter introduced the problem area being tackled and the motivation for this work, highlighting the need for such work in this area. With the clear goal defined at the outset, the next chapter provides the required theory and to tackle the defined problem.

2

Background & Literature Overview

This chapter introduces the areas of protein similarity, protein databases, protein functionality, machine learning techniques, performance metrics and approaches to protein function prediction. The first section provides the necessary background information and introduces the techniques used in the different areas. The second section reports on the work performed in the area of protein function prediction and the current state area of research.

2.1 | Biological Classification

In biology organisms are categorised in different classes based on different characteristics. A taxon is a set of organisms grouped at a specific taxonomic rank. The biological classification defines different hierarchical taxonomic ranks, ranging from life to species. The second level of the taxonomic rank is the domain, that categories organisms based on the cell structure. The domain defines three categories:

- Bacteria, whose cells do not have a nucleus.
- Eukaryote, whose cells have a nucleus with DNA.
- Archaea, which are single cell organisms.

2.2 | Proteins

Proteins are composed of amino acid chains of arbitrary length. The amino acid sequence of the protein represents the flat representation of the protein. The chemical attraction of the different atoms within the protein chain creates a bond that causes the

protein to fold into a complex 3D structure. Protein folding is described in the four structural levels (Lesk, 2013):

- Primary structure is the amino acid sequence.
- Secondary structure considers the conformation of the peptide chain in the protein molecular. Many proteins have the conformation of α-helices or β-sheets.
- Tertiary structure considers pattern formed with the interaction between the α-helices or β-sheets.
- Quaternary structure, for proteins with more than one subunit, considers the binding of monomers to the tertiary structure.

Structural biology studies proteins and investigates the 3D conformation of the protein. To determine the structure of a sequence, tools such as X-ray crystallography, Nuclear Magnetic Resonance (NMR) and cryo-electron microscopy are utilised (Bruno et al., 2017; Lafita et al., 2018). The ability to determine the structure of a protein for its sequence is being actively researched (Moult et al., 2018).

The folding of the protein sequence into a 3D conformation gives the protein its functionality (Hunter, 1993). Within the protein structure, there are structural units called domains that are structurally independent from the protein itself. Each domain has its own structure that can be associated with a given functionality. A domain can occur in different proteins and species. Proteins can have multiple domains that enable the protein to have multiple functions.

Domains are associated with one function. "Moonlighting" proteins are a special class of proteins whereby a single domain performs multiple functions. Huberts and van der Klei (2010) report that the functionality provided is in majority of the cases dissimilar. Currently, this area is being actively researched to identify these types of proteins and determine the reason for this mechanism.

Within an organism, genes are used to transcribe proteins that perform different functions within the organism. The evolution of organisms might trigger gene mutations in reaction to external factors during the duplication process. The modified gene might produce a slightly modified protein called "paralog" (Baxevanis and Ouellette, 2004).

Since proteins are the building blocks of life, the ability to map a protein with a specific function is key to understanding organisms. This knowledge is central to drug development, whereby a pathology is studied, and molecules are identified to target specific reactions. This process can be more expedite and effective if the exact behaviour

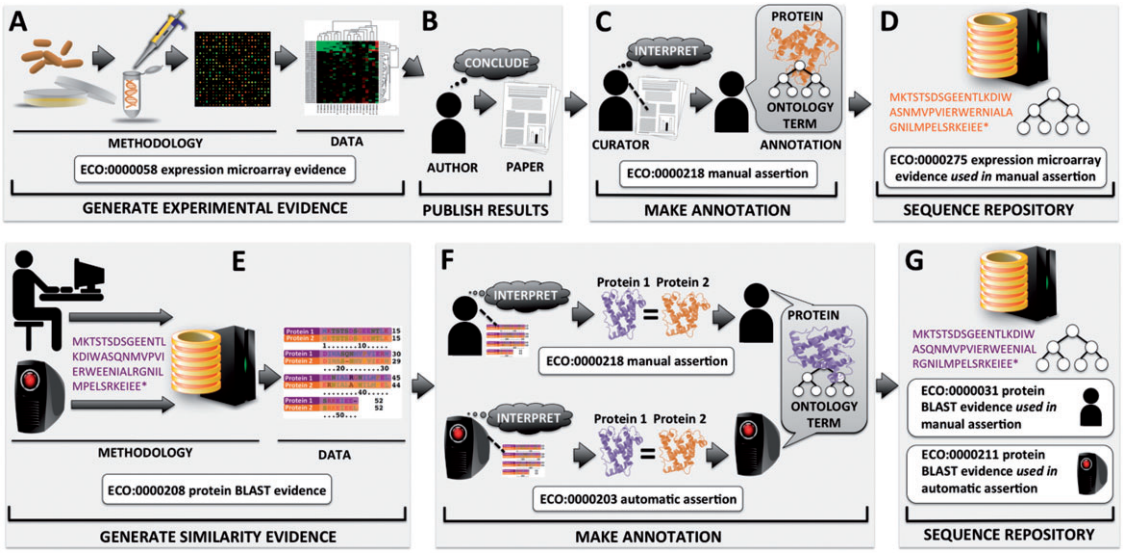


Figure 2.1: Protein annotation process and evidence description using ECO ontology reproduced from Chibucos et al. (2014). The top row, steps *A-D* illustrates steps for experimental evidence. The bottom row, steps *E-G* detail the steps for electronic/automatic evidence.

of all the proteins involved is known beforehand, ideally computationally (Dudley et al., 2011). Moreover, since there are only a few thousand domains that re-occur in nature, the problem is much simpler since it does not involve a search of all the combinations of amino acids that make up the protein sequences.

2.3 | Protein Function Annotation

Annotations are the mechanism used to attach a protein function descriptions in protein databases. Through annotations the recorded functionalities of a protein are stored using a defined protocol to facilitate retrieval.

The function of a protein can be determined using wet laboratory experiments and computational methods. Protein function determined through wet laboratory experiments that follow a rigorous methodology. Experiment findings are published in literature detailing the experiment setup and the outcomes observed. These results are stored in the biological databases as annotations. Annotations originating from laboratory experimentation are high-quality annotations as they are physically verified (UniProt Consortium, 2017).

Computational methods are used to predict the function of a protein. These methods

use different approaches to garner information from different data sources to improve the performance of the model. Mazandu et al. (2017) report that prediction methods are improving, however, their reliability is still not at par with experimental curation.

The quality of the annotation is denoted through evidence codes. In bioinformatics, there are two main standards namely the Gene Ontology (GO) evidence codes (Consortium, 2004) and Evidence Code Ontology (ECO) (Chibucos et al., 2014). Figure 2.1 illustrates the process of how annotations are generated and annotated using ECO. GO evidence codes can be mapped to ECO codes using the information from GO website[1].

A protein sequence can have multiple annotations for its domains. Each annotation is independent of the other annotations describing the same or a different behaviour. Through evidence codes, annotations can be filtered to consider only annotations originating from specific methods.

2.4 | Protein Homology

Protein homology defines an evolutionary relationship between proteins, such as a common ancestor (Pearson, 2013). This evolutionary link between protein can be utilised to transfer protein function between related proteins (Hamp et al., 2013).

Two species can be linked together if they have common features, for example, adaption to a specific environmental condition. In bioinformatics, gene homology is divided into two subclasses, paralogues and orthologues. Paralogues are genes linked through gene duplication whilst orthologues are genes related through speciation. Within the evolutionary context, these two subclasses have different endings. Paralogues have duplicate functionality and consequently, in the long term, they either diverge functionality or are lost. On the other hand, orthologues tend to take the function of their precursor and thus are conserved (Das et al., 2015a; Theißen, 2002).

Thornton et al. (1999) report that there are two approaches to structurally classify proteins; the phylogenetic approach that considers the evolutionary link to other proteins and phenetic approach that describes the structure and folds. Proteins grouped via a homologous relation have similar 3D structures (Thornton et al., 1999).

Homology (ancestry) relationship between proteins is established using sequence similarity measures. The degree of sequence similarity required to confirm homology is generally defined as at least 30% identity, however this value filters out homologues (Pearson, 2013). Sequence alignment tools such as BLAST report the expect value (E-value) for each hit (Camacho et al., 2009). E-value defines the frequency that a match

[1] `http://www.geneontology.org/page/guide-go-evidence-codes` (Accessed 2018-12-28)

is found by coincidence in the database (Tatusova and Madden, 1999). Pearson (2016) reports that the E-value is more reliable to identify homologues and that a value less than 10^{-6} identifies a highly probable homology link. Homology detection is being tackled through different techniques such as sequence alignment, profile alignment, HMM alignment and Machine learning based methods (Li et al., 2017b; Ponting and Russell, 2002). The performance of the different method varies Saripella et al. (2016) report that the top performing methods are CSBLAST and HMMER3, both in terms of accuracy and runtime.

Within bioinformatics, sequence alignment is used to determine the degree of similarity between the two sequences. Altschul et al. (1990) identifies two types of alignment: global or local. Global alignment considers the overall length of the sequence, whilst local alignment aims to find conserved regions within the sequence (Altschul et al., 1990; Smith and Waterman, 1981).

Needleman and Wunsch (1970) proposed an algorithm that enables computational detection of sequence relationship (homology). For the two sequences under consideration, a matrix is built of size *length(seq1)* and *length(seq2)*. The process of computing the matrix values considers a 'Scoring system', whereby matches contribute positively

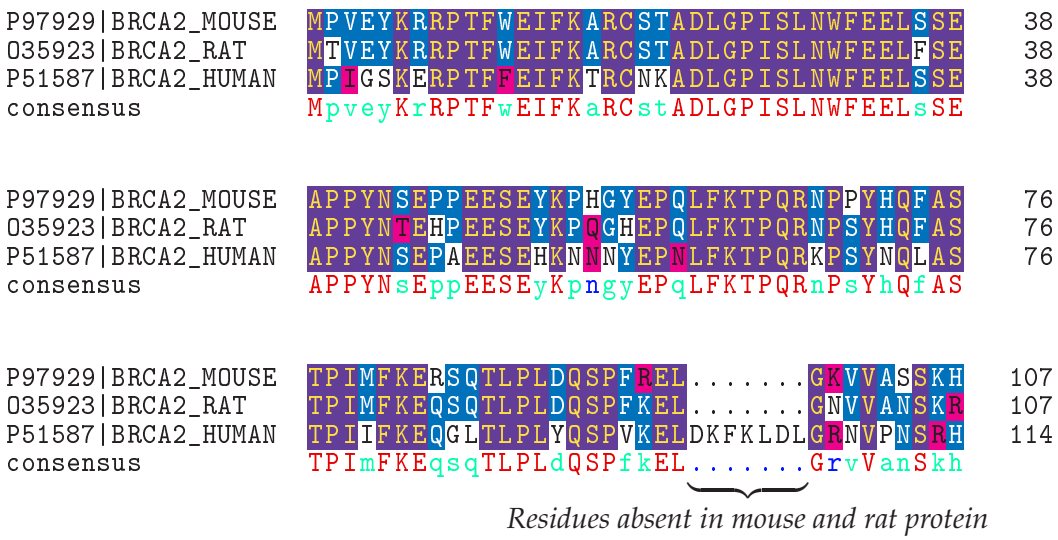


Figure 2.2: Multiple sequence alignment of BRCA2 protein in mouse, rat and human. Alignment data was generated using UniProtKB UniProt Consortium (2017) and rendered using TEXshade.

whilst for mismatches and gaps, a penalty factor contributes negatively. The similarity of the two sequences is read out starting from the highest values and moving toward zero using the path of the cells that contributed to the high value.

The work proposed by Smith and Waterman (1981) tackle a limitation of the method proposed by Needleman and Wunsch (1970), namely the ability to tackle sequences of arbitrary length and add statistical rigour. Smith and Waterman (1981) proposed a different 'Scoring system', whereby, matches contribute positively, mismatches and gaps contribute negatively. Cells with a negative value are assigned a value of zero. To determine alignments, the highest positive values in the matrix are highlighted and traced back until the value reaches zero.

The process of aligning multiple sequences together is called Multiple Sequence Alignment (MSA). Through Multiple Sequence Alignment (MSA), the common residues of the sequences are determined. This is generally referred to as "consensus sequence". Figure 2.2 illustrates MSA performed on the BRA2 protein for three species; human, mouse and rat. The consensus sequence shows that the three proteins have high sequence similarity, implying that they are related.

2.5 | Protein Databases

Protein sequences are deposited in protein databases including related information. Different databases focus on specific area of the protein and generally cross-reference each other.

2.5.1 | UniProt

The UniProt protein database consists of three databases namely UniProtKB, UniParc and UniRef. UniProt knowledge (UniProtKB) database is split in two sections, the manually annotated section "Swiss-Prot" and the automatically annotated section "TrEMBL" (UniProt Consortium, 2017). UniProt Archive (UniParc) retrieves protein sequences from a number of databases and stores non-redundant proteins in UniProt archive. Each protein sequence is assigned an immutable ID together with database cross reference information (Leinonen et al., 2004).

UniProt provides UniRef databases, that clusters all entries in UniProtKB and selected UniParc entries using sequence identity. Each UniRef cluster, includes the best annotated sequence, common taxonomy and a list of all cluster members is provided (Suzek et al., 2015). Uniref builds the clusters with 50%, 90% and 100% sequence identity to generate UniRef50, Uniref90 and Uniref100 respectively.

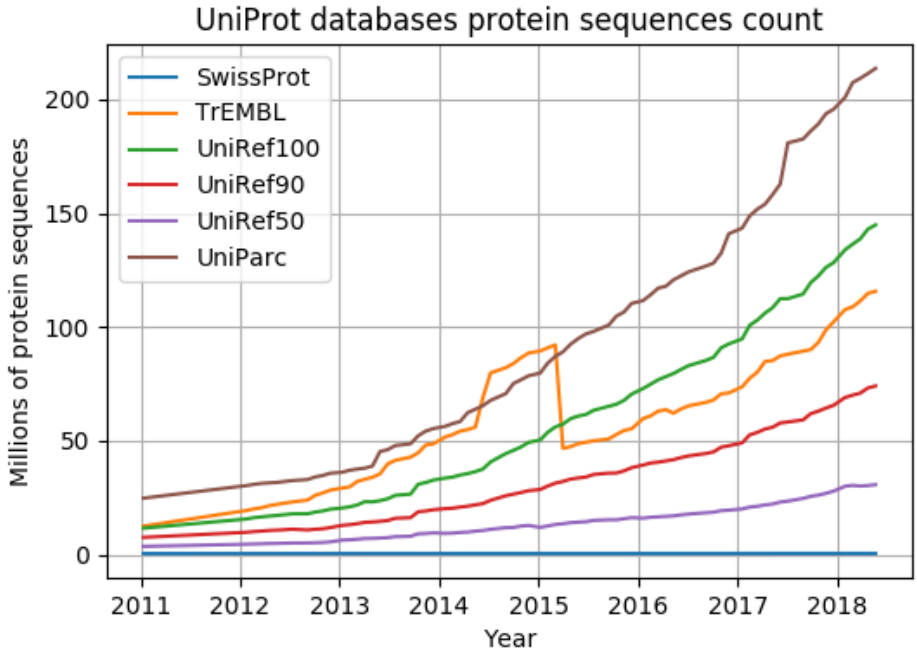


Figure 2.3: Number of sequences in UniProt databases. Figure compiled using UniProtKB release statistics, available from UniProt FTP site[2]. The drop in TrEMBL growth is linked to the implementation of Proteome Redundancy Minimisation (PRM) implemented on March 2015.

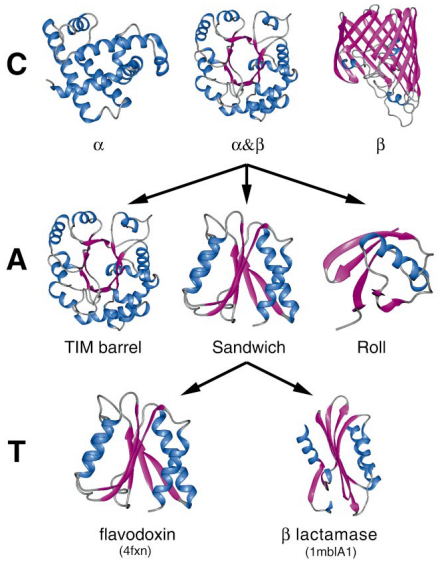


Figure 2.4: Graphical representation of the first three levels of CATH, figure reproduced from Orengo et al. (1997).

[2] `ftp://ftp.uniprot.org/pub/databases/uniprot/previous_releases` (Accessed 2018-12-28)

The number of protein sequences deposited in UniProt databases is constantly increasing. This growth is illustrated in Figure 2.3, that reports the number of sequences in the different UniProt databases through the years 2011 and 2018.

2.5.2 | CATH

CATH is a protein structure classification database. Within CATH, proteins are categorised using structural aspects to facilitate functionality mapping (Sillitoe et al., 2015). Figure 2.4 illustrates the representation of the first three categorisation levels in CATH, *C*lass, *A*rchitecture and *T*opology. The four structural levels used in CATH are (Orengo et al., 1997; Sillitoe et al., 2015):

- *C*lass - is the general classification of the secondary structure, mainly α-helices, main β-sheets, mixed $\alpha - \beta$ and few secondary structures.
- *A*rchitecture - the main architecture formed by the secondary structure.
- *T*opology - groups have the same overall fold.
- *H*omologous Superfamilies - groups proteins that have high structural and function similarly.

Proteins grouped within the same superfamily can indicate that they have some common ancestor. Sillitoe et al. (2015) report, that within Homologous superfamilies, both structural and functional variations at different degrees exist. For this purpose, the functionality of the proteins is used to build "Functional Family" clusters. These clusters are formed by using the protein functionality as a distance measure. FunFams attempt to link functionality with the occurrence of a specific sequence (Sillitoe et al., 2015).

CATH is publicly accessible through website[3]. The website provides a search facility that permits proteins to be searched using UniProt accession ID or FASTA sequences. The database browser enables browsing classification hierarchy and viewing details of the selected superfamily.

CATH classification database clusters the fourth level (Homology) into functionally coherent Functional Families (FunFams). CATH FunFams have been using in PFP and ranked amongst the top ten methods in the Critical Assessment of Functional Annotation (CAFA) shared task (Rentzsch and Orengo, 2013).

[3] `https://www.cathb.info/` (Accessed 2018-12-28)

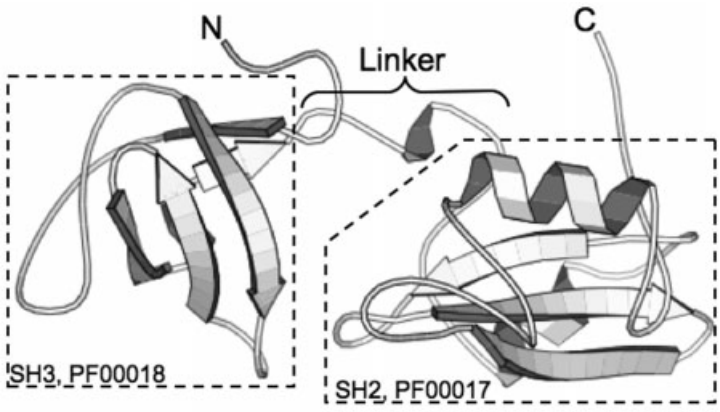


Figure 2.5: Graphical representation of PFAM domain human ABL1 protein structure, reproduced from Sammut et al. (2008)

2.5.3 | PFAM

PFAM is a database of protein domains (Bateman et al., 2000) which are grouped into families. Each PFAM family entry consists of annotations, seed alignment, HMM profile and a full alignment (Bateman et al., 1999). The seed alignment for the family is built by performing multiple sequence alignment on verified members of that PFAM family (Sonnhammer et al., 1997). The seed alignment is used to create the Hidden Markov Model (HMM) profile for the family. The HMM profile is refined until it aligns all members of the PFAM family (Sonnhammer et al., 1997). The full alignment is obtained by aligning the UniProt SwissProt with the HMM alignment.

Domains in the same sequence can be classified into different PFAM families. Figure 2.5 shows the architecture of human ABL1 protein structure with PFAM classified domains evidenced.

PFAM classification provides the data to map domain to functionality. Adding PFAM data to a ML dataset will provide additional information based on a domain to functionality mapping.

An overarching rule in PFAM is that families cannot have overlap. To tackle homology PFAM uses clans, where its members share a common ancestor (Finn et al., 2006). The clan membership is determined by structural and functional similarity, sequence matches multiple family HMM and HMM profile similarity (Finn et al., 2006). Sonnhammer et al. (1998) report that the HMM's profiles can handle different levels of conservations, enabling detection of remote homologues.

2.6 | Gene Ontology

The Gene Ontology (GO) provides a vocabulary to describe protein functionality. The GO aims to address the issue that organism-specific databases classify proteins using a classification or hierarchy oriented towards the target organism. Organism specificity in describing proteins hinders the ability to ask organism independent questions (Consortium et al., 2001). The idea of using a common methodology to describe functionality is supported by the fact that genes are shared amongst life kingdoms (Ashburner et al., 2000).

The Gene Ontology (GO) initiative by the Gene Ontology Consortium has three main aims (Consortium, 2004):

1. Develop ontologies to describe molecular biology.
2. Apply GO ontologies to biological databases.
3. Publish ontologies to enable universal access.

GO defines three non-overlapping ontologies (Ashburner et al., 2000; Consortium, 2004; Consortium et al., 2001):

1. **Biological Process** describes the series of events or molecular functions.
2. **Cellular Component** describe locations, at the level of subcellular structures and macromolecular complexes.
3. **Molecular Function** describes the basic abilities at the molecular level.

GO is defined as a Directed Acyclic Graph (DAG) whereby a vertex represents a term and an edge denotes the relationship between the terms. In the DAG, a vertex representing a term can have multiple parents. GO terms are structured to support "is_a" (subtype of), "part_of" (subcomponents of parent), regulates (necessarily-regulates), "positively_regulates", "negatively_regulates" and "has_part" (necessarily part of) relationships. The GO follows the "True Path Rule" whereby the path between term and the ontology root must always be true (Consortium, 2004; Consortium et al., 2010). Figure 2.6 illustrates the GO DAGs' generated for three GO terms within different ontologies.

The "True Path Rule" ensures that all GO terms always have a path to the ontology root. Example, referencing Figure 2.6, the true path of annotation GO:0022402, is the term set GO:0008150, GO:0009987, GO:0007049, GO:0022402.

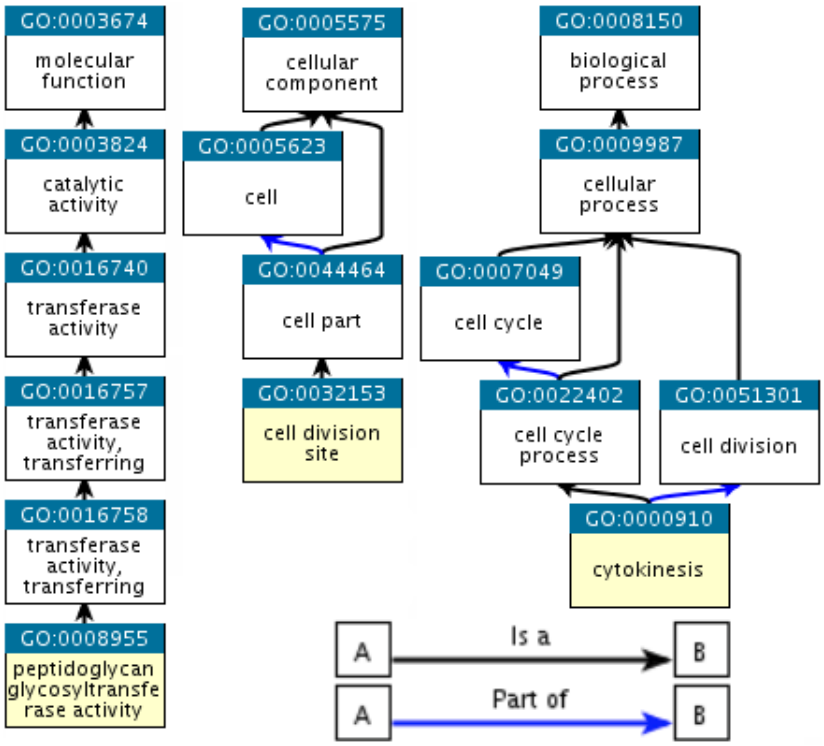


Figure 2.6: Gene Ontology Direct Acyclic Graph of different terms. Figure adapted from QuickGO website[4].

2.7 | Machine Learning

This section gives on overview of the Machine Learning (ML) techniques that will be utilised in this dissertation. For each technique namely Random Forest (RF), Support Vector Machine (SVM) and Neural Network (NN), the high-level concepts and underlying principles are detailed.

Computer programs require programmer input to develop code to perform a specific task. In machine learning, the algorithm learns patterns from data to enable it to make decisions on unseen data based on what it learnt. Machine learning is divided into three main types, supervised, unsupervised and reinforcement learning. In supervised learning, the training data is labelled, whilst in unsupervised learning, the data is not labelled.

In unsupervised learning the expected outcome is not available. For this purpose, a number of techniques are utilised to extract knowledge from the dataset. The association technique computes the probability of the co-occurrence of feature sets. Another

[4] `https://www.ebi.ac.uk/QuickGO/` (Accessed 2018-12-28)

technique is clustering, whereby the samples are mapped to a feature space and grouped in clusters in a distance measure. There are different types of clustering such as agglomerative and spectral clustering (Murphy, 2012).

In supervised learning, the training data consists on N sample maps, where each sample consists of a number of inputs (features) $\{x_1, x_2, ..x_n\}$ mapped to a response variable (output) y. In classification problems, the response variable y is an element of a finite set of classes $y \in \{C_1, .., C_k\}$ where k is the number of classes. The value of k defines the problem type when k is equal to 2 the problem is defined is a binary classification problem, otherwise, it is a multiclass classification problem. In case y is a real number the problem is a regression problem (Murphy, 2012).

Reinforcement Learning (RL) is a branch of ML that tackles autonomous systems. The system operates in an environment and is trained through a system of rewards and penalties. The main aim is the maximisation of cumulative rewards. An important aspect in RL is that the software system must engage in exploring the environment to have a strategy between known rewards and the unknown states (Mitchel, 1997).

The complexity of the dataset is not a function of size but of dimensionality (Breiman et al., 1984). Highly dimensional datasets are demanding in terms of processing memory and training time (Bolón-Canedo et al., 2015). Bolón-Canedo et al. (2015) refers to work carried out by Richard Bellman where the term "curse of dimensionality" is used to describe such circumstances. These datasets are tackled via "feature selection" or "feature extraction" (Bolón-Canedo et al., 2015). In "Feature selection" selected features that are deemed to be "good" are kept discarding the rest, reducing the size of the dataset. In "feature extraction" the input features are used to generate a new dataset. The new dataset maps the input features into intermediate features reducing dimensionality in the process. ML techniques are utilised on the new dataset.

2.7.1 | Random Forest

The process of determining the appropriate splits and which variables to consider is not straight forward. Quinlan (1986) proposed Iterative Dichotomiser 3 (ID3), that iteratively selects a random subset of features and builds a decision tree. The effectiveness of the tree is evaluated against all the training set. In case there are incorrect responses, more features are added.

ID3 computes two metrics Entropy (uncertainty) as per Equation 2.1 and Information Gain as defined in Equation 2.2 to determine the node split criteria. The dataset feature that contributes most to the information gain is selected as the node split criteria.

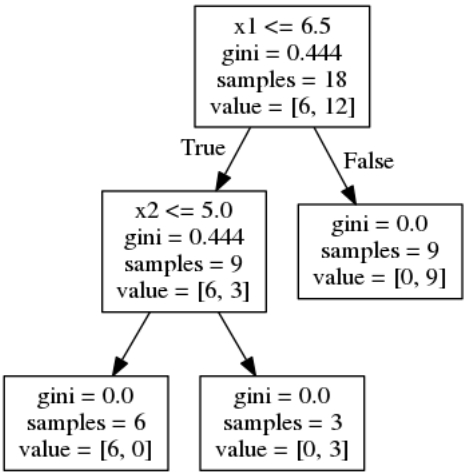


Figure 2.7: Decision Tree representation, including the Gini value and the samples at each node. Figure generated using Scikit learn.

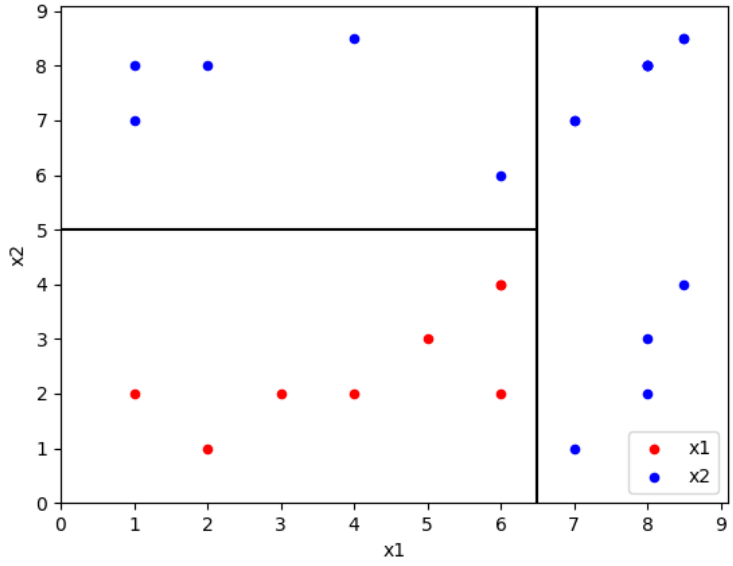


Figure 2.8: Decision Tree feature space partitioning.

$$H(S) = \sum_{x \in X} -p(x,S) log_2(p(x,S)) \tag{2.1}$$

$$IG(S) = H(S) - \sum_{t \in T} p(t,T) H(t) \tag{2.2}$$

where:

S = the data on which entropy will be calculated.
X = the set of classes in dataset S.
$p(x,S)$ = the proportion of class x in dataset S.
$H(S)$ = the entropy value of dataset S.
T = the set of classes in dataset S after the split.
$p(t,T)$ = the proportion of class t in dataset T.
$H(T)$ = the entropy value of dataset T.

The work on ID3 decision trees was improved in C4.5 and C5. C4.5 improves ID3, to handle regression and support post tree creation pruning (Hssina et al., 2014). C5 is second iteration of ID3 that improves scalability and prediction performance due to improved tree creation rules (Hssina et al., 2014).

Breiman et al. (1984) proposed Classification And Regression Trees (CART), that builds a binary tree with the node conditions determined from the training dataset. The resulting tree partitions the decision space into rectangles that are mapped to the response variable y. Figure 2.7 illustrates a decision tree generated for a binary classification problem whilst Figure 2.8 illustrated how the criteria selected by the decision tree partition the feature space.

The process of building the tree can result in a large tree that does not generalise well. CART determines the appropriate size of the tree through cross validation. The tree is pruned using weakest link pruning recursively (Breiman et al., 1984).

Within CART, the node splitting criterion is based on Gini, which as defined in Equation 2.3. Gini impurity measures the mis-classification rate of each class. The feature that contributes most the node impurity is selected as the splitting criteria.

$$I_G(S) = 1 - \sum_{x \in X} p(x,S)^2 \tag{2.3}$$

where:

S = the data on which Gini index will be calculated.
X = the set of classes in dataset S.
$p(x,S)$ = the proportion of class x in dataset S.
$I_G(S)$ = the entropy value of dataset S.

The built decision tree using the defined process increases purity as the tree depth increases. Noisy training data can lead to a different tree topology; however, the predictive performance of the tree is almost the same (Breiman et al., 1984). Breiman (1996) defines classification and regression trees as unstable because a small change can result in a large change in the misclassification rates.

Work carried out by Breiman (1996) aims to improve the performance of classifiers through "Bootstrap Aggregating" or "Bagging". The main idea behind "Bagging" is to train a number of classifiers on independent datasets and aggregate the prediction results through voting for classification or average in regression. "Bagging" is performed as follows:

- From the training set generate DS_n training sets by sampling the complete training dataset with replacement.
- Train n classifiers using training dataset DS_n.

To perform predictions, each classifier generates the prediction based on the tree built. The final of the Bagging classifier is determined through majority voting (Breiman, 1996).

Ensemble methods such as Bagging uses voting to aggregate multiple classifiers in one, improving accuracy. "Random Forest (RF)", presented in Breiman (2001), builds on Bagging to provide several advantages. Breiman (2001) reports that RF is faster to train, relatively more resilient to outliers, computes internal performance metrics based on "unseen" samples (out of bag error) and that training can be parallelised. RF tackles node splits differently from Bagging as at each split it randomly selects a subset of features to consider for the split (Breiman, 2001).

2.7.2 | Support Vector Machines

Support Vector Machine (SVM) is a classifier that uses a hyperplane to separate classes in a two-dimensional space. Training data is used to determine the position of the hyperplane, as per Figure 2.9. Two linearly separable classes can be separated using a linear plane. The edge samples of the two classes are considered to determine the placement of the hyperplane. The samples selected determine the position of the hyperplane and are called support vectors. The support vectors are the red data points illustrated in Figure 2.9.

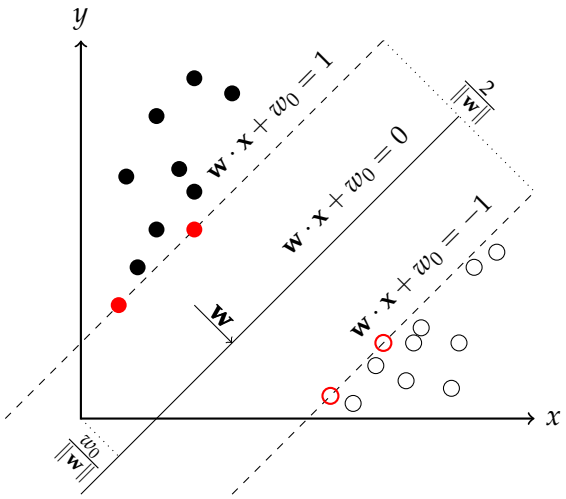


Figure 2.9: SVM hyperplane positioning

The hyperplane is defined by Equation 2.4, such that $g(x) > 0$ is true of one class and $g(x) < 0$ is true of the other (Konstantinos and Sergios, 2008).

$$g(x) = w^T x + w_0 = 0 \tag{2.4}$$

where:

w^T = weight vector.
x = input training vector.
w_0 = the bias parameter.

Multiple solutions exist however SVM improves generalisation of the model by increasing the distance between the edges through a margin. The edges of the margin are used to determine the classes, when Equations 2.5 is true the class is $w1$, and, when Equations 2.6 is true the class is $w2$. For a given point the perpendicular distance from the hyperplane is computed using Equation 2.7.

$$g(x) = w^T x + w_0 \geq \quad 1, \forall x \in w_1 \tag{2.5}$$
$$g(x) = w^T x + w_0 \leq -1, \forall x \in w_2 \tag{2.6}$$
$$z = \frac{|g(x)|}{\|w\|} \tag{2.7}$$

The minimum distance of the sample of a sample of class w_1 from the hyperplane is $\frac{1}{\|w\|}$. Likewise, the distance of minimum distance of a class w_2 from the hyperplane is

also $\frac{1}{\|w\|}$. Therefore, the minimum distance between the samples is as defined in Equation 2.8. To improve the generalisation of the model, this distance must be maximised by, maximising $\frac{2}{\|w\|}$ or minimising $\|w\|$. The minimisation of $\|w\|$ is rewritten as $\frac{1}{2}\|w\|^2$.

$$s(w_1, w_2) = \frac{1}{\|w\|} + \frac{1}{\|w\|} = \frac{2}{\|w\|} \tag{2.8}$$

SVM are also applicable on non-separable classification problems. Slack variables are introduced to relax the classification condition for each class as shown in Equations 2.9 and 2.10. The value of the slack variable is between zero and one for correctly classified samples and less than zero for incorrectly classified samples enclosed between the hyperplane and the support vector of the other class. The minimisation of $\|w\|$ is rewritten as per Equation 2.11 (Konstantinos and Sergios, 2008).

$$g(x) = w^T x + w_0 \geq \quad 1 - \xi, \forall x \in w_1 \tag{2.9}$$

$$g(x) = w^T x + w_0 \leq -1 - \xi, \forall x \in w_2 \tag{2.10}$$

$$\text{minimise } \|w\| = \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{N} \xi_i \tag{2.11}$$

where:

ξ_x = slack variable for class x.
N = number of classes in training dataset.
C = influence control variable.

Linear SVM's enable classification of linearly separable classes thus non-linear relationship cannot be handled. To use SVM's on non-linearly separable classes, the relationship must be linearised and subsequently linear SVM's used to tackle the problem (Shawe-Taylor and Cristianini, 2004). The mapping is performed using a kernel function that maps $x \rightarrow \Phi(x)$ (Konstantinos and Sergios, 2008). The inner product with the kernel function is generally referred to as the "kernel trick". Figure 2.10 illustrates the mapping of non-linear data to linear using the kernel trick.

SVM can be extended to handle multiclass classification using either One vs Rest (OVR) or One vs One (OVO). OVR trains a classifier for each class with the negative samples being the labels of the other classes (Bishop, 2009). OVO tackle multiclass problem differently whereby a classifier is trained against each class, a n class problem requires $n(n-1)/2$ classifiers (Bishop, 2009).

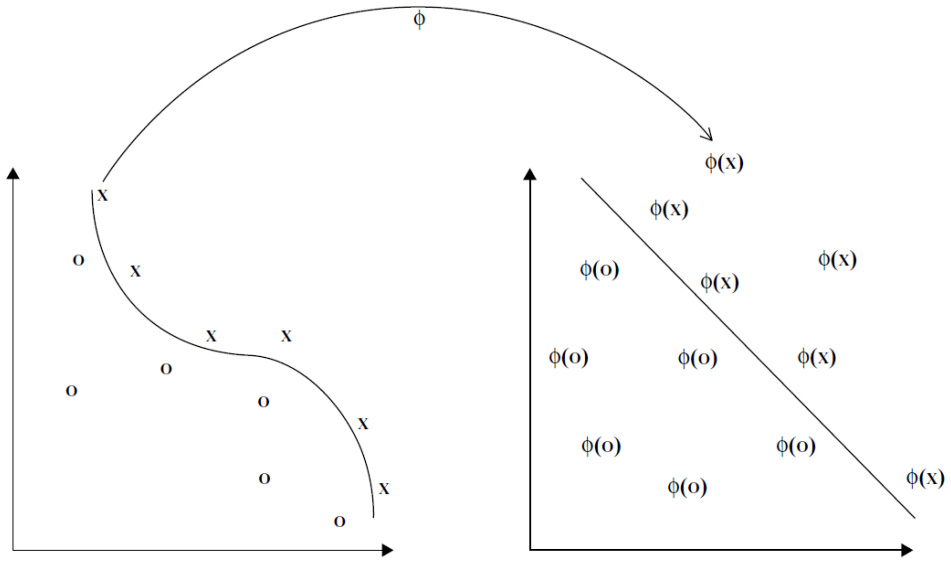


Figure 2.10: Application of SVM classifier for non-linearly separable classes. Data points are multiplied with the ϕ to linearise the relation between the classes. Figure reproduced from Shawe-Taylor and Cristianini (2004)

.

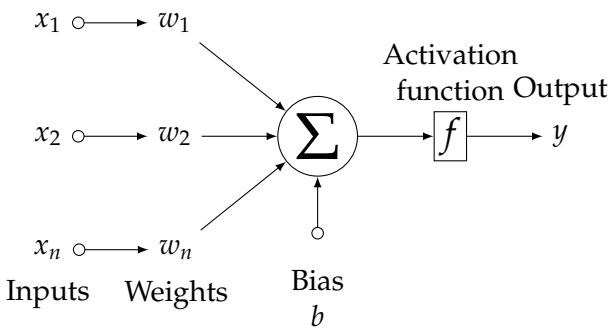


Figure 2.11: Neuron architecture

2.7.3 | Neural Networks

The basic building block of Neural Network (NN) is the neuron. The neuron has an arbitrary number of inputs, for each input, there is a corresponding weight, a bias and has a single output as illustrated in Figure 2.11. To determine the output value, the neuron considers all its inputs, their corresponding weights and bias as per Equation 2.12.

$$Z = \theta(\sum_{i=1}^{N} x_i w_i + b) \tag{2.12}$$

where:

Z = output of the neuron.
N = number of neuron inputs.
x_i = input from previous layer.
w_i = weight for given connection.
θ = the activation function of the neuron.
b = bias of the neuron.

The activation function determines the output of the neuron, typical examples include sigmoid, hyperbolic tangent (Bishop, 2009). Neurons are organised in networks to mimic the learning models found in nature (Mitchel, 1997). The architecture consists of several interconnected neurons, organised in layers such as in Figure 2.12.

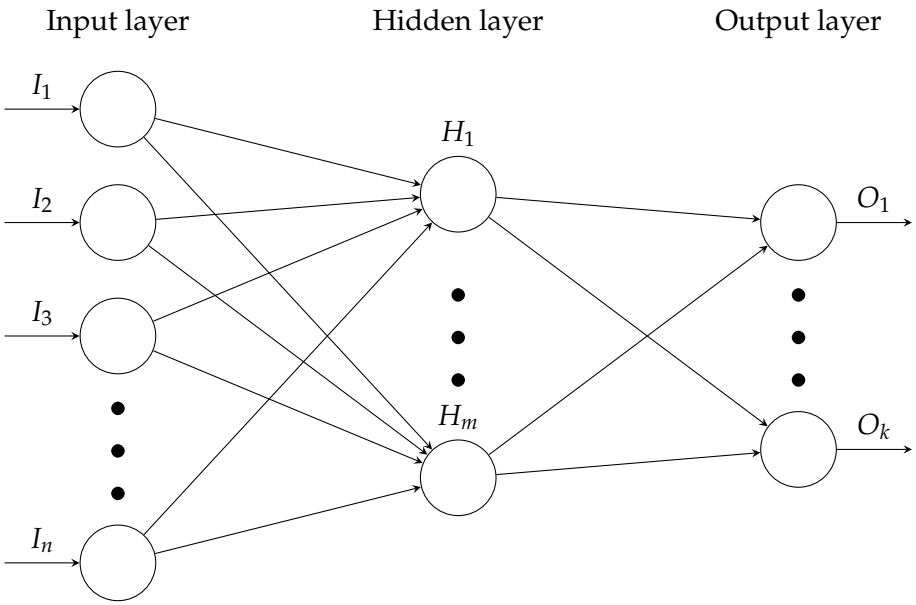


Figure 2.12: Neural network architecture

In the training phase, the input weights are adjusted to minimise network error. The training optimiser implements algorithms to reduce the network error. For classification problems the error is computed using cross entropy as defined in Equation 2.13.

$$R(\theta) = \sum_{k=1}^{K} \sum_{i=1}^{N} y_{ik} log f_k(x_i) \tag{2.13}$$

where:

$R(\theta)$ = cross entropy.
K = number of neurons in the network.
N = number of neuron inputs.
y_{ik} = target output of the neuron.
f_k = activation function of the neuron k.
x_i = input vector.

The computed error is propagated back so that the error of each neuron is determined. The weight adjustments are gradually adjusted, the adjustment rate is governed by the learning rate. This value determines the amount of error that is considered when adjusting the weights. A high learning rate value will cause the network to move rapidly on the error surface but will not find the minima due to large adjustments. On the other hand, small learning rate values will cause an increase in the number of epochs required for training (Orr and Müller, 1998).

Gradient descent is one of the algorithms used in training that aims minimise error. This area is being actively researched especially for deep networks. The traditional approach in gradient descent is to evaluate the gradient after each epoch and apply adjustments limited by the learning rate to the weights (Ruder, 2016). Stochastic gradient descent does not consider all the training data in the epoch but takes random samples for the training dataset. Ruder (2016) reports that this technique reduces training time due to a small training dataset but a decay in the learning rate increase the likelihood of finding the minima. A further improvement on gradient descent uses mini-batch to speed up training. The training dataset is shuffled at each epoch and mini batches are extracted. For each batch, the gradient is computed, and weights are adjusted (Ruder, 2016).

Active research on techniques to speed up identification of global minima resulted in different proposals. Ruder (2016), reports that Ning Qian proposed momentum to incorporate a fraction of the previous adjustment in the current, to escape local minima. Kingma and Ba (2014) proposed the ADAM optimiser (ADAptive Moment) that incorporates adaptive momentum and variable learning rates to speed up the learning process.

The input vector of the output layer of a NN is a set of discrete values z. Some applications of NN require the prediction confidence. For this purpose, vector z can be converted to a vector of probabilities using the SoftMax function. The properties of the output vector are that the summation of the vector is equal to one and that each value is between zero and one (both values inclusive) (Goodfellow et al., 2016). The SoftMax function is defined as per Equation 2.14.

$$SoftMax(z)_i = \frac{exp(z_i)}{\sum_1^j exp(z_j)} \tag{2.14}$$

where:

z = the vector to discrete values to be converted to probabilities.
i = the current element in z.
j = the size of vector z.

2.7.4 | Machine Learning Performance Evaluation Criteria

The performance classifier must be evaluated using metrics that permits comparison against previous runs or other systems. The output of a classifier can be categorised using a confusion matrix as shown in Table 2.1.

		Actual Value	
		Positive	Negative
Predicted Outcome	Positive	True Positive (TP)	False Negative (FN)
	Negative	False Positive (FP)	True Negative (TN)

Table 2.1: Machine Learning results types

Accuracy, as defined in Equation 2.15 determines the number of correct predictions of a classifier. This metric can be misleading on a skewed dataset and in cases where the classifier performs well for the over-represented class (Kubat, 2015).

Precision (Pr) defined in Equation 2.16, returns the percentage of correctly predicted positive samples out of the predicted positive samples. Recall (Rc) defined in Equation 2.17 returns the percentage of actually positive samples to be correctly classified. To facilitate performance comparison, precision and recall can be combined in a single metric, F_{max} using Equation 2.18 (Kubat, 2015). The balanced F_{max} ($\beta = 1$) is the harmonic mean of precision and recall values.

$$A_{cc} = \frac{N_{TP} + N_{TN}}{N_{TP} + N_{TN} + N_{FP} + N_{FN}} \tag{2.15}$$

$$Pr = \frac{N_{TP}}{N_{TP} + N_{FP}} \tag{2.16}$$

$$Rc = \frac{N_{TP}}{N_{TP} + N_{FN}} \tag{2.17}$$

$$F_{max} = \frac{(\beta^2 + 1) * Pr * Rc}{\beta^2 * Pr + Rc} \tag{2.18}$$

where:

N_{TP} = number of positive samples correctly classified
N_{TN} = number of negative samples correctly classified
N_{FP} = negative samples classified as positive samples
N_{FN} = positive samples classified as negative samples
β = $\beta < 1$ prioritise recall, $\beta > 1$ prioritise precision, $\beta = 1$ harmonic mean.

2.7.5 | Cross Validation

In ML generalisation performance describes the performance of the model on unseen data. A model that performs poorly on unseen data does not generalise well. To estimate the ability of the ML model generalisation, a benchmark on unseen data is required.

To determine the generalisation of a model, the labelled dataset is split into two datasets, the training and testing dataset. The training dataset is used to train the ML model whilst the test dataset is used to determine the performance of the ML model. The labels of the test dataset are the true values, whilst the predicted values are outputted by the ML model based on the testing data. The true and predicted values are used to determine the performance of the model.

$$EPE_k(x_0) = E[Y - \hat{f}_k(x_0)]^2 \tag{2.19}$$

$$= \sigma^2 + Bias^2(\hat{f}_k(x_0)) + Var_\tau(\hat{f}_k(x_0)) \tag{2.20}$$

$$= Irreducible\ Error + Bias^2 + Variance \tag{2.21}$$

where:

$\hat{f}_k(x_0)$ = the model fitted to map $f_k(x_0)$

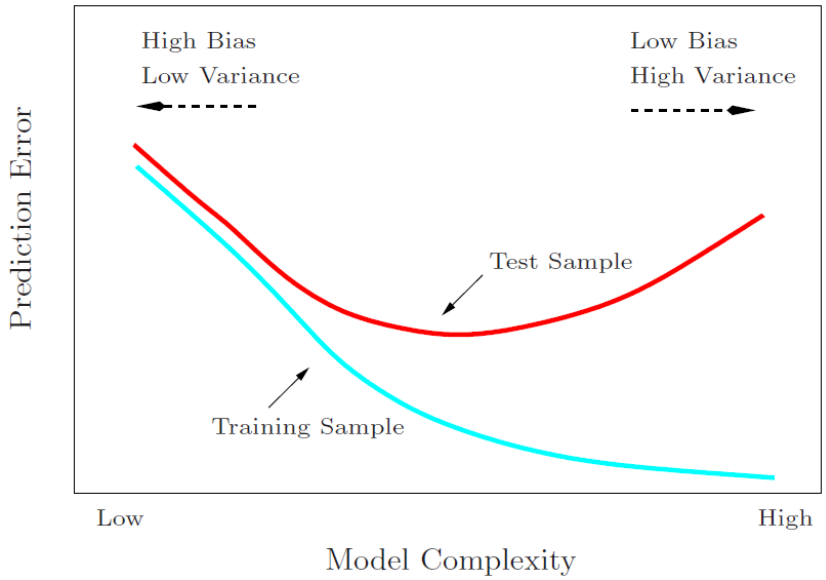


Figure 2.13: Prediction error in relation to ML model complexity, figure reproduced from Hastie (2009).

The expected prediction error can be decomposed in three components are per Equations 2.19, 2.20 and 2.21 (Hastie, 2009). The first component is the irreducible error that is the error part that cannot be controlled. The second component is the Bias, which is defined as the amount by which the mean predictor estimate differs from the mean true value. The third component is the variance, which is defined as the deviation of the predicted output from its mean value.

The complexity of the ML model governs the richness of the model. The richer the model, the more information from the training data can be extracted. The downside of this is that the model performs exceptionally well on the training data but does not generalise well. Figure 2.13 illustrates the general relation between the ML model complexity and prediction error. Low complexity modes tend to have high bias as the information extracted from the training data is limited. On the other hand, high complexity models adapt to the training data too much thus causing the training performance (shown in cyan) to be good and generalise poorly (show in red). The optimal model complexity for a given problem is a trade-off between bias and variance which is identified by experimentation.

Cross validation is a technique used to estimate the prediction error. The leave one out cross validation estimates the prediction error based on the left-out data partition. The application of the leave one out cross validation is illustrated in Figure 2.14. The

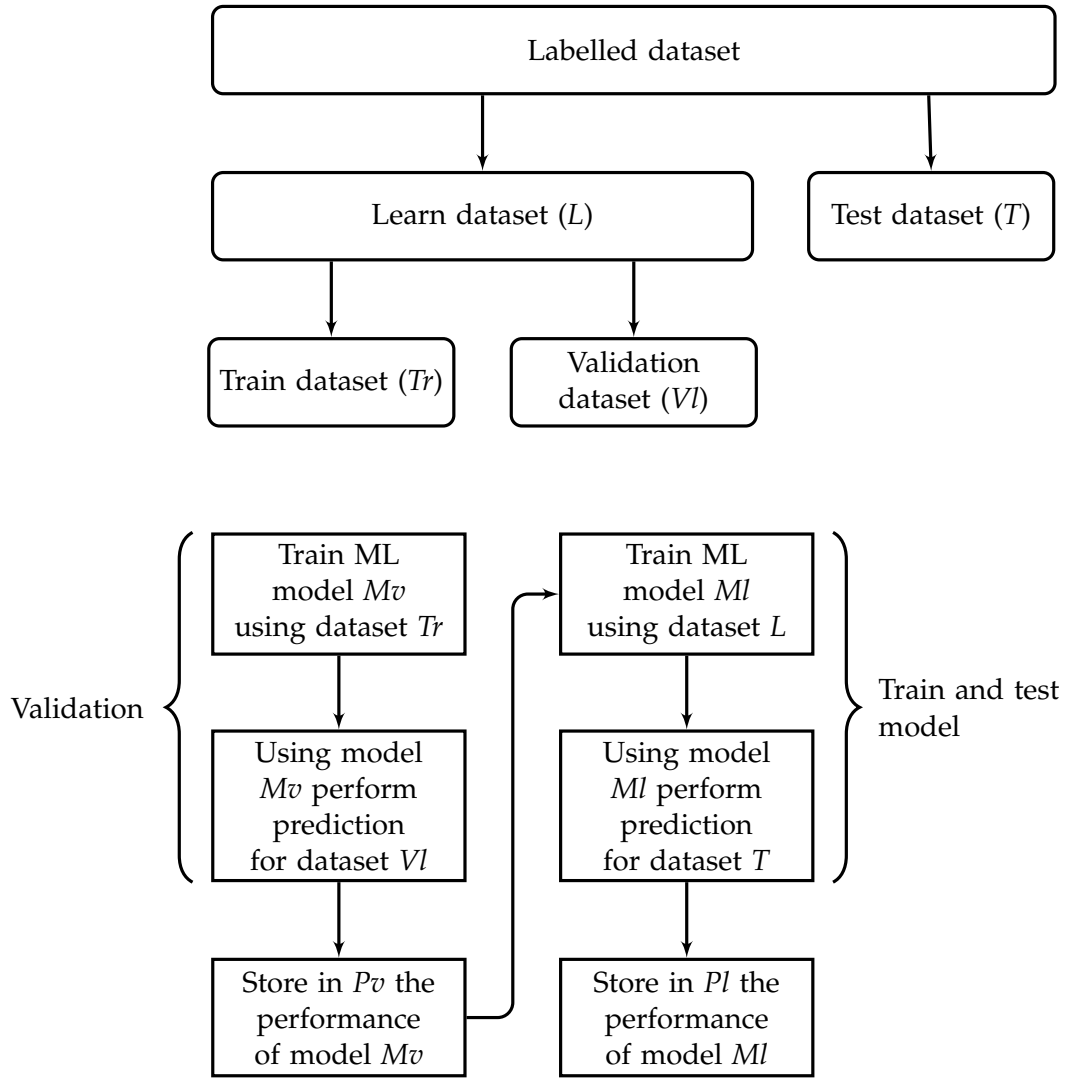


Figure 2.14: ML cross validation and training. The labelled dataset is split into the learn *L* and test *T* datasets. The learn dataset is in turn split into train *(Tr)* and validation (*Vl*) datasets that are used for cross validation. The *Tr* dataset is used to train the model and predictions are generated for dataset (*Vl*). Dataset (*L*) is used to train the model and generate predictions for the test (*T*)dataset. The performance of the models on datasets (*Vl*) and (*T*) must be similar. Figure adapted from Simon (2007).

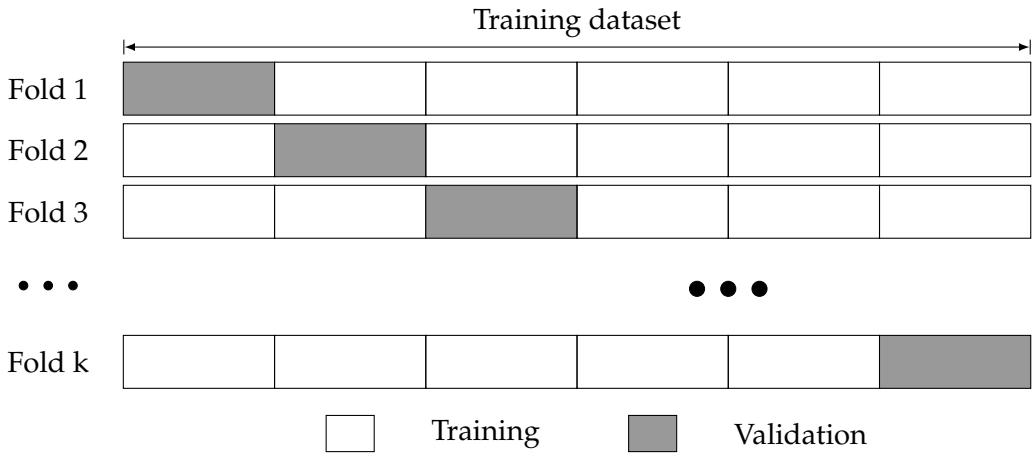


Figure 2.15: K-fold cross validation dataset splits across the different folds. Each fold partitions the training set in two non-overlapping datasets used to train and validate ML model performance.

labelled dataset is split into two partitions, the learn (L) and test (T) datasets. The learn dataset is split into training (Tr) and validation (Vl) datasets as illustrated in the top part of Figure 2.14. The ML model Mv is trained on the train dataset (Tr) and evaluated on the validation (Vl) dataset. Subsequently the ML Ml is trained on the learn (L) dataset and evaluated on the test (T) dataset. Both classifier Mv and Ml have been evaluated on dataset that were unseen by the classifier. The ML classifier is performing as expected when the performance of the two classifiers is similar.

The leave one out cross validation relies on a single dataset partition and is susceptible to variance (Hastie, 2009). A more robust way to estimate the prediction error is to utilise different validation partitions and use the mean value as the estimated error. Another advantage of this approach is that generally the amount of labelled data available is limited and does not permit generating three datasets to assess model performance.

The k-fold cross validation is a type of cross validation that splits the training dataset into K splits. K training iterations are performed each time using different training and validation partitions as illustrated in Figure 2.15. The estimated prediction performance of the classifier is computed as the mean error computed across the K iterations. The mean value across all folds ensures that the value obtained is not an outlier obtained from a specific split (Hastie, 2009).

K-fold cross validation technique requires K training iterations, increasing the training time and resources required to train the classifier. Each iteration uses (K-1) partitions for training and one partition for validation as illustrated in Figure 2.15 (Hastie, 2009).

2.8 | Feature Selection

Feature selection is the process of selecting features that have correlation with the output variable. A training dataset has several features that describe the instance and the output variable. Correlation is a measure of association between the feature being investigated and the output variable. The correlation between variables can be strong, weak or irrelevant. Strongly correlated values can be used to can be used to predict each other. Weakly correlated variables contribute limited information about the other variable. Irrelevant correlation implies that the variable is adding redundant information or just noise (Bonev, 2010).

Feature selection entails selection process to identify the subset of strongly correlated features to be used in ML. The reduced subset of features increases the ML efficiency as the problem complexity (dimensionality) is reduced. Another key aspect of feature selection is that discarded features are not considered by the ML model, thus do not need to be captured or computed/generated.

2.8.1 | Manual Feature Selection

Manual feature selection requires manual intervention to determine the features to use in ML. A feature is selected on domain knowledge or on the outcome of statistical analysis on the data. The statistical analysis considers aspects such as low variance whereby a feature that has the same value for a high percentage of the data (Li et al., 2017a). Typical examples of low variance fields are flags associated with processes used to generate the data or flags in data related to categories.

Expert domain knowledge is central in manual feature selection. Through expert domain knowledge, dataset features are understood in depth and non-trivial links between features is learnt. Using the expert knowledge, noisy and low information features can be discarded.

Manual feature selection can utilise ML techniques to identify features that have high correlation with the output variable. Viewing the model of build decision tree can evidence important features.

2.8.2 | Automatic Feature Selection

In machine learning, dataset features are used to train a model which in turn, makes predictions based on the same features. The input feature set effects the predictive power of the model built. An exhaustive search for the best features in a dataset with n features

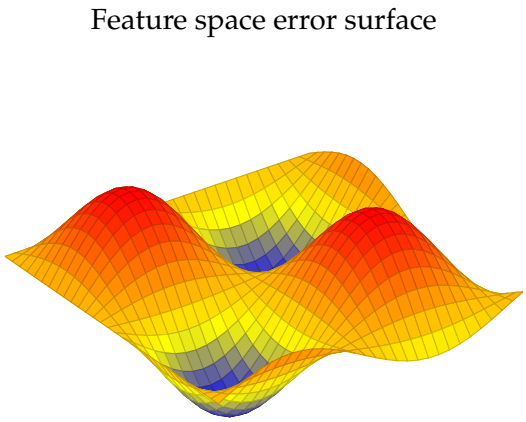


Figure 2.16: Error surface across the feature space.

requires attempting 2^n combinations. GA is an optimisation technique that explores the feature space to find the best feature set mimicking biological evolution (Mitchel, 1997).

The error surface of a ML classifier needs to be mapped out through experimentation. The error surface consists of a number of maxima and minima. Figure 2.16 illustrates a typical error surface for a group of features. A difference between local and global minima is that the global minima is the lowest possible point on the error surface. A local minimum is a minimum on the error surface, that is not the global minimum. Genetic Algorithm (GA) performs feature space exploration to identify a minimum. The GA does not guarantee finding the global minima (Sastry et al., 2014).

Figure 2.17 illustrates in detail the workings of a typical GA. The GA chromosome pool is evaluated using the fitness function (Haupt and Haupt, 2004). This function is agnostic from the GA implementation. The fitness score is used the GA to identify best performing chromosomes. Selecting the best performing chromosomes increases the likelihood of producing better off springs (Haupt and Haupt, 2004).

The best chromosomes are crossed over and mutated to generate off springs which are in turn evaluated using the fitness function. Through this mechanism the GA introduces randomness that enables it to explore different regions of the features space.

Chromosome crossover requires two source chromosomes to generate two destination chromosomes. Sections of the source chromosomes are taken and assembled into new chromosome using pre-define masks. The masks define the segment to be copied to the target chromosome. Mitchel (1997) describes three crossover masks: single point, two point and uniform crossover masks. The variations between these masks is the offset and number of bits copied (Haupt and Haupt, 2004; Mitchel, 1997). Figure 2.18, illustrates the single point crossover.

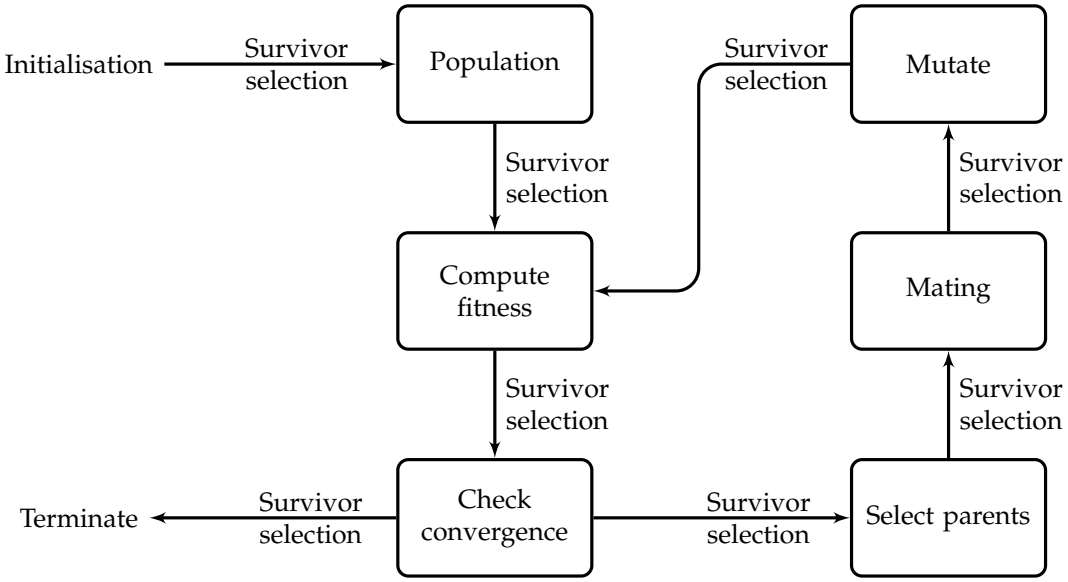


Figure 2.17: Genetic Algorithm flow chart. The flow chart illustrates the iterative process of generating a chromosome population, checking fitness. The best chromosomes are bred and mutated and re-evaluated. This process is repeated until the required fitness is reached.

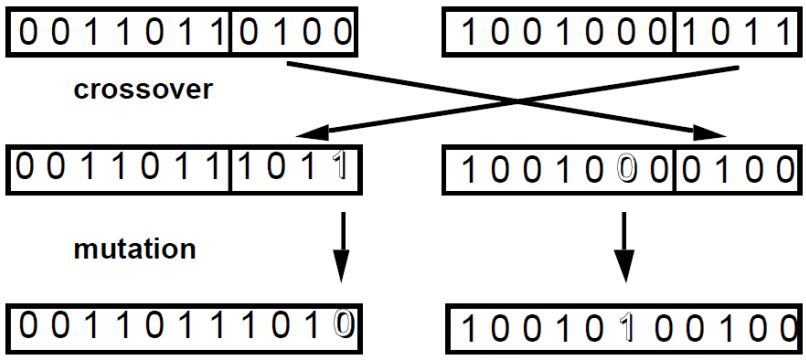


Figure 2.18: Genetic algorithm single point crossover and mutation, figure reproduced from Comellas and Ozón (1995)

Chromosome mutations introduces random changes on the chromosome to explore the feature space. The mutation process entails flipping bits are pre-defined positions in the chromosome (Haupt and Haupt, 2004; Mitchel, 1997). Figure 2.18 illustrate a single bit mutation.

The termination criteria of the GA are an arbitrary decision that considers two criteria. The first criterion is the definition of a fitness threshold. This threshold requires knowledge of performance figures achievable with the dataset being utilised. The second criterion is a pre-defined number of epochs to be executed. This parameter controls the maximum number of population generations that will be evaluated.

Each chromosome in the pool is evaluated using the provided "cost" or "fitness" function (Haupt and Haupt, 2004). The fitness function is used to compute the fitness value for each chromosome. The GA determines whether the termination criteria have been reached or more iterations are required. The best performing chromosomes are crossbred and mutated to form the new chromosome population. Breeding "good" chromosomes increases the likelihood of producing better off springs (Haupt and Haupt, 2004).

2.8.2.1 | Cross Validation

In automatic feature selection, the GA chromosomes are ranked on the performance score obtained on the unseen dataset. The performance of the ML classifier can be dependent on the training and unseen data used. Consequently, dataset variations can affect the overall outcome of the GA (Kuhn and Johnson, 2013).

K-fold cross validation was introduced as a technique that estimates the prediction error of a classifier. The estimate was made more robust by computing the error estimates as the mean of the values obtained in the different folds.

This technique can be combined with GA to determine the performance variation generated by using different training and unseen (validation) datasets. An overarching principle when combining these two techniques is that all decisions based on the output variable must be performed within the cross validation loop (Hastie, 2009). For example, if a subset of chromosomes were discarded prior to running feature selection, this would improve the likelihood of the selecting better chromosomes, as the bad ones were discarded on the outset. Such circumstances positively skew the performance estimate of the GA by improving the chromosome pool of the GA.

The GA is filtering chromosomes based on the performance score computed by the fitness function. The fitness function is using output variable to compute a score of the chromosome. Effectively if placed outside the cross validation loop it violated the prin-

cipal that all decisions on the output variable must happen within the cross validation loop.

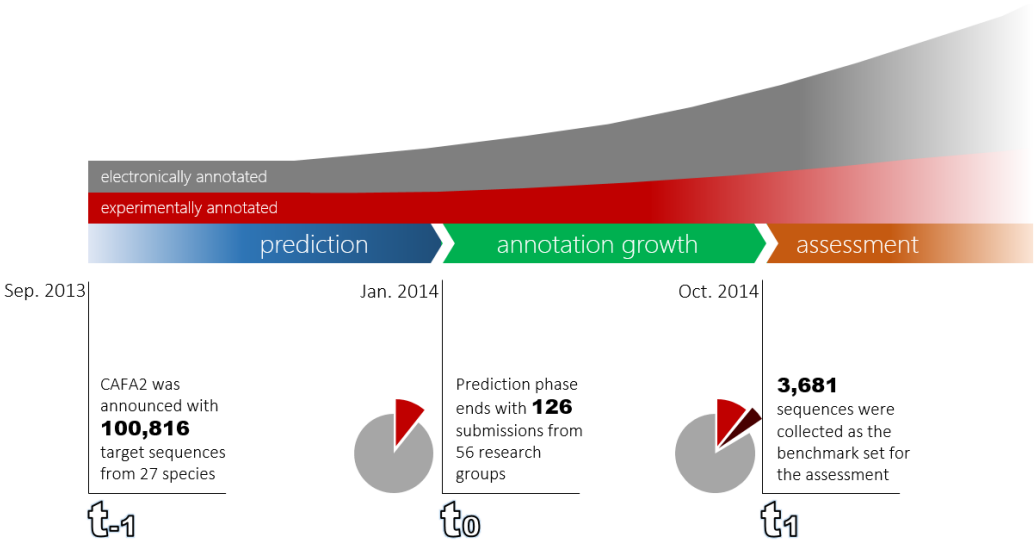


Figure 2.19: CAFA shared task schedule, reproduced from CAFA website[5]. At t_{-1}, CAFA organisers select a number of proteins that do not have laboratory annotations. Participants have 3 months to submit their predictions for the selected targets, until t_0. At t_0, the shared task pauses for a number of months, to allow time for targets to gain laboratory annotations. Subsequently, at t_1 in the evaluation phase, submissions are evaluated against proteins that acquired laboratory curation.

2.9 | CAFA Shared Task

Within the scientific community, scientists are developing different systems in the attempt to reduce the accumulation of unannotated protein sequences found in databases such as TrEMBL. To determine the most appropriate methods that are up for the task, a standard benchmark is required. Friedberg and Radivojac (2017) proposed a community challenge that triggers networking between participants and provides a solid evaluation framework.

The Critical Assessment of Functional Annotation (CAFA) shared task is a time-based and provides participants with a training dataset. Participants must train their models and submit predictions of the identified targets.

CAFA shared task has three phases as shown in Figure 2.19. The first phase is the prediction phase. At the beginning of this phase, a set of unannotated proteins is se-

[5] `http://biofunctionprediction.org/cafa/` (Accessed 2018-12-28)

lected and defined as CAFA targets. Participants have until the end of the prediction phase to submit their predictions. CAFA enters the annotation phase, whereby the organisers wait for a subset of the selected targets to be laboratory curated. After a number of months, the assessment phase starts. The evaluation of the submission is executed on those targets that have received laboratory annotation in phase 2.

2.9.1 | Performance Metrics Used in CAFA

Clark and Radivojac (2013) proposed a framework that models a Bayesian network with prior distributions on the GO DAG. The computation of prior distributions references a database of annotated proteins such as SwissProt. Using the annotations found in the dataset, marginal probabilities and information content for each term are computed using Equation 2.22.

The "information accretion" is utilised to compute two metrics, namely remaining uncertainty and misinformation as defined in Equations 2.23, 2.24 respectively. The remaining uncertainty is information that is yet to be predicted when compared to the true positive set, whilst "misinformation" is terms that were incorrectly predicted. To facilitate ranking and evaluation of function prediction methods these metrics are combined into "Semantic distance" defined in Equation 2.25. "Semantic distance" is the minimum distance between the origin and the curve $(ru^k(\tau) + mi^k(\tau))_\tau$. The preferred distance metric is Euclidean distance attained by using $k = 2$.

$$ia(v) = \sum_{v \in T} log \frac{1}{Pr(v|\mathcal{P}(v))} \tag{2.22}$$

$$ru(T,P) = \sum_{v \in T-P} ia(v) \tag{2.23}$$

$$mi(T,P) = \sum_{v \in P-T} ia(v) \tag{2.24}$$

$$S_k = \min_\tau (ru^k(\tau) + mi^k(\tau))^{\frac{1}{k}} \tag{2.25}$$

where:

v = is a node in the graph.
$\mathcal{P}(v)$ = is the set of parent nodes of v.
T = is the true positive sub graph.
P = is the predicted function sub graph.

2.9.2 | CAFA Evaluation Criteria

The CAFA evaluation utilises set based and information theoretic metrics based on Equations 2.23, 2.24, 2.25 (Jiang et al., 2016). Set based metrics are computed using Equations 2.26, 2.27, 2.28, whilst information theoretic are computed using Equations 2.29, 2.30, 2.31. CAFA submissions are evaluated using F_{max} and S_{min} performance metrics.

$$pr(\tau) = \frac{1}{m(\tau)} \sum_{i=1}^{m(\tau)} \frac{\sum_f \mathbb{1}(f \in P_i(\tau) \wedge f \in T_i)}{\sum_f \mathbb{1}(f \in P_i(\tau))} \tag{2.26}$$

$$rc(\tau) = \frac{1}{n_e} \sum_{i=1}^{n_e} \frac{\sum_f \mathbb{1}(f \in P_i(\tau) \wedge f \in T_i(t))}{\sum_f \mathbb{1}(f \in T_i(t))} \tag{2.27}$$

$$F_{max} = \max_{\tau} \frac{2.pr(\tau).rc(\tau)}{pr(\tau) + rc(\tau)} \tag{2.28}$$

$$ru(\tau) = \frac{1}{n_e} \sum_{i=1}^{n_e} \sum_f ic(f).\mathbb{1}.(f \notin P_i(\tau) \wedge f \in T_i) \tag{2.29}$$

$$mi(\tau) = \frac{1}{n_e} \sum_{i=1}^{n_e} \sum_f ic(f).\mathbb{1}.(f \in P_i(\tau) \wedge f \notin T_i) \tag{2.30}$$

$$S_{min} = \min_{\tau} \sqrt{ru(\tau)^2 + mi(\tau)^2} \tag{2.31}$$

where:

$P_i(\tau)$ = the set predicted with score equal or greater than τ for a protein sequence.
T_i = the true positive set the sequence.
$m(\tau)$ = the number of sequences with at least one score greater than or equal to τ.
$\mathbb{1}$ = the indicator function that returns 1 when the condition is true otherwise 0.
n_e = the number of targets in evaluation.
$ic(f)$ = the information content for term f.
τ = a value for 0.01 to 1.00 incremented by 0.01.

	Annotations									
Protein	A	B	C	D	E	F	G	H	I	J
P1	1	1	1	1	1	1	0	0	0	0
P2	1	1	1	0	1	0	0	0	1	0
P3	1	1	1	1	1	0	0	1	0	1
P4	1	1	1	1	0	0	1	1	0	0

Table 2.2: Listing of protein annotations.

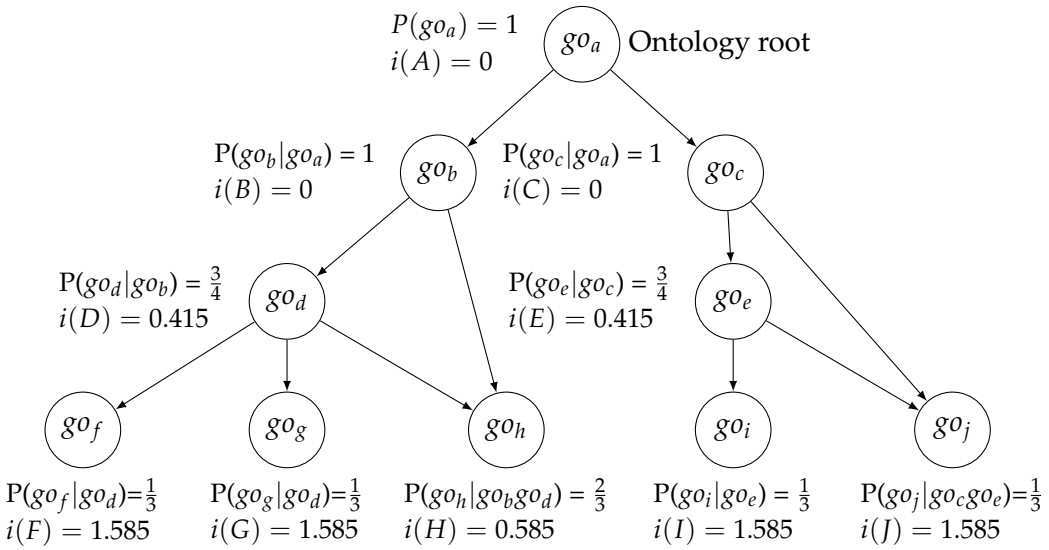


Figure 2.20: Information content computed for the proteins listed in Table 2.2. Conditional probabilities are computed and subsequently the information content is computed using Equation 2.22.

Protein	True Terms	Predicted Terms	RU	MI
P1	ABCDEF	ABCDG	EF	G
P2	ABCEI	ABCDEI	-	DI
P3	ABCDEHJ	ABCDEGJ	H	G
P4	ABCDGH	ABCDGH	-	-

Table 2.3: Predictor predictions, ground truth and information content term sets. Information content term sets contain the term set for Remaining Uncertainty (RU) and Mis-Information (MI).

Protein	i(RU)		i(MI)	
P1	$\frac{i(EF)}{i(ABCDEF)}$	0.828	$\frac{i(G)}{i(ABCDEF)}$	0.626
P2	$\frac{i(-)}{i(ABCEI)}$	0	$\frac{i(DI)}{i(ABCEI)}$	0.415
P3	$\frac{i(H)}{i(ABCDEHJ)}$	0.195	$\frac{i(G)}{i(ABCDEHJ)}$	0.58
P4	$\frac{i(-)}{i(ABCDGH)}$	0	$\frac{i(-)}{i(ABCDGH)}$	0

Table 2.4: Semantic distance workings.

2.9.2.1 | Worked example

For illustration purposes, this subsection presents a worked example of the CAFA metric for a classifier. The method consists of two steps, namely;

1. Compute the information content of each node in the ontology using manually curation proteins. Using the annotation table is provided in Table 2.2, the information content is computed as shown in Figure 2.20.

2. For each protein the set of true term, predicted term, remaining uncertainty and misinformation are determined as shown in Figure 2.3.

The remaining uncertainty and misinformation is computed with information content values from Figure 2.20 and terms list from Table 2.2. Equation 2.32 shows the working to compute the remaining uncertainty based on Equation 2.29. Similarly, Equation 2.33 shows the working to compute the misinformation based on Equation 2.30. The metrics are combined into the semantic distance as per Equation 2.34, based on Equation 2.31.

$$ru = \frac{0.828 + 0 + 0.195 + 0}{4} = \frac{1.023}{4} = 0.255 \tag{2.32}$$

$$mi = \frac{0.626 + 0.415 + 0.580 + 0}{4} = \frac{1.621}{4} = 0.405 \tag{2.33}$$

$$sd = \sqrt{0.255^2 + 0.405^2} = 0.478 \tag{2.34}$$

2.10 | Related Work

Protein function prediction is an open problem being actively researched by scientists from different research groups. Researchers used different approaches to tackle the problem. The following review outlines the main techniques used.

2.10.1 | PPI-network

The main rationale behind using Protein-Protein Interaction (PPI) in protein function prediction is that those interacting proteins are likely to exhibit the same functionality (Deng et al., 2003; Schwikowski et al., 2000). Different research groups adopt different PPI based approach to tackle protein function prediction.

PPI networks can be viewed as a connected graph whereby proteins are represented by vertices and interactions are represented by edges. Protein functionality is a property of the protein vertices. The work proposed by Schwikowski et al. (2000) established a functional relation through direct links between the protein and its neighbour. Hishigaki et al. (2001) externalised the functionality to a vertex. Through this change in the PPI structure, the annotation process considers the number of proteins having a specific GO term. The idea of the ratio is extended to probability through Markov random fields (MRF) in the work proposed by Deng et al. (2003). This work computed two values to determine whether to annotate a protein, namely occurrence of the GO term in the PPI and probability obtained from the MRF.

PPI network connectivity level varies between heavily active and less active proteins. Connectivity metrics can be achieved through clustering algorithms. Spectral clustering is a type of clustering that identifies, and groups, strongly connected vertices in the graph. The work proposed by Xiong et al. (2014) uses spectral clustering to determine the minimum set of annotated proteins required to perform protein function prediction. To work used spectral clustering on the PPI network. For each strongly connected cluster, the central node is determined using graph closeness, betweenness and degree to determine the important node in the cluster (Xiong et al., 2014).

Functionality transferred using neighbourhood metrics on a PPI network may not apply for the target species. The work reported by Moosavi et al. (2013) tackles this specific aspect. For target species, GO term specific scores are computed to determine whether a given protein has the specific functionality (GO term).

The concept of network neighbourhood can be extended from a direct connection to a path. This shift in neighbourhood definition requires the use of graph concepts. A random walk from a given node explores all the paths that start from the given node with a specified length. Random walks are a central tool to measure similarity in the work proposed by Cao et al. (2013). The main rationale in this work is that similarity is measured as the number of times a specific vertex is traversed in random walks of a specific length. Higher value implies that the proteins are similar, thus annotation transfer can be performed. Random Walks on PPI were utilised to add missing annotations. Yu et al. (2015) proposed a downward Random Walk on the PPI network. Similar proteins

are identified using the semantic similarity measure and a probability value is used to determine the walk stopping criteria.

Random Walks explore the network for reachability. However, it does not compute relevance. An algorithm that identifies important vertices in a graph is Google's PageRank algorithm. The PageRank algorithm computes the importance of web pages by using the quality of the links and the number of inward and outward links (Page et al., 1999). Freschi (2007); Jiang et al. (2017) proposed PageRank based methods that rank proteins for a specific GO term. To rank proteins for a specific GO term, a subgraph with all proteins and a specific GO term is extracted from the PPI. PageRank is executed to rank proteins according to GO connectivity. This process is repeated for all GO terms.

Similarity measurement can be tackled by projecting features in a lower dimensional space and measure the distance between the points. This approach was performed on PPI data by Wang et al. (2015, 2017). In the work proposed by Wang et al. (2015) to predict rare protein functions, Random walks are used to garner information from two PPI networks. The information extracted from the PPI networks is projected into a lowered dimensional space. "Nearness" is the measurement used to assess similarity. In the proposed by Wang et al. (2017) multiple data sources including PPI, gene ontology, co-expression, text mining and homology are utilised. The combined dataset is mapped to a low dimensional space through dimensionality reduction. Inter and intra species scores are computed, to enable annotation transfer across species. The query sequence is processed and mapped to the feature space amongst the other sequences. Cosine distance is used to find the nearest neighbour.

The criterion to determine whether the annotation should be made can be determined from multiple sources. Cao and Cheng (2016) proposed Statistical Multiple Integrative Scoring System (SMISS) that uses three different data sources to perform protein function prediction. The first component is determined by performing a PSI-BLAST search against SwissProt. For each GO term, a probabilistic score is computed based on PSI-BLAST e-value. The second component is based on interaction scores computed on multiple PPI networks. The GO term annotation probability is computed on the interaction score. The third component is computed on the relation between sequences residues and GO terms. The link between sequence and GO term is computed by linking a moving window on the protein sequence to GO term. The final probability is computed based on the three components with different weights.

Laboratory curation methods are used to determine the function of a protein, yet the research goal of the laboratory can be targeting a specific area thus the annotations generated are correct but incomplete (Yu et al., 2015). Sun et al. (2018) proposed a different approach to build PPI networks to address some deficiencies of the PPI network.

The highlighted deficiencies are a consistent number of false positives that reduce the efficacy of the network, missing interactions and the shortage of annotations. To address these issues a new approach to build PPI network is proposed. The new PPI is a weighted PPI, the first component of the weight is based on the interactions from the STRING database whilst the second component is a computed neighbour ratio. Additional edges are added to similar sequences identified using BLAST. Sequence GO terms are added to the interaction using the interaction factor computed on BLAST.

2.10.2 | Homology

Homology-based methods, transfer functionality through ancestry relationship. The homology relationship is determined through sequence alignment using tools such as BLAST (Altschul et al., 1990) or MAFFT (Katoh et al., 2005). These tools identify the conserved part of the protein, providing sequence match location and length.

Homology-based methods transfer functionality based on similarity, on the assumption that there is sufficient confidence in the source annotation. The work proposed by Jones et al. (2008) factors in the confidence of the source annotation in the annotation transfer. The work uses a dataset that contains only curated annotations and filters out annotations propagated through sequence similarity. For each GO term, the best five representative sequences were kept. A decision tree was utilised to determine the confidence score for the different annotations. In the prediction phase, a BLAST search is performed, and the confidence score is used to determine whether the specific GO term should be outputted.

The mechanism to determine the annotation confidence can utilise parameters such GO term co-occurrence. The work proposed by Hawkins et al. (2009) uses sequence similarity and GO term co-occurrence to determine the likelihood of a protein having a specific GO term. The query sequence is aligned against sequences having a specific GO term to determine the similarity degree. For each GO term, the likelihood that it occurs with other GO terms is computed. The two values are used to compute the likelihood of a specific annotation. The notion of testing annotation confidence was tackled statistically by Koskinen et al. (2015). For a given query, sequence alignment is performed against the UniProt protein database keeping only hits with predefined criteria, such as sequence identity percentage and sequence length. The hits are re-scored to factor in taxonomic distance between species, the nearer the species is the more relevant the annotation.

Annotation confidence can be increased if the annotation confidence is determined through a group of sequences. Bartoli et al. (2009) proposed a method that builds clus-

ters of proteins that have a sequence identity greater than 40% and coverage greater than 90%. A protein sequence can be a member of one cluster. UniProt protein annotations are linked to individual proteins. Statistical tests (using p-value) are utilised to determine whether all of its members should have the GO term. Subsequently, two improvements were made: update clusters to use sequences from UniProt 2011 and added an HMM profile for each cluster (Piovesan et al., 2011). The use of HMM profiles facilitates query sequence membership identification. The clusters were recently updated by to use UniProt 2016 (Profiti et al., 2017).

Sequence similarity (homology) based methods require sequence identity of above 30% identity (similarity) to be effective. The twilight zone is just within the limits of homology-based methods (20-35%). The work proposed by Wass and Sternberg (2008) performs protein function prediction method targeting the twilight zone. Position Specific Scoring Matrix (PSSM) is a technique to match a given sequence against a consensus sequence. For each GO term, a PSSM (profile) is built. In the prediction phase, the unknown sequence is aligned with the PSSMs' of the GO terms and the prediction is determined by the outcome of the alignment.

A sequence can be composed of multiple domains. Aligning multiple proteins that have the same domain will generate the consensus sequence of the domain and possibly including other residues. Working at the domain level will decrease noise and possibly improve performance. Lee et al. (2009) proposed "GeMMA" framework to classify proteins into functional clusters. With evolution, the protein sequence mutates, however, the folded structure of the protein tends to be more stable (Lee et al., 2009). GeMMA works at the domain level to enable handling of multidomain proteins and caters for domain shuffling. GeMMA generates several functional clusters, each one represented by the consensus sequence of the sequences in that cluster.

Rentzsch and Orengo (2013) proposed a framework that maps functionality to protein domains based on "GeMMA". The clustering protocol is modified to ensure that each cluster has least 90% similarity between sequences. Clustering stops when only one cluster remains. A quality control check drops any non-functionally coherent clusters. Each cluster (family) is associated with a list of functionalities and probabilities computed on its members. To perform annotation prediction, the target sequences are scanned against a superfamily HMM. The top matching hit is used to determine the list of annotations together with the probabilities. The aspect of cluster functional coherence is tackled by the work of Das et al. (2015b). This work adds functional coherence checks at the stage of cluster merging by similarity.

2.10.3 | Machine Learning

Machine Learning (ML) is a set of techniques that learn from the training data provided. These techniques have been applied in various fields including protein function prediction. This subsection reviews, the work applying ML techniques to tackle protein function annotation prediction.

The Support Vector Machine (SVM) classifier was utilised in various protein function prediction methods. SVM is an ML technique utilised in the three iterations of FFpred (Cozzetto et al., 2016; Lobley et al., 2008; Minneci et al., 2013). The first version of FFred targets a specific set of GO terms originating for the Human species. For each GO term, five SVM classifiers are trained using partitions extracted using homology related information (Lobley et al., 2008). FFpred determines whether a sequence has the GO term using majority voting on the five classifiers for each GO term. Prediction confidence is computed using Platt distance to probability translation. The second iteration of FFpred improves the training protocol used to train FFpred (Minneci et al., 2013). The main improvements proposed include a larger training dataset originating from UniProtKB and using K-Fold cross validation. The third iteration of FFred expanded the prediction capabilities to tackle the three GO ontologies (Cozzetto et al., 2016).

Classification methods combine different techniques in order to increase the data available to the methods. The work proposed by Guan et al. (2008) uses SVM and Bayesian network of SVM's. The architecture utilised includes an SVM classifier and a Bayesian network. The SVM classifier was trained for all GO terms with bagging. The Bayesian network maps the GO hierarchy, each node contains a specifically trained SVM classifier.

SVM separates different classes using a hyperplane. The distance of classes in the feature space is a key factor for the performance of the classifier. This rationale is central in the work proposed by Sokolov and Ben-Hur (2010). The work proposed two methods that modified the ML technique to increase the distance between the classes. In the case of the NN method, the error function factors in the GO hierarchy. Whilst the SVM hyperplane placement algorithm was modified to separate the true and best match by a given margin. The work proposed by Kahanda and Ben-Hur (2017) is the second iteration of the SVM based method. This work was improved in three areas: use of cross validation, the dataset was enriched with PPI data and changed the hyperplane positioning to consider the first and second-best sample as one.

In order to apply NN on protein sequences, the sequence information must be converted to a matrix. For this purpose, the work proposed by Clark and Radivojac (2011) uses NN to perform protein function prediction from a sequence. The process of con-

verting the protein sequence into the matrix required experimentation to find the optimal method encoding method. Dataset containing protein features and annotations were used with NN and SVM for protein function prediction (Jensen et al., 2003).

2.10.4 | Deep Learning

Within the domain of artificial intelligence, Deep Learning (DL) is being applied in different areas. This subsection reports on the application of DL techniques in protein function prediction.

DL techniques are a subclass of ML technique that applies neural networks to a complex problem. In DL neural networks are stacked to create intermediate abstract features that optimise the classifier performance.

Recurrent Neural Network (RNN) is a DL architecture that has a memory of past data. RNN has been applied to perform protein function prediction from a protein sequence (Cao et al., 2017; Liu, 2017). The work proposed by Liu (2017) performs a prediction of four specific GO terms. The architecture uses multiple forward and reverse Long Short-Term Memory (LSTM) layers. The output layer is a fully connected layer that uses "SoftMax". SoftMax enables a classification network to perform prediction together with confidence. The work proposed by Cao et al. (2017) uses a different approach, whereby each protein sequence is split into k-mers and associated with GO terms. K-mers occurring less than 1000 times are discarded. The k-mers and annotations are fed into the RNN, padding the input matrix if needed. The output of the RNN network are the predictions (GO terms).

Autoencoders are a DL technique that enables efficient data coding and decoding. Singular Value Decomposition (SVD) technique enables the decomposition of a given matrix (A) into three components, $(A = U \sum V^T)$. $\sum$ is a diagonal matrix that maps the strength of the relationships. Truncated Singular Value Decomposition (tSVD) performs SVD on the given matrix keeping only the strongest k components (dimensionality reduction). The work proposed by Chicco et al. (2014) uses autoencoders to perform protein function prediction and compares its performance against tSVD methods. Autoencoders are configured to have a small hidden layer so that the network "learns" the most important features in the dataset (to perform dimensionality reduction). The work proposed by Gligorijević et al. (2018) used Multimodal Deep Autoencoder (MDA) and SVM on PPI data to perform protein function prediction. This work uses two PPI networks for yeast and human species based on STRING database. The role of the MDA to map the input data into vectors of numbers (embedding). For a set of proteins, the PPI information is extracted using Random Walk with Restarts and computed into a Posi-

tive Pointwise Mutual Information (PPMI) matrix. The matrices are fed into an MDA to perform the embedding process. The generated matrix is used to train an SVM model which in turn was used to perform function predictions.

Restricted Boltzmann Machines are a DL architecture that has just one hidden unit. Zou et al. (2017) applied Deep Restricted Boltzmann Machines (DRBM) on PPI data to predict missing functionality in current annotations. DRBM were selected due to their resiliency to missing information, convergence speed and stability. This work is based on the GO ontologies and Gene Ontologies Annotations (GOA). The GOA dataset was filtered to keep only four species and GO terms associated with at least one protein.

2.11 | Summary

This chapter provided an introduction of the different areas required for this dissertation and a review of the related work. The overview introduced protein representation and protein structure. This was followed by the processes used to determine protein annotations (functionality description) and explored of protein similarity. A number of protein databases and the mechanism to store protein functionality (GO) were detailed. These were followed by an overview of ML techniques and feature selection. The last part of the overview tackled ML performance metrics and CAFA shared task performance metrics.

The final part of the chapter reviewed the work performed in the area of protein function prediction. The review reported on the work on protein function prediction using four main approaches, namely PPI, homology based, ML and DL. The four different approaches utilised different datasets whilst some methods utilise multiple data sources and multiple techniques.

3

Methodology

This chapter presents a detailed description of the process used to determine the experiments and the experiment environment. The first section discusses the experiment design considerations building on the knowledge acquired in Chapter 2. The second section details the experiment design. The third section details architecture of the experiment setup including software modules required to perform the experiments. The fourth section describes the datasets used and the augmentation process in detail. The last section details the process used to generate species-specific datasets required for ML.

3.1 | Design Considerations

The aim of this dissertation is to improve Protein Function Prediction (PFP) though the use of ML techniques. This work will use the datasets provided by Dr Lees from the Orengo Bioinformatics Group at UCL to investigate the application of different ML techniques.

One of the objectives defined at the outset requires the identification of features in the dataset to perform PFP. Given a dataset with n features, there are $n!$ different combinations. The time required to evaluate one combination is given by t, assuming evaluation of individual permutation is done in constant time. The total time to evaluate all the feature space is thus $t \times n!$. Although this approach guarantees the identification of the feature set linked to the global optimum (highest F_{max} value), the time required is unfeasible.

In ML feature selection is critical as it determines the features required. Different feature combinations have different information levels that enable the ML technique to learn and perform predictions. The identification of the pertinent subset of features can

be performed manually or automatically. In case of manual feature selection, domain knowledge and data analysis are essential to isolate key features.

The expert knowledge of the domain can either select a subset of features to use in ML or discard a set of features *a priori* as their information contribution is insignificant. In the latter case, further feature analysis is required. For this purpose, automatic feature selection is utilised.

Automatic feature selection is implemented as a GA that mimics the process of biological selection. Dataset features are represented by binary bits and a combination of features are called a GA chromosome. The fitness function is used by the automatic feature selection to a determine the performance of the chromosome. The best chromosomes are cross bred and mutated to explore the feature space. The termination criteria used in automatic feature selection are either reaching a predefined fitness level or the number of epochs to be performed.

The application of automatic feature selection using a GA on a dataset will identify the best performing chromosome in a specific run. This chromosome is associated with a maximum, but it can a either a local or a global maximum. The GA does not guarantee that the global maximum can be identified as only a fraction of feature space is explored.

The fitness function used to evaluate the GA chromosome, in turn uses ML techniques to train a model, generate predictions of an unseen dataset and evaluate its performance. Different ML techniques perform differently on the same data. The difference is due to different internal workings, thus the optimal feature set is machine learning technique specific. For this purpose, different machine learning techniques must be evaluated.

The training dataset contains protein features derived from sequences originating from different species. Organisms living in different habitats necessitate different protein functionality for survival. The protein functionality implemented by a specific organism can be different from those another species. For example, the zebra fish necessitates different protein functions when compared to *Escherichia coli* Bacteria. The evolutionary distance between species introduces protein sequence and functional divergence.

A PFP trained using proteins originating from different species can predict functionality that is not valid for the target species. For this purpose, the GO provides a species-specific GO DAG called GO subsets or GO slims[1] that describes functionality of a specific species. Species-specific GO DAG can be utilised to check whether the predicted functionality is valid for the target species.

[1] `http://www.geneontology.org/page/go-subset-guide` (Accessed 2018-12-28)

Another solution is to utilise species specific classifiers, that predict functionality relevant to the target species. It is expected that the performance of this classifier is good on the target species, but its performance will drop as the evolutionary distance increases. This dissertation tackles protein function prediction using species specific classifiers aiming to prioritise prediction performance. Due to data constraints the *Homo sapiens* (Human) and *Escherichia coli* (Bacteria) biological taxonomies will be tackled.

3.2 | Experiment Design

This section details the different experiments that were designed to fulfil the objectives of this dissertation. Experiments were designed to identify the putative features from the dataset and the appropriate ML technique for reliable PFP.

3.2.1 | Application of Genetic Algorithm to the Dataset

The application of the GA requires splitting the training dataset available to 80% for training and 20% for validation datasets. The training dataset will be used to train the classifier whilst the validation dataset will be used to evaluate its performance. The GA will be configured to perform 30 epochs, in each epoch the best ten and the worst six chromosomes are single point cross bred and seven bits are mutated. To increase the feature space exploration, two totally random chromosomes are added to the chromosome pool at each epoch.

3.2.1.1 | Genetic Algorithm Feature Selection on Different Taxonomies

Different species adapt to different habitats and external factors. Consequently, a protein function found in a taxonomy can be absent in another taxonomy. Two taxonomy specific datasets were generated for *Homo sapiens* and *E. coli*. The two species-specific datasets were used with automatic feature selection. This experiment will determine whether features for PFP are species specific.

3.2.1.2 | Genetic Algorithm Feature Selection using Different Machine Learning Methods

Different ML techniques utilise different internal workings that have a bearing on the classifier performance. This experiment aims to use the GA to explore the feature space using different ML techniques. For this purpose, Random Forest (RF) and Support Vector Machine (SVM) will be used to evaluate the feature chromosomes.

3.2.1.3 | Machine Learning Tuning

The performance of ML techniques is effected by hyperparameters that modify the behaviour of the technique affecting its performance. This experiment assesses different hyperparameters of ML techniques.

ML techniques have different internal workings thus each technique requires specific tuning. Identification of the best hyperparameters for the given dataset can be achieved through experimentation. The experimentation entails keeping the experiment setup constant varying only specific hyperparameters, to determine the change in performance and/or runtime.

In this experiment the SVM penalty variable referred to as C will be investigated. This parameter controls the penalty for misclassified samples during training. High C values reduce training error but tend to reduce the distance between the classes. Conversely, low C values increase the distance between classes. In this experiment two penalty values are considered C=1.0 and C=0.1 due to resource constraints.

In the case of RF, the leaf splitting criterion is explored. For this experiment two criteria are considered, entropy that is based on information gain and Gini that is based on misclassification probability.

The best performing set of features for each ML technique with the respective hyperparameters will be tested for generalisation. The test setup entails, training the ML technique with the training dataset and generate predictions on the testing dataset.

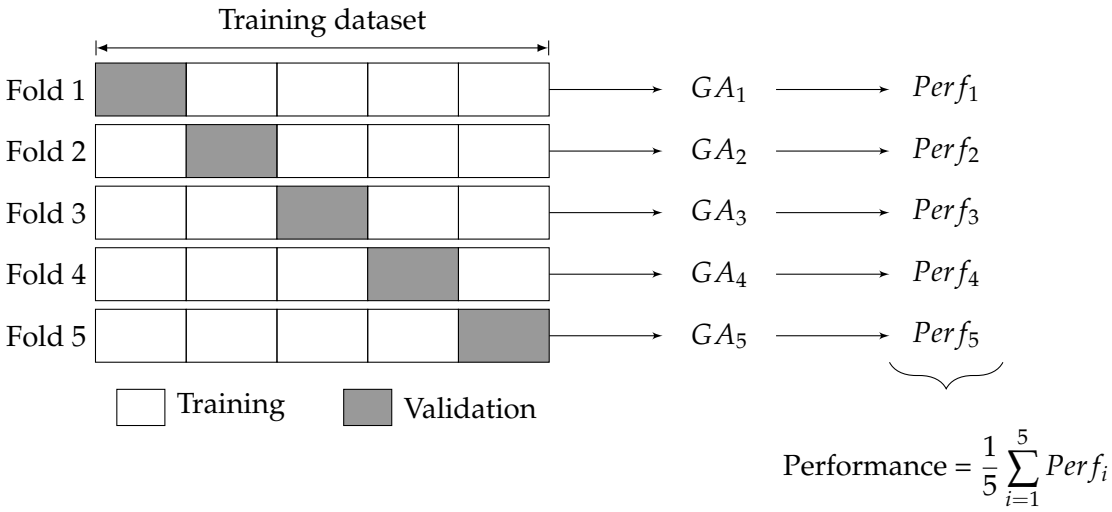


Figure 3.1: Cross validation applied to feature selection process.

3.2.2 | Cross Validation and Feature Selection

The performance of the classifier is dependent on the data used during training. Within the GA dataset variation can generate different results.

The experiment illustrated in Figure 3.1 entails performing k-fold cross validation with K = 5 on the training data. For each fold, the GA will be executed, and the chromosomes ranked. The final result is the mean score of five different GA that have performed feature selection on training and validation data different partitions.

3.2.3 | Application of Neural Networks to the Dataset

During NN training the dataset features are prioritised through the weight adjustments. The downside of using NN is that all the dataset is required. In the hidden layers of the NN abstract features are generated. This experiment will train a number of NN architecture to determine the most appropriate for PFP.

The amount of training data available to train the species-specific models is limited. For example, in the case of *Homo sapiens* which is the most represented species there are just over ten-thousand annotations. In view of these limitations, DL techniques will not be applied and the multi-layer perception networks will be used.

The NN training termination criteria used impacts the network performance and the training time. The decision can be based on the number of epochs or on the change in network error rate. The main aim of the training phase is to minimise the network error. During training, the weights are adjusted to find a better optimum. The equation of the error surface is unknown thus the only way to minimise it is to explore it through training.

The neural network architecture used in this experiment consists of one input layer, different number hidden layers and an output layer. The experiment will use up to three hidden layers with neuron counts based on the distinct number of GO terms in the dataset. The number of neurons start from two-hundred and increases to two-thousand in four steps as described in section 4.3. The hidden layers use the ReLU activation function and the output layer uses SoftMax activation function. The SoftMax output layer, permits the classifier to output both the predicted class and prediction confidence.

Defining the training termination criteria based on the number of training epochs, does not factor in network convergence. For this purpose, the output of the loss function is used to compute the error of the epoch. Training is stopped when three successive epochs do not further reduce the error. Allowing three epochs should enable the NN to escape a local optimum and find a better one.

To speed up training the training epoch is split into mini batches. The mini batch shuffles the training set and run a mini epoch on two-hundred samples. The motivation is that smaller adjustments will help the network converge faster (Mishkin et al., 2016).

During training, the network errors are computed and through back propagation the error is apportioned using the neuron weights. In the weight adjustment phase, the ADAM optimiser considers previous adjustments (momentum) and variable specific learning rates.

The different NN architectures will be tested for generalisation. The test setup entails, training the NN architecture with the training dataset and generate predictions on the testing dataset

3.2.4 | Determine the Amount of Training Data Needed

This experiment aims to investigate the link between the amount of training data and the performance of the ML model. For this experiment the top performing method for *Homo sapiens* and *E. coli* biological taxonomy was tested. In this experiment the protein sequence annotations contained in the training dataset are read to build a number of datasets. The sizes of the dataset are $500, 1,000, 1,500, ..., maxsize$.

For each generated dataset, a ML model is trained and the performance is evaluated against the same unseen testing dataset. This experiment will enable analysis of different aspects of the ML model in relation to the training dataset size.

3.2.5 | Cross Species Model Experiments

A control experiment was performed to determine the performance of a model trained on a specific species when it is applied in the context of another species. To assess the performance of the model on other taxonomies, all the species available in the training dataset will be used. In order to avoid skewing the results, only taxonomies with one hundred sequences or more will be reported. The expected outcome of this experiment is to show the relation of the predictor performance with the taxonomic distance between species.

3.3 | Solution Architecture

The main aim of the dissertation is to build a ML model that can perform reliable protein function prediction. This aim requires the evaluation of the predictive performance of the different features available in the dataset using different ML models.

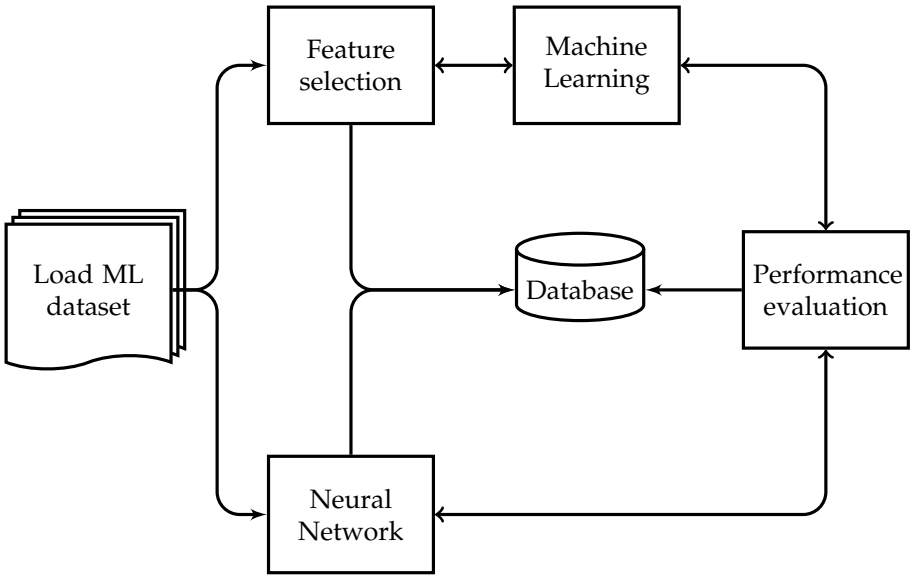


Figure 3.2: Solution Architecture used in this dissertation. Independent software modules are interfaced to provide the required experiment environment.

For this purpose, the architecture defined four modules interacting as illustrated in Figure 3.2 through the fifth module. The four modules are completely decoupled from each other to facilitate development and component reuse. The proposed functionality of the modules is:

- *Pipeline* - Wrapper modes that orchestrates the other modules to run the experiments.
- *Feature Selection* - Implements the automatic feature selection.
- *Machine Learning* - Wrapper module that enable testing of different ML methods.
- *Performance evaluation* - Implements CAFA hierarchical evaluation metrics.
- *Neural Network* - Wrapper module that enables testing of NN.

Apart from the five core modules two additional software projects were developed. These two modules enable the creation of species-specific datasets and processing of

external and generated data.

- *Dataset generators* - Implements the logic to generate the required datasets for ML.
- *Miscellaneous* - Consists utility code to manipulate the various datasets and generate figures.

3.3.1 | Software Components

The experiments defined in the section 3.2 require software components to be defined and developed. This section describes the implementation aspect of each module.

3.3.1.1 | Feature Selection

The feature selection module is an independent module that provides the framework to explore the input dataset feature space. The exploration is performed by generating chromosomes representing dataset features. Each chromosome is evaluated using a fitness function. Through this approach, the GA is agnostic of the method used to evaluate the chromosome. In this work, the fitness function will return two values, the actual performance metric and the prediction confidence. This additional information enables the assessment of the predictor quality at the reporting stage.

In this work the GA was modified to be more aggressive in the exploration of the feature space. The main improvements were:

- Initial seeding of the population includes random chromosomes and individual features chromosomes.
- In each generation random chromosomes are added to the chromosome pool to search the feature space more aggressively.

3.3.1.2 | Machine Learning

The machine learning module provides the necessary infrastructure to pre-process the dataset, train ML models and use an external Python module to compute performance metrics. The module uses the ML models provided by Scikit-learn (Pedregosa et al., 2011).

In order to facilitate the evaluation phase, the ML module has an option to store the prediction results to an SQLite database. This option was added to avoid re-executing the specific experiment to run the evaluation code.

3.3.1.3 | Evaluation

The evaluation module implements the CAFA evaluation framework in Python. The implementation is a clean room implementation based on work of Clark and Radivojac (2013). Apart from the coding aspect, two additional data sources are required. The first one is the GO DAG and the corresponding precomputed information content value for each GO term. Dr Lees from Orengo Group at UCL provided us with both an OBO file containing the GO DAG and a Python pickle file containing the information content of 29,090 GO term originating three ontologies.

The correctness of the implementation was verified against a manually worked-out example, exact matches and the evaluation code used in CAFA2. The CAFA2 evaluation is a Matlab implementation developed by Yuxiang Jiang[2]. The GO DAG is available in OBO file format. This file is a human readable file and provides the information of the GO terms and the GO hierarchy. To verify that the two implementations generate the same result, ground truths, predicted terms, OBO file containing the GO DAG and GO term information content must be the same. A set of ground truths, predicted terms were extracted from Matlab and formatted to be used by the Python implementation. The precomputed information content data was extracted from Matlab and stored in the required format for Python. A number of tests were performed and confirmed that the two implementations are generating the same output given the same inputs.

The results of an evaluation runs are stored into MariaDB database. This facilitates checking the status of the run, using standard SQL syntax to retrieve the required information. Furthermore, the experiment setup involves three machines, each one having specific evaluation data. Through standard database administration tools, the database is exported from the experiment machine and imported onto the development machine. The availability of the data on a single machine simplifies data analysis.

3.3.1.4 | Neural Network

The neural network provides the necessary support infrastructure to load the input dataset, perform training, generate prediction against the testing data and use an external module to compute performance metrics. The module uses the NN models provided by PyTorch (Paszke et al., 2017). The implementation of PyTorch was preferred over the one provided by Scikit-learn as it provided more control over the training termination criteria.

[2] `https://github.com/yuxjiang/CAFA2` (Accessed 2018-12-28)

3.3.1.5 | System Pipeline

The system pipeline is a Python module developed to run the experiments. Through the appropriate interfaces built into the software, different experiments can be executed by orchestrating the API's of the different modules. All the information pertinent to the experiments such as timings and results are stored in MariaDB to facilitate retrieval. Within the pipeline a class was implemented to read the predictions stored in SQLite and trigger the evaluation code.

3.3.1.6 | Dataset Generators

The training and evaluation datasets used in this dissertation were provided by Dr Lees at UCL. These two datasets required augmentation to generate species-specific datasets. Within the two datasets, sequences are identified differently. In the training dataset the identifier used is the UniProt identifier. Whilst in the evaluation dataset, the identifier is the CAFA target identifier. Although the result of the two process is the same, processing the evaluation dataset requires an additional step to map CAFA targets with UniProt identifiers. To simplify the codebase, the training and evaluation datasets are generated using two different Python projects.

3.3.1.7 | Miscellaneous

This work required inclusion of additional data sources and the creation of intermediate artefacts. The utility code was developed to check the feasibility of an idea and to assess whether it addresses specific needs. The code includes: GO DAG analysis, tool to download annotations for protein from QuickGO and UniProt web services, CATH Funfam annotation downloader, GOA QuickGO dataset processing and GA result analysis tool.

An important python implementation within the miscellaneous package is the figure generator. This source file is used to generate the result visualisations presented in this dissertation. Each visualisation has a data source declaration that defines which MariaDB database to use and the primary key ranges for the experiment concerned. The required data is retrieved from the MariaDB database using the information of the data source definition and the visualisation generated using Matplotlib (Hunter, 2007). This approach ensured that figures are consistent throughout and re-execution of a specific experiment requires updates of the data sources definition, database name and primary key ranges.

Python package	Version	Usage
Cython	0.29	Used to convert Python code to C.
SQLAlchemy	1.2.14	Used to integrate MariaDB and Pandas.
bioservices	1.5.2	To access QuickGO and UniProt API as Python API.
bitstring	3.1.5	Utilised in the GA to manipulate bit patterns.
goatools	0.8.9	Used to parse and query GO OBO files.
joblib	0.13	The job library used in the GA to parallelise tasks.
matplotlib	3.0.2	Utilised to generate high quality visualisations.
mysql-connector	2.1.6	Used to connect to MariaDB.
neo4j-driver	1.7.1	Used to analyse the GO DAG.
numpy	1.15.4	Used to speed up the evaluation code calculations.
pandas	0.23.4	Used to read and manipulate datasets.
psutil	5.4.8	Used to get Linux process ID.
requests	2.20.1	Used to query QuickGo web services.
scikit-learn	0.20.0	The main ML library used in this dissertation.
sklearn-pandas	1.7.0	To facilitate usage of Pandas with SciKit-Learn.
tables	3.4.4	Pandas pre-requisite to read HDF files.
torch	0.4.0	Used as the NN implementation in this dissertation.

Table 3.1: Python packages used in this dissertation

Environment	RAM	CPU Type	Total Cores
Development	4GB	i7-6500U CPU @ 2.50GHz	4
Experiment	96GB	Intel(R) Xeon(R) CPU E5620 @ 2.40GHz	16
Experiment	96GB	Intel(R) Xeon(R) CPU E5620 @ 2.40GHz	16
Experiment	128GB	Intel(R) Xeon(R) CPU E5-2660 v4 @ 2.00GHz	56

Table 3.2: Machines used in dissertation experiments.

Type	Version
Operating system	Ubuntu 16.04 LTS
Programming Language	Python 3.6
Database Back end	MariaDB 10.3
Graph Database	Neo4J Server 3.4.7

Table 3.3: Software used in this dissertation.

3.3.2 | System Architecture

The software for this dissertation was developed using the software environment listed in Table 3.3. All the software developed in this work is based on freely available open source software.

The software for this dissertation was written in Python version 3.6. The main driver for this choice is that Python has the necessary modules to connect to the various data sources and tools to process bioinformatics data. Python version 3.6 was compiled from source and installed on the target machines. A python virtual environment was created and installed with the packages listed in Table 3.1 using python install manager (pip).

For this dissertation, four machines were utilised; a development machine and three machines to run the experiments. The development machine is a small virtual machine running on a laptop that was used to develop, test and debug all the software components. To ensure a consistent software environment for all software developed, all software components and experiment configuration files are appropriately versioned in Git. The development environment did not have sufficient resources to run ML against the complete dataset due to memory constraints. This constraint was addressed by using bigger machines to run the experiments as per Table 3.2.

Whilst executing the GA experiments, two observations were made, the memory usage during the RF runs sometimes peaks to around 13GB and that SVM uses a single CPU for a significant amount of time with low memory consumption. For this purpose, the GA execution parameters were modified to control the parallel section illustrated in Figure 3.3. The parallelisation was configured to run five concurrent job RF ML technique and run twelve concurrent jobs SVM ML technique. The cross validation experiments were executed on the machine with higher core count. In order to benefit from the additional resources, the parallelisation parameters were revised to run six concurrent jobs for RF and fifty-three concurrent jobs for SVM.

During debugging of the evaluation module described in section 3.3.1.3, it was noted

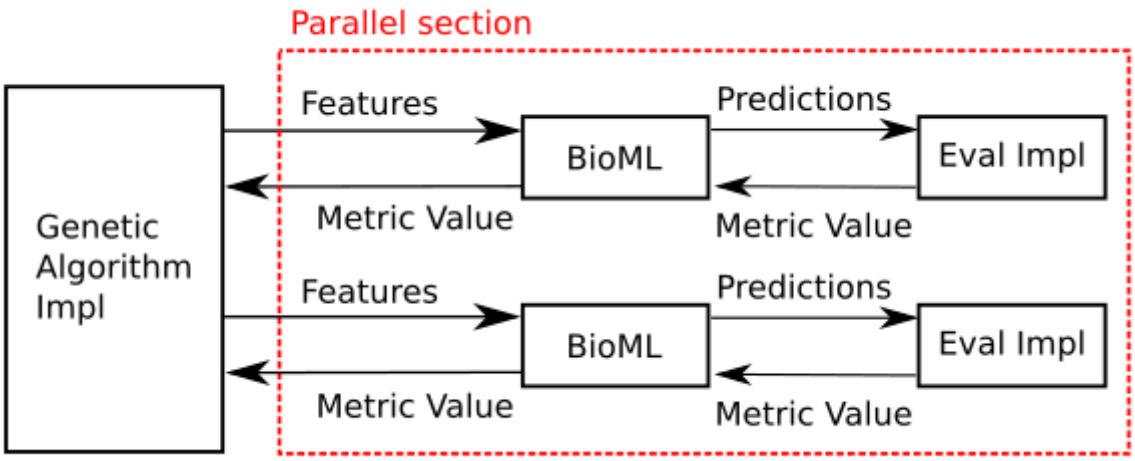


Figure 3.3: Automatic feature selection with the parallel section evidenced.

that the runtime performance of some functions within the developed modules was not optimal. In these cases, the code was rewritten, changing data structures to improve performance. To further improve performance of computationally intensive sections, the Python code is converted to C code using Cython. This transition improved the speed considerably without significant code modifications.

3.4 | The Dataset

This research utilises two datasets that were made available by Dr Lees. The first dataset comprises of the features computed for the proteins in the CAFA2 training dataset. The second dataset contains the features computed for the CAFA3 targets. The first dataset will be used to train the ML algorithm and the second dataset will be used to evaluate the model.

The training dataset was made available as a single Python pickle file of 1.5GB in size. The Python pickle file was de-serialised and contained a Pandas DataFrame (McKinney, 2010). The Pandas dataframe was analysed and it was confirmed that all the GO terms in the file originated from the Molecular Function GO ontology.

Within the Big Data contexts, the Hierarchical Data Format (HDF)[3] file format is used for organising large datasets. The HDF file format is supported by different languages thus enabling different applications to exchange data.

[3] `https://www.hdfgroup.org/HDF5` (Accessed 2018-12-28)

The second dataset comprised of 166 HDF files with a total size of 50GB. The naming convention of the files, denote the GO ontology of the terms in the files. The files for Cellular Component Ontology start with "C", files for Molecular Function Ontology start with "M" and files for Biological Process Ontology start with "P". This dissertation focuses on molecular function ontology, for this purpose, the data of the Molecular Function ontology was extracted.

The Molecular Function dataset was distributed amongst multiple files. To facilitate processing the data was read and aggregated into a single HDF file and loaded into the MariaDB database. The steps up to concatenate the HDF files were:

1. For each Molecular Function ontology HDF file (filename starts with "F").
   a) Read the HDF file in Pandas DataFrame.
   b) Removed extra columns that are not useful for this dissertation.
   c) Add the Pandas DataFrame to a list.
2. Concatenate the list of Pandas DataFrames into one DataFrame.
3. Save the Pandas DataFrame into HDF file and in MariaDB.

The datasets consist of twenty-nine features that describe the protein and annotations. An overview of the data available in the dataset is included in Appendix B. The features available in the dataset are:

- *GO* - The target GO term of the entry.
- *taxonomic description fields* - a marker of the protein origin Eukaryote, Bilateria and Gramme Negative Bacteria.
- *disorder* - a marker to determine lack of stable tertiary structure computed using RAPID[4].
- *disorder information* - for the given sequence such as fraction of charged residues, kappa value and hydropathy are computed using CIDER[5].
- *GO term measurements* - Measure computed on the GO term itself such as occurrence frequency, level in GO DAG and whether GO term is linked to a domain.
- *number of PFAM domains* - number of PFAM domains associated with this GO term.

[4] `http://biomine.cs.vcu.edu/servers/RAPID` (Accessed 2018-12-28)
[5] `http://pappulab.github.io/localCIDER` (Accessed 2018-12-28)

- *mmseq information* - mmseq information extracted from sequence, software available from GitHub MMseqs2[6]. The information includes closest hit information, such as bit score, sequence identity and rank of sequence hit from all hits.
- *PFAM to GO mapping* - whether there is a PFAM to GO mapping using pfam2go list[7].
- *signalp* - where sequence has presence of signal peptide cleavage sites in amino acid sequences from different organisms using the SignalP server[8].
- *tmhmm* - the number of predicted transmembrane helices in proteins using the TMHMM server[9].
- *all_go_prop_cath_funfam* - The proportion of proteins that are members of the same CATH funfams and have this GO term, considering all annotation evidences.
- *all_go_prop_dc* - The proportion of proteins that have the same domain combination and have this GO term, considering all annotation evidences.
- *all_go_prop_pfam* - The proportion of proteins that are members of the same PFAM Family and have this GO term, considering all annotation evidences.
- *all_go_prop_pfam_funfam* - The proportion of proteins that are members of the same PFAM domain and have this GO term, considering all annotation evidences.
- *strict variants of variables prefixed with all_go_prop_* - the ratio is computed the same but annotation filtering criteria is more strict, discarding electronic annotations.
- *yval* - whether the data used by Dr Lees represents a true positive (1) or not (-1).

The training dataset was loaded into MariaDB for analysis. The dataset contains a total of 3,293,302 annotations originating from 31,097 sequences. GO term occurrence was analysed and the most common ten functionalities are reported in Table 3.4.

3.5 | Cleaning and Augmenting the Data

After the two datasets were loaded and analysed, it transpired that both datasets contain features computed for a number of GO terms including for the true annotations. The

[6] `https://github.com/soedinglab/MMseqs2` (Accessed 2018-12-28)
[7] `http://geneontology.org/external2go/pfam2go` (Accessed 2018-12-28)
[8] `http://www.cbs.dtu.dk/services/SignalP` (Accessed 2018-12-28)
[9] `http://www.cbs.dtu.dk/services/TMHMM` (Accessed 2018-12-28)

Occurrences	GO Term	Description
28,343	GO:0005488	binding
24,150	GO:0003824	catalytic activity
23,353	GO:0005515	protein binding
21,930	GO:0097159	organic cyclic compound binding
21,612	GO:1901363	heterocyclic compound binding
21,195	GO:0043167	ion binding
17,864	GO:0003676	nucleic acid binding
17,509	GO:0043169	cation binding
17,030	GO:0016787	hydrolase activity
16,954	GO:0043168	anion binding

Table 3.4: Top 10 most occurring GO terms in the training dataset.

augmentation process uses external datasets such as GOA QuickGO to identify the true positive GO terms for protein sequences. Due to data differences in the way proteins are identified, the training and evaluation datasets are augmented differently.

The augmentation process of the training dataset is outlined in Figure 3.4. The Pandas dataframe containing the dataset is loaded from the Python pickle file. The GOA dataset version 124 was downloaded from GOA FTP site[10]. The release date of the version 124 matches the release date of the dataset. The downloaded GOA dataset is a compressed tab delimited file, containing UniProt ID, GO term, evidence code and source database amongst other details. This file was parsed using a specifically developed Python utility to read the file and extract the required fields. For efficiency, the set of proteins in the training dataset was used to determine which proteins to process in the GOA dataset. Subsequently, filters were applied related to the molecular function ontology and evidence codes. In order to train the ML models with reliable data, only laboratory confirmed evidence were used. The annotation information extracted from the GOA dataset was stored in the MariaDB database.

The enrichment term set for a GO term includes all the GO terms up to the GO ontology root. The deeper the term is on the GO DAG the bigger the enrichment term set. The retrieved GOA annotations were analysed, and it was noted that some sequences had enrichment terms to the ontology root. For this purpose, the annotation subgraph was analysed and only the leaf nodes were kept. This step was important to ensure

[10] `ftp://ftp.ebi.ac.uk/pub/databases/GO/goa/old/UNIPROT/` (Accessed 28 Dec. 2018)

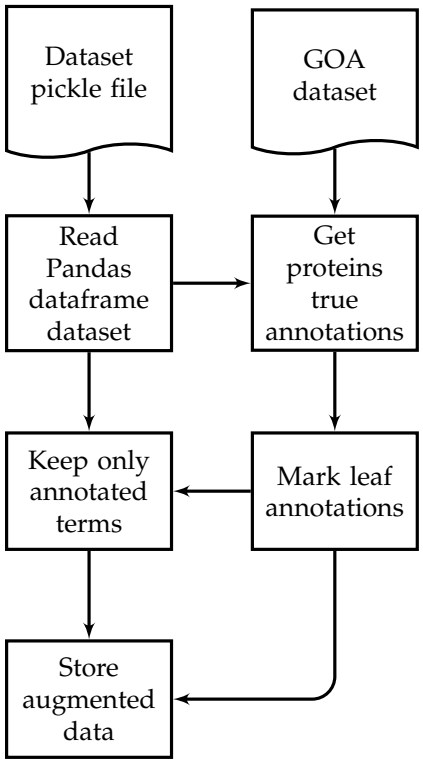


Figure 3.4: Training dataset augmentation though multiple data sources.

that the classifier trained on the dataset would output specific terms rather than generic ones. The process used to identify the leaf nodes is as follows:

1. For each protein sequence in the dataset
    a) For each GO annotation, get the enrichment term set from GO DAG.
        i. For each GO annotation, mark node as leaf node if it does not occur in the enrichment term set of other GO terms.
        ii. Flag leaf nodes in database.

The GOA dataset containing the most specific annotations of the proteins are used to build the training dataset. The intersection of the Dr Lees' dataset and the GOA dataset on the protein identifier and the GO term fields generates the augmented dataset. This dataset contains all the annotated proteins and the respective ML features.

The process to augment the evaluation data requires more steps to map the CAFA target name to UniProtID and to retrieve UniProtID taxonomy identifiers. The process is illustrated in Figure 3.5.

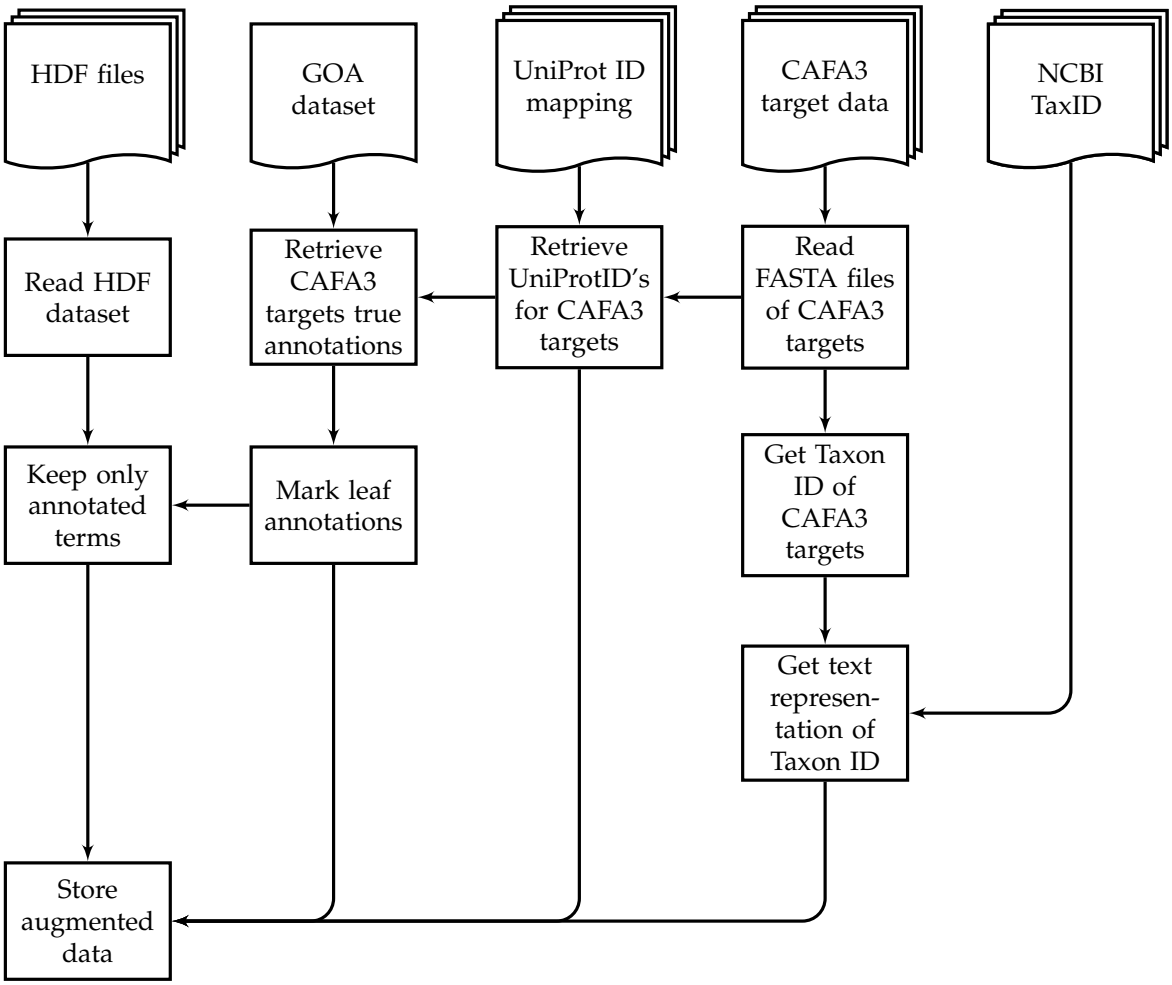


Figure 3.5: Evaluation dataset augmentation though multiple data sources.

Within the HDF file, sequences are labelled using CAFA target identifiers. In order to search for the CAFA target details in protein databases, the UniProt accession identifier is required. CAFA provides the target sequences and sequences identification information in the FASTA file for each taxonomy, marked as "CAFAtarget data" in Figure 3.5. The sequences identification information consists of CAFA target identifier and UniProt entry name. The FASTA files, one for each taxonomy were extracted to a directory. Each file was parsed, extracting the CAFA target ID and UniProt entry name. All the information was inserted in the MariaDB database.

The data from the FASTA files provided the required link to map the CAFA targets to UniProt entry name. A list of UniProt entry name was extracted from the database and

used in UniProt ID mapping service[11], marked as "UniProt ID mapping" in Figure 3.5. The mapping service, generates a tab delimited file which was downloaded, parsed and loaded in MariaDB database.

Within the "CAFA3 dataset" the taxon identifier is a numeric value representing the taxonomy identifier (taxonID). This identifier must be converted to species names using "NCBI TaxId". For this purpose, the list of taxonomic identifiers obtained from the CAFA FASTA files were mapped to species using the NCBI Taxonomy mapping service[12]. The information pertinent to the taxonomy identifier and species description was stored in MariaDB database.

The GOA dataset contains the annotations data used by the QuickGO website. The true annotations of the CAFA3 targets can be retrieved from the "GOA dataset". For the scope of this dissertation, the GO annotations version 152 were downloaded from GOA FTP site[13] and loaded to MariaDB. The QuickGO annotations were retrieved for all CAFA targets using the UniProt identifier. The retrieved annotations were filtered to keep only laboratory curated quality annotations originating from the Molecular Function ontology. The ground truths for the CAFA3 dataset were processed to remove leaf annotations using the same procedure for the HDF file.

The QuickGO annotations were found to be enriched to the ontology root. For this purpose, the leaf annotations were identified using same approach as for the training dataset. The intersection between of the QuickGO database and that of the Dr Lees CAFA3 dataset evidenced as "Keep only annotated term" generated the evaluation dataset used in this research.

3.6 | Generating Machine Learning Datasets

The augmented dataset contains protein measurements of proteins originating from different species. For the scope defined in the earlier chapters, species-specific datasets must be generated.

3.6.1 | Manual Feature Selection

In order to reduce the size of the problem, the dataset was analysed to determine whether some features should be removed from the ML datasets. The reduction of a single fea-

[11] `https://www.uniprot.org/uploadlists` (Accessed 2018-12-28)

[12] `https://www.ncbi.nlm.nih.gov/Taxonomy/TaxIdentifier/tax_identifier.cgi` (Accessed 2018-12-28)

[13] `ftp://ftp.ebi.ac.uk/pub/databases/GO/goa/old/UNIPROT/` (Accessed 2018-12-28)

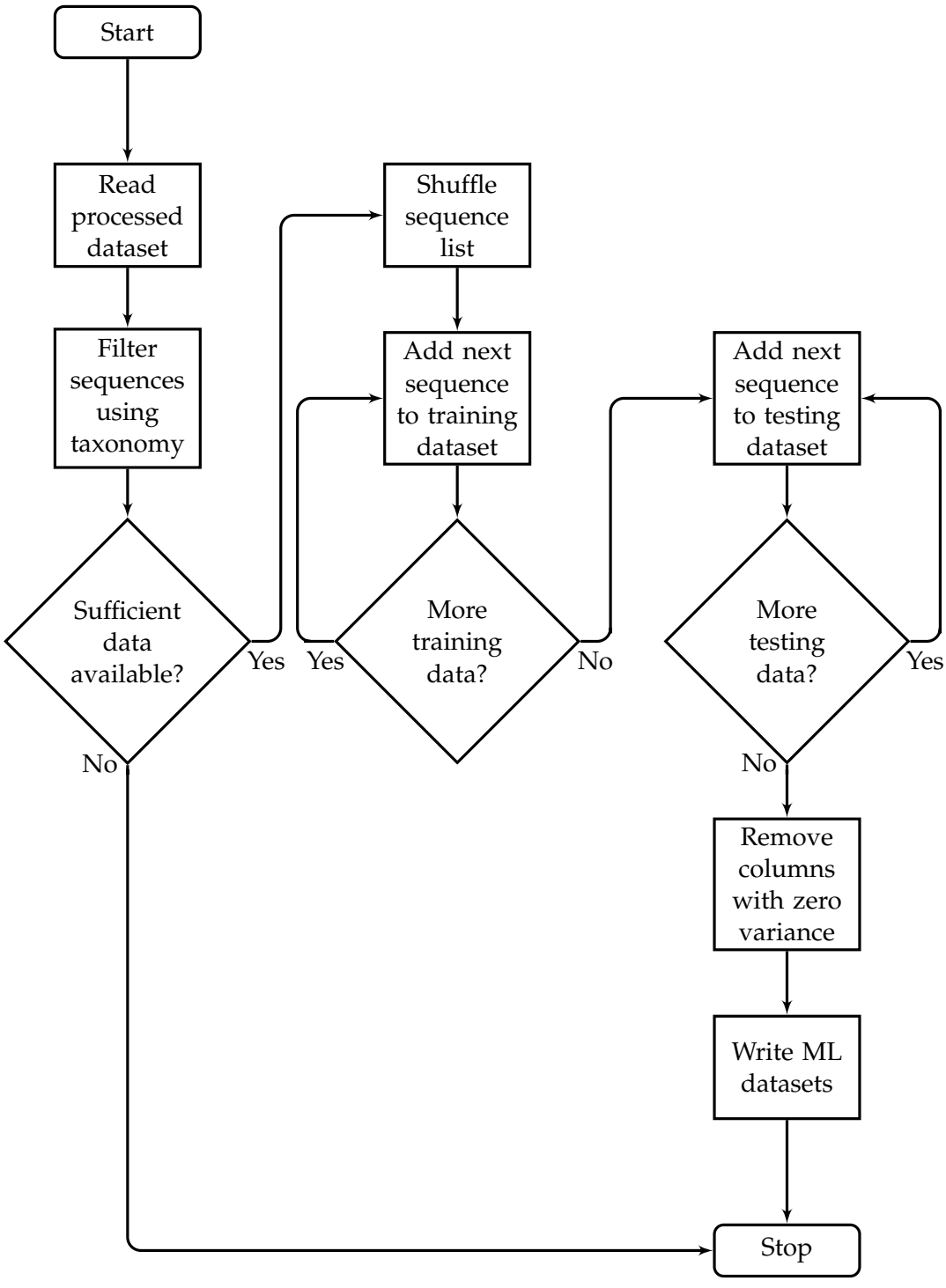


Figure 3.6: ML dataset generation from augmented dataset.

ture reduces the number of combinations from $n!$ to $(n-1)!$. With a large value of n the reduction of combinations is considerable. From the dataset, five features, were removed, namely:

- *yval* - flag to determines whether entry is considered part of true positive set by Dr Lees model. This feature is irrelevant for protein function prediction.
- *go_freq_score*, *go_level*, *go_with_domain*, *go_with_mmseq* - These features describe different dimensions of the GO terms. Given that the model is predicting GO terms. These metrics create a strong correlation between prediction features and ground truths, thus they were discarded.

3.6.2 | Dataset Generation

The augmented dataset was used to generate species specific datasets for the defined experiment. The process used to generate the species-specific dataset is shown in Figure 3.6.

The first step of the process is to read the complete dataset and apply a taxonomic filter. A check is performed to determine whether the sufficient annotations are available to fulfil requested amounts. In case sufficient data is available, a list of sequence identifiers is extracted and shuffled. A running count of the annotations is computed. The sequences are added to a list till the required annotation count is reached. The process is repeated for the testing dataset. After selecting the training and testing sequences the dataset entries of the selected sequences are stored to the disk in comma delimited format.

3.7 | Summary

This chapter provided the detail of the setups and experiments conducted in this work. In the definitions of the environment, the different components are defined in terms of architecture and testing perform to verify the correctness of the implementation. The next chapter reports and discusses experiment execution and results obtained.

4

Results and Discussion

Chapter 3 defined the experiments and the required software architecture. The proposed architecture was implemented, and experiments executed. This chapter reports and discusses the experiment outcomes.

4.1 | Application of Genetic Algorithm to the Dataset

Automatic feature selection was executed on two species namely *Homo sapiens* and *E. coli*. For each species, two ML techniques were utilised to evaluate GA chromosomes. The following are the techniques and hyperparameters utilised:

- RF with Gini as the node splitting criterion, denoted as RF-Gini.
- RF with Entropy as the node splitting criterion, denoted as RF-Entropy.
- SVM with the misclassification penalty value set to 1.0, denoted as SVM-C1.0.
- SVM with the misclassification penalty value set to 0.1, denoted as SVM-C0.1.

The GA execution is a resource intensive process that takes a considerable amount of time. Work was performed to speed up the GA chromosome evaluation through code optimisation and parallelisation. Parallel chromosome evaluation required resource management to avoid CPU starvation or memory swapping. Different ML techniques have specific resource requirements. RF requires a large memory footprint to build the trees and has inbuilt support for parallelisation. The SVM technique is CPU intensive and does not support parallelisation. In order to cater for the different parallelisation requirements, the degree of parallelisation used is governed by the configuration parameter of the GA. Parallelisation of chromosomes evaluation reduced the overall execution

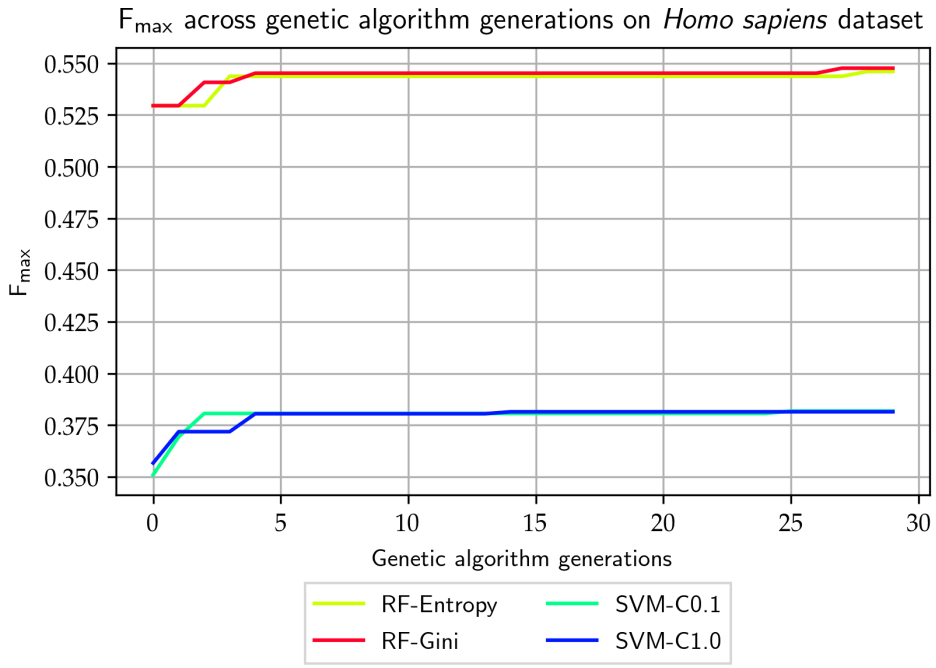


Figure 4.1: F_{max} across feature selection training epochs on *Homo sapiens* dataset.

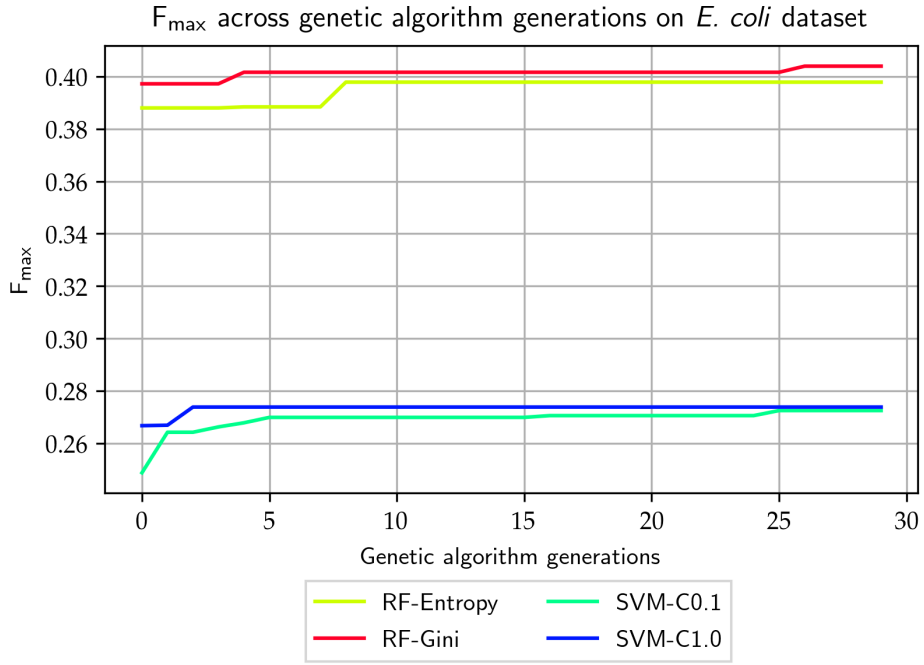


Figure 4.2: F_{max} across feature selection training epochs on *E. coli* dataset.

ML Technique	*E. coli*		*Homo sapiens*	
	Run time (h)	Total time (d)	Run time (h)	Total time (d)
RF-Gini	1.590	0.330	6.880	1.410
RF-Entropy	2.210	0.450	11.200	2.290
SVM-C1.0	10.810	5.210	92.090	43.850
SVM-C0.1	10.350	4.990	87.630	41.860

Table 4.1: Genetic Algorithm run time details. The user time in hours, is the time elapsed for the experiment process to finish. The total time in days, accounts for all the CPU resources utilised on the experiment machine.

time. Table 4.1 reports the experiment timings, specifying the user process time and the total CPU runtime. The SVM ML technique is a CPU intensive and takes a considerable amount of time to train and generate predictions. All the runs for this experiment were executed on the same type of machine to enable duration comparison.

Each GA chromosome identifies a set of features. The features are evaluated by training a classifier on the training dataset using the selected features. The trained classifier is used to generate predictions for the validation dataset. The generated predictions are evaluated using CAFA metrics, that considers the hierarchical structure of the GO DAG and computes an F_{max} value. Figure 4.1 illustrates the F_{max} value of the different ML techniques across the training epochs of *Homo sapiens*. The outcome of the experiments performed on *E. coli* are illustrated in Figure 4.2.

The RF classifier had superior performance when compared to SVM both in terms of performance as illustrated in Figure 4.1 and also in the required run time as per Table 4.1. Two RF runs were reported, one using Gini as node splitting criterion and the other using Entropy. The Gini run improved the F_{max} by 0.010 and reduced the runtime by four hours nineteen minutes. The two SVM runs with different penalty values (C) had similar F_{max} and run times. The work to speed up chromosome evaluation through parallelisation reduced the run time considerably.

The features selected by the GA for *Homo sapiens* depend on the ML technique utilised in the fitness function. The top 6 GA chromosomes for the RF-Gini classifier are reported in Table 4.2. The top 6 chromosomes for RF with Entropy as splitting criterion are reported in Appendix D Table D.2. The chromosomes for SVM with penalty values of 1.0 are reported in Appendix D Table D.6, whilst chromosomes for the experiment with SVM penalty value of 0.1 are reported in Appendix D Table D.4.

The top-ranking *Homo sapiens* chromosomes, have five features in common, orig-

Chromosome	all_go_prop_cath_funfam	all_go_prop_dc	all_go_prop_pfam	all_go_prop_pfam_funfam	cider_FCR	cider_hydropathy	cider_kappa	disorder	mmseq_bit_score	mmseq_ident	mmseq_prox	number_pfam_domains	pfam_to_go	signalp	strict_go_prop_cath_funfam	strict_go_prop_dc	strict_go_prop_pfam	strict_go_prop_pfam_funfam	tmhmm
1	✓	✓	✓					✓			✓	✓	✓			✓	✓	✓	✓
2		✓	✓			✓					✓	✓	✓			✓	✓	✓	✓
3		✓	✓	✓	✓			✓				✓	✓			✓	✓		
4		✓	✓	✓		✓					✓	✓	✓		✓	✓	✓	✓	✓
5	✓	✓	✓	✓				✓		✓			✓			✓	✓	✓	✓
6	✓	✓	✓				✓					✓	✓	✓	✓	✓	✓	✓	✓

Table 4.2: Top 6 GA chromosomes for *Homo sapiens* taxon using RF-Gini as the ML technique.

Chromosome	all_go_prop_cath_funfam	all_go_prop_dc	all_go_prop_pfam	all_go_prop_pfam_funfam	cider_FCR	cider_hydropathy	cider_kappa	disorder	mmseq_bit_score	mmseq_ident	mmseq_prox	number_pfam_domains	pfam_to_go	signalp	strict_go_prop_cath_funfam	strict_go_prop_dc	strict_go_prop_pfam	strict_go_prop_pfam_funfam	tmhmm
1		✓	✓	✓	✓				✓			✓			✓	✓	✓	✓	
2	✓	✓	✓	✓	✓	✓	✓		✓		✓	✓	✓		✓	✓			
3	✓	✓		✓		✓					✓	✓		✓	✓	✓			
4		✓	✓						✓			✓			✓	✓		✓	
5		✓	✓	✓	✓	✓		✓			✓	✓		✓	✓	✓		✓	
6	✓	✓		✓		✓	✓				✓	✓	✓		✓	✓	✓		

Table 4.3: Top 6 GA chromosomes for *E. coli* taxon using RF-Gini as the ML technique.

inating from ratios computed on protein family membership in protein classification databases (PFAM and CATH). The information from protein databases, originated from four ratios: *all_go_prop_pfam* (proportion of members of the same PFAM family with the GO term) and *all_go_prop_dc* (proportion of proteins having the same protein/domain with the GO term) and their *strict* variants. The fifth feature *pfam_to_go* is the flag that indicates whether a PFAM to GO mapping exists.

The analysis of the top chromosomes of the RF-Gini method, shows a bias towards the proportions computed on protein databases. The top 6 chromosomes include features computed from protein databases and features computed from the protein sequence. The feature *number_pfam_domains* (is the number of PFAM domains associated with this GO term) that is computed from protein databases was selected in five times of the top 6 chromosomes. Two features computed from protein sequences: *tmhmm* (number of predicted transmembrane helices) and protein disorder information prefixed by *cider* were selected five times.

Figure 4.2 reports the performance of the GA runs on the *E. coli* dataset. The reported F_{max} values show that RF was superior to SVM with an F_{max} margin of 0.110. The RF with Gini as node splitting criterion was the best performing ML model with an F_{max} of 0.404. The RF entropy trailed behind with an F_{max} of 0.398 and an increase of 20 minutes in run time. The SVM runs obtained an F_{max} of 0.274 and required eight hours of run time. The different SVM penalty value did produce a negligible change in F_{max}.

The top 6 GA *E. coli* chromosomes for the RF classifier using Gini as splitting criterion are reported in Table 4.3. The top 6 chromosomes for RF using Entropy as splitting criterion is reported in Appendix D, Tables D.1, for SVM using 1.0 as penalty value is reported Appendix D Table D.5, whilst SVM with penalty value of 0.1 is reported in Appendix D Table D.3.

Amongst the top 6 *E. coli* chromosomes, four features were constantly selected by the GA. This set of features includes three features based on protein structural databases and a flag. The three ratios are: *all_go_prop_dc* (ratio of proteins having the same domain/protein combinations), *strict_go_prop_dc* (considering only laboratory annotations) and *strict_go_prop_pfam* (variant of *all_go_prop_pfam* considering only laboratory annotations). The inclusion of *strict* ratios highlights that laboratory curated annotations are include adequate information for PFP. The mapping flag *pfam_to_go* indicated whether a mapping exists between PFAM to GO.

Further analysis of the top 6 *E. coli* GA chromosomes includes both additional ratios based on the protein structural databases and also measurements computed on the protein sequence. The top 6 chromosomes include more *all* ratios with respect to *strict*, highlighting that *all* ratios provide more information for PFP. From the structural

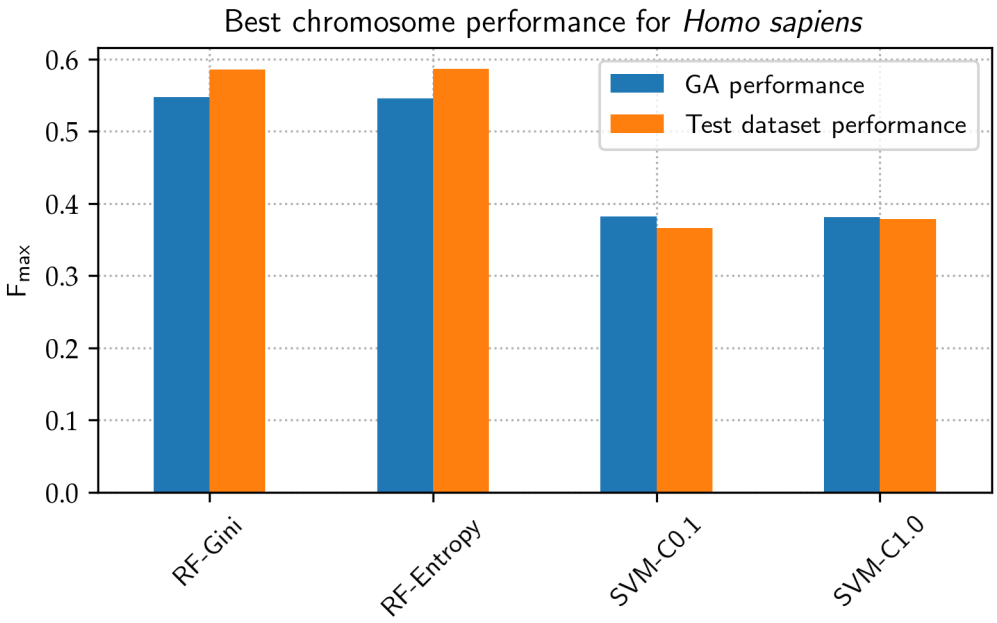


Figure 4.3: Performance of best chromosome on GA dataset and on testing dataset for *Homo sapiens*.

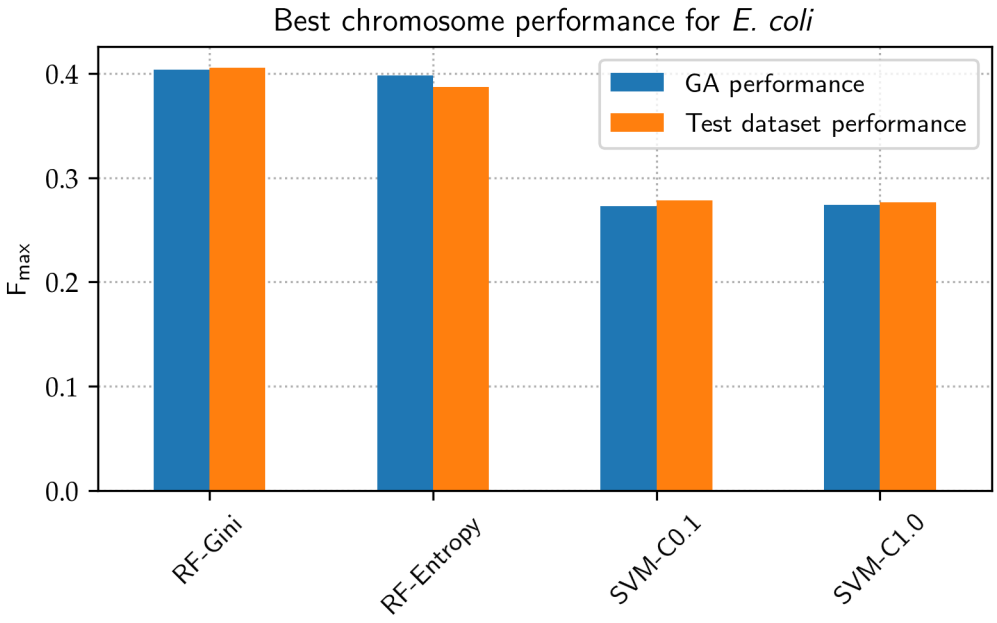


Figure 4.4: Performance of best chromosome on GA dataset and on testing dataset for *E. coli*.

features available, the protein disorder information prefixed by *cider* and *tmhmm* were included.

The four common dataset features selected by the GA runs on the two species are: *all_go_prop_dc*, *pfam_to_go*, *strict_go_prop_dc* and *strict_go_prop_pfam*. The commonality of the features indicates that the protein databases are effective in capturing functional and structural information within protein families.

The "best" GA chromosome of each ML technique was tested against the test dataset. Each ML technique was trained using the complete training dataset and predictions were generated for the testing dataset which were evaluated using CAFA metrics. In order to assess the generalisation of the ML models, the performance of the model is reported both during GA run and also on the test dataset.

The F_{max} of the *Homo sapiens* species on both validation dataset during the GA execution and on the test dataset is reported in Figure 4.3. All four ML methods generalise well, as the F_{max} obtained during the GA execution almost identical performance on the test dataset. Both the validation and test dataset were unseen by the classifier. In case of RF, the additional training data improved the F_{max} of the classifier. The additional training data was not beneficial for SVM as the classifier performance decreased slightly (0.020).

The performance of the classifiers for *E. coli* is reported in Figure 4.4. The generalisation of the ML models is good, as the performance on the GA validation dataset is similar to that of the test dataset. The performance of RF-Gini and both SVM methods improved marginally on the testing dataset. However, in case of RF-Entropy the F_{max} value decreased by 0.011.

ML Technique	*E. coli*		*Homo sapiens*	
	Run time (d)	Total time (d)	Run time (d)	Total time (d)
RF-Gini	0.200	1.200	0.750	4.400
RF-Entropy	0.250	1.470	0.900	5.300
SVM-C0.1	0.410	19.080	4.430	206.970
SVM-C1.0	0.420	19.480	4.660	220.020

Table 4.4: Experiment duration for genetic algorithm applied within cross validation.

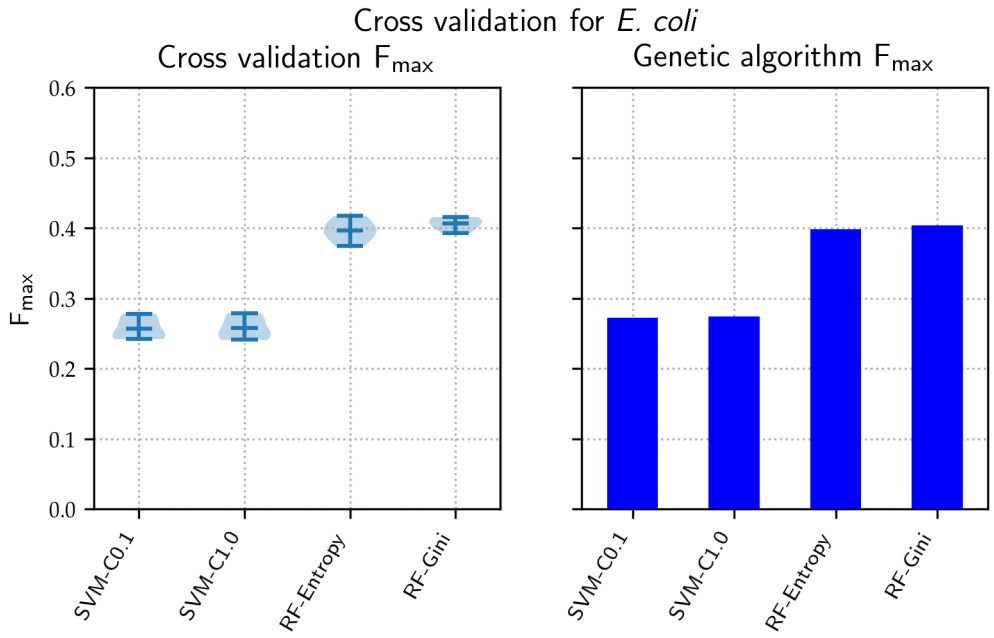


Figure 4.5: F_{max} of GA applied within cross validation folds and on the pre-split training dataset for *E. coli*.

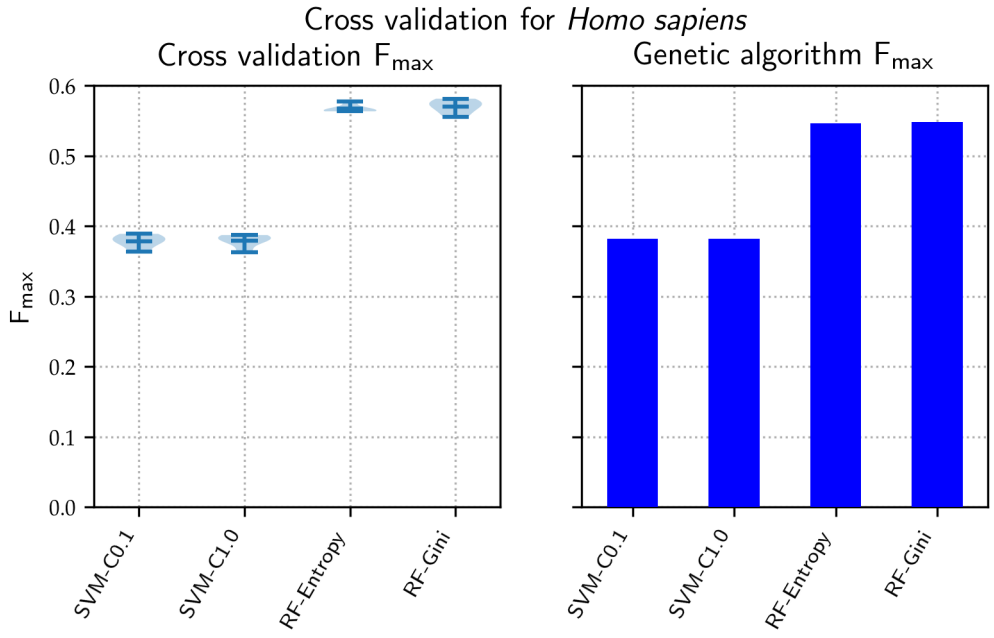


Figure 4.6: F_{max} of GA applied within cross validation folds and on the pre-split training dataset for *Homo sapiens*.

4.2 | Cross Validation of Feature Selection

This experiment internalises the GA within the cross validation folds to determine the mean performance of the GA for a given ML technique. The training and validation datasets were generated from the training dataset using k-fold cross validation with $k = 5$, with 80% of the data used for training and 20% for validation. For each value of k, the respective training and validation datasets were generated, feature selection was executed, and the fold F_{max} is computed. The final F_{max} value is the mean F_{max} across all folds. Table 4.4 reports the time required to execute this experiment.

This experiment executed the GA within the cross validation loop for *E. coli* and *Homo sapiens*. For each taxon, the following ML techniques were utilised:

- RF with Gini as the node splitting criterion, denoted as RF-Gini.
- RF with Entropy as the node splitting criterion, denoted as RF-Entropy.
- SVM with the penalty values set to 1.0, denoted as SVM-C1.0.
- SVM with the penalty values set to 0.1, denoted as SVM-C0.1.

The cross validation results of each taxon are illustrated in Figures 4.5 and 4.6. The left part of the figure shows a violin plot of the F_{max} at the end of the GA runs in each fold. The violin plot shows the minimum, mean and maximum value of F_{max}. The thickness dimension of the violin plot shows the data distribution. The plot on the right of the figure is a bar chart with the performance data of the GA runs using training and validation dataset.

Figure 4.5 illustrates the performance of the cross validation experiment and the GA F_{max} of *E. coli* species. For the four ML techniques used, the F_{max} value obtained during the GA run was within the range of that obtained by the cross validation runs. This illustrates the change in the dataset introduces a change in F_{max} ranging from 0.020 to 0.040.

The cross validation and GA F_{max} performance of the *Homo sapiens* is reported in Figure 4.5. The F_{max} of the GA using SVM to evaluate GA chromosomes was within the range of the cross validation runs. The RF based methods were more susceptible to variation of the dataset as the GA had a lower F_{max} when compared to the cross validation runs. The two methods RF-Entropy and RF-Gini had a F_{max} lower by 0.020 and 0.010 respectively. The GA chromosomes and the cross validation GA chromosomes for the two RF runs were analysed. The chromosomes of the runs show that the change in the dataset shifted the GA towards different chromosomes.

While analysing Figures 4.5 and 4.6, it stands out that the cross validation in *E. coli* had a greater variance when compared to *Homo sapiens*. The main reason attributed to this variance is size of the training dataset. Whilst the in case of *Homo sapiens* 7,677 training samples are used, the *E. coli* taxon had 2,120 training samples.

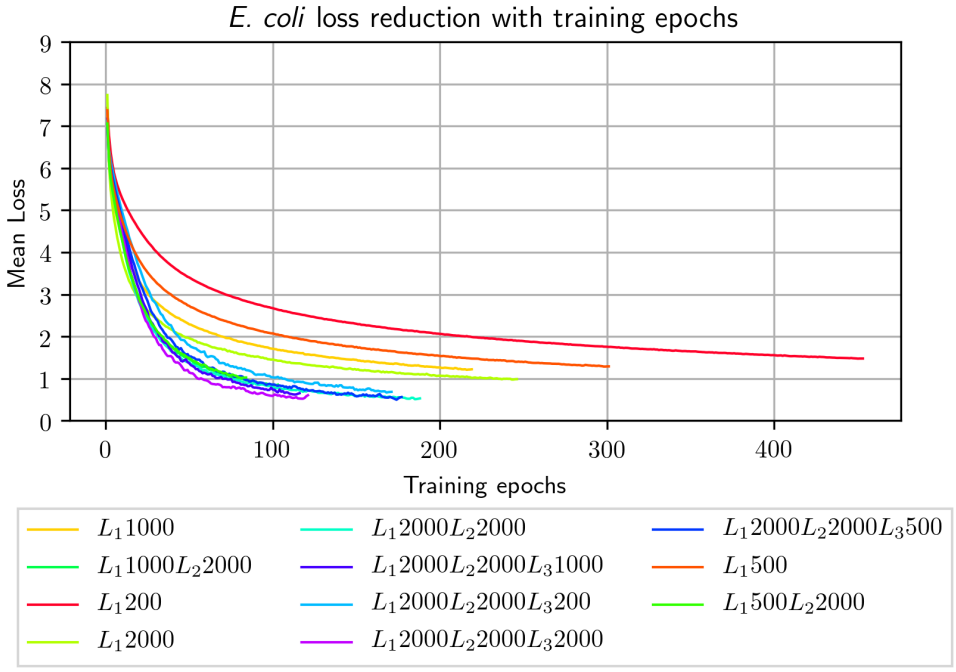


Figure 4.7: Neural network loss error during training epochs on *E. coli* dataset.

4.3 | Application of Neural Network to the Dataset

This experiment applied NN on the two taxonomy datasets namely *E. coli* and *Homo sapiens* generated specifically for ML. In this experiment the training dataset was split into eighty percent training and twenty percent validation. The NN was trained using the training termination criteria set to terminate training if the error did not reduce for three successive epochs. The experiment entailed using different hidden layers architectures. The convention used describe the neural network architecture is L_hC, where h is the hidden layer level and C is the neuron count. For example, a neural network with two hidden layers, with the first layer of 700 and a second layer of 500 is represented as $L_1 700 L_2 500$.

During the training epochs, predictions for the training datasets were generated.

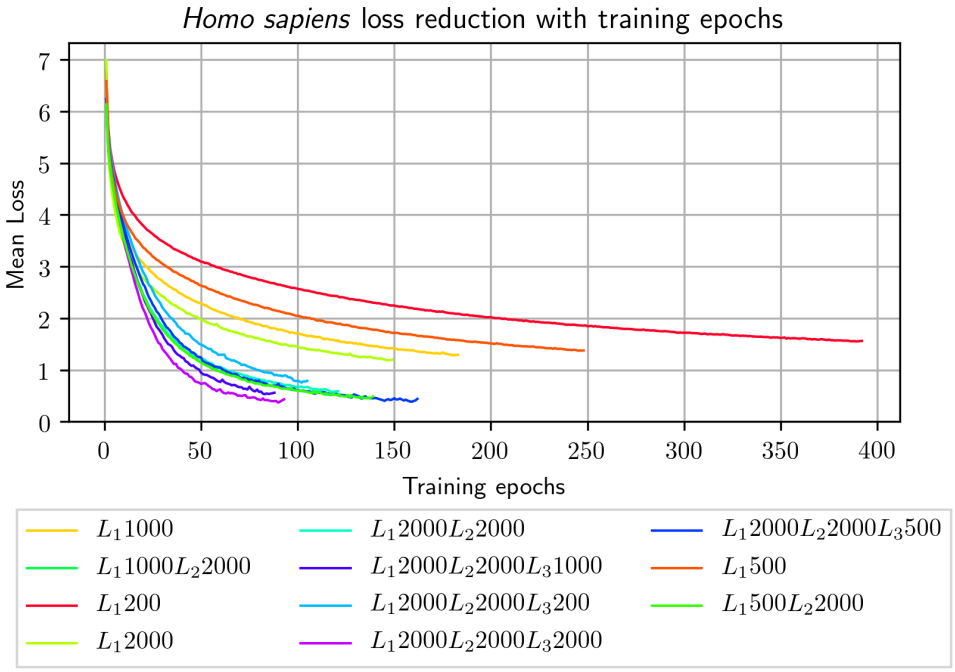


Figure 4.8: Neural network loss error during training epochs on *Homo sapiens* dataset.

The predictions and the true labels were used to compute the error of the NN. The NN optimiser uses the error to adjust the weights of the neurons. To optimise the training time, each training epoch was split into mini batches of 200 to speed up network convergence. The aim of the NN training phase is to reduce the network loss error to a minimum.

Different NN architectures were trained for the two species, namely *Homo sapiens* and *E. coli*. The loss error across the training epochs for the different NN architectures is illustrated in Figure 4.7 for *E. coli* and Figure 4.8 for *Homo sapiens*.

The loss error of the network is reduced with the training epochs. The different network architectures have been trained on the same dataset. The architecture configuration effects the training time, number of training epochs required and minimum network error. Smaller networks tend to require more training epochs and the loss error curve is smoother. In the initial phases of the training the rate of reduction loss error reduction is large, however as the number of training epochs increases; the rate of reduction decays. Table 4.5 shows the training time required for each network.

The training termination criteria was defined to terminate training after three successive epochs that did not reduce the loss error. The increase of the loss error is visible as a small increase at the end of the error loss line. Thus, the network with the smallest

Architecture	*E. coli*		*Homo sapiens*	
	Epochs	Training time (h)	Epochs	Training time (h)
L_1200	453	0.078	392	0.296
L_1500	301	0.069	248	0.412
L_1500L_22000	84	0.061	139	2.700
L_11000	219	0.099	183	0.425
L_11000L_22000	103	0.144	135	3.820
L_12000	246	0.205	149	0.640
L_12000L_22000	188	1.084	121	5.315
$L_12000L_22000L_3200$	171	0.530	105	2.424
$L_12000L_22000L_3500$	177	0.785	162	6.564
$L_12000L_22000L_31000$	116	0.191	88	3.096
$L_12000L_22000L_32000$	121	0.287	93	5.104

Table 4.5: NN architecture training duration details.

error was the one at three epochs before training stopped. For example, a network that stopped training at the k epoch, the epoch with least loss is $k-3$. In order to use the network as at epoch $k-3$, during training each network stored. The storage required to store the file for all architectures is considerably large (30-70GB), making experiment scheduling more challenging.

To assess the performance of the different networks, the validation dataset was utilised. The different neural network architectures were initialised and loaded with the weights of the third epoch from the last. After loading the weights, the predictions for the validation dataset were generated and evaluated using CAFA performance metrics.

For each set of inputs in the validation dataset, the NN outputs the predicted GO term and confidence values. The CAFA evaluation metrics evaluate the prediction and the respective confidence that determines the predictor's F_{max} at a specific confidence threshold. Figure 4.9 reports the performance of the different neural architectures on the *E. coli* validation dataset. The upper plot in the figure shows the F_{max} obtained by the specific architecture. The lower plot reports the confidence of which the corresponding F_{max} was obtained. From Figure 4.9 the different architectures had comparable F_{max}, between best and lowest F_{max} are separated by 0.050. The different models had a considerable variation in the confidence ranging from 0.070 to 0.250. The best performing

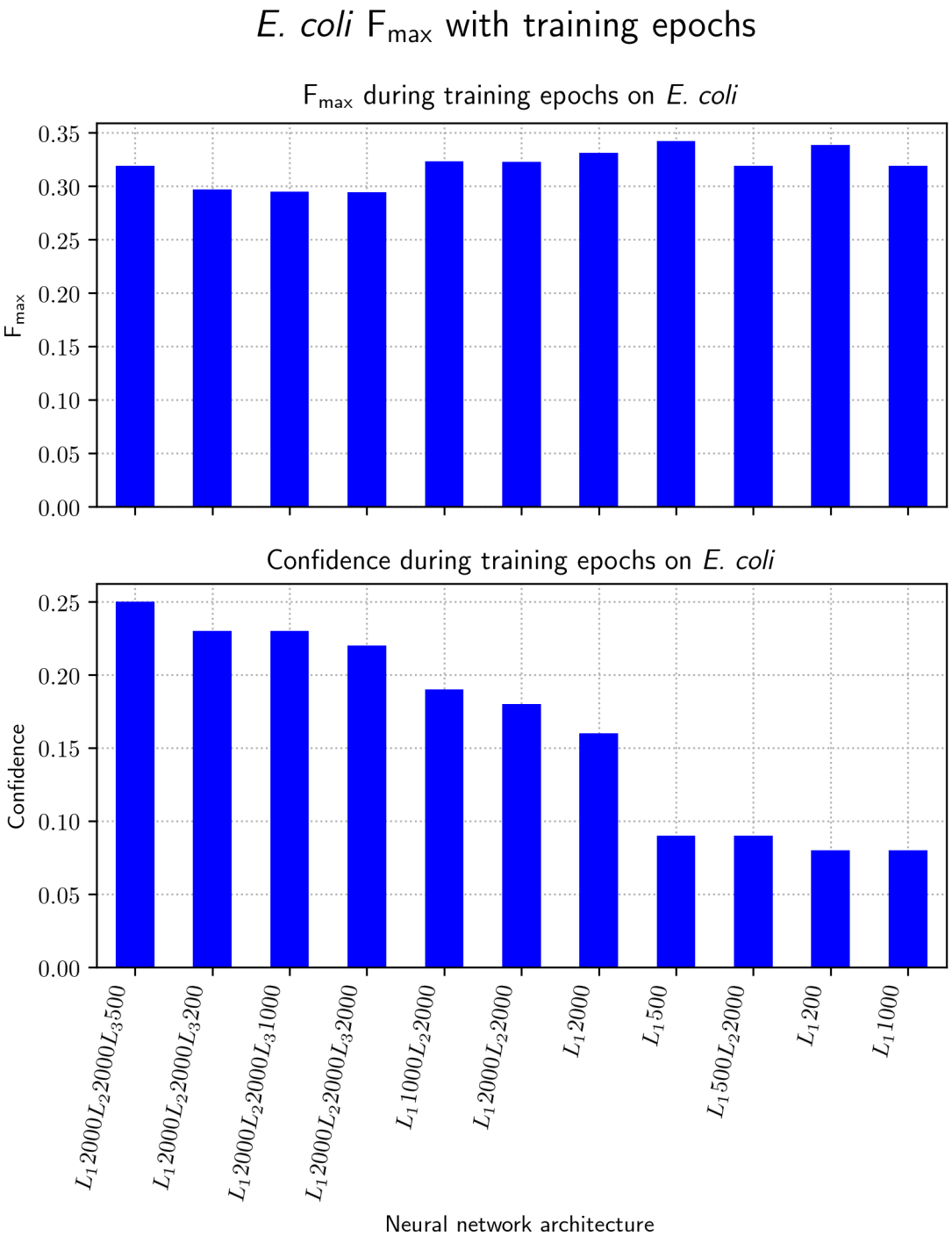


Figure 4.9: Performance of neural network architectures on *E. coli* validation dataset.

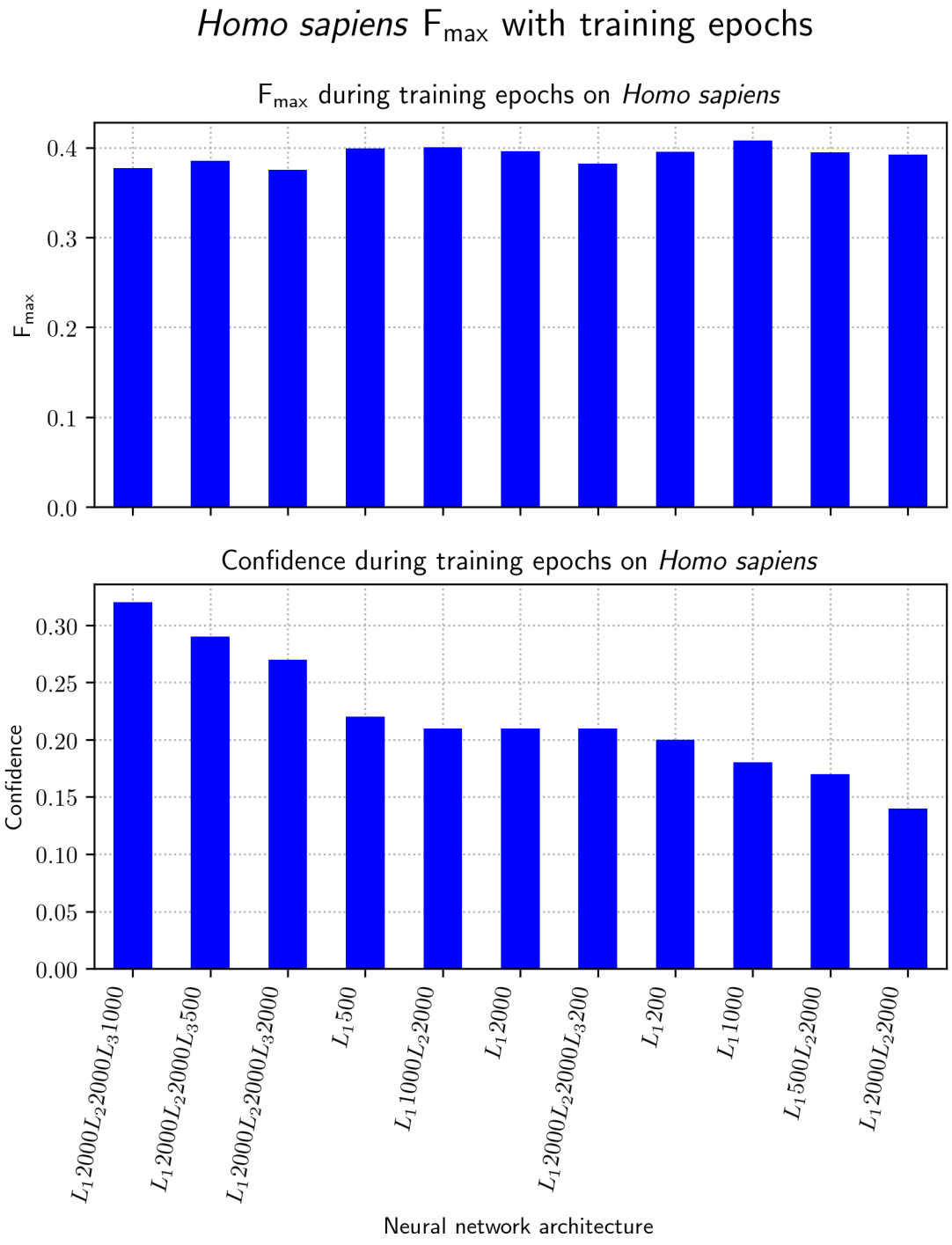


Figure 4.10: Performance of neural network architectures on *Homo sapiens* validation dataset.

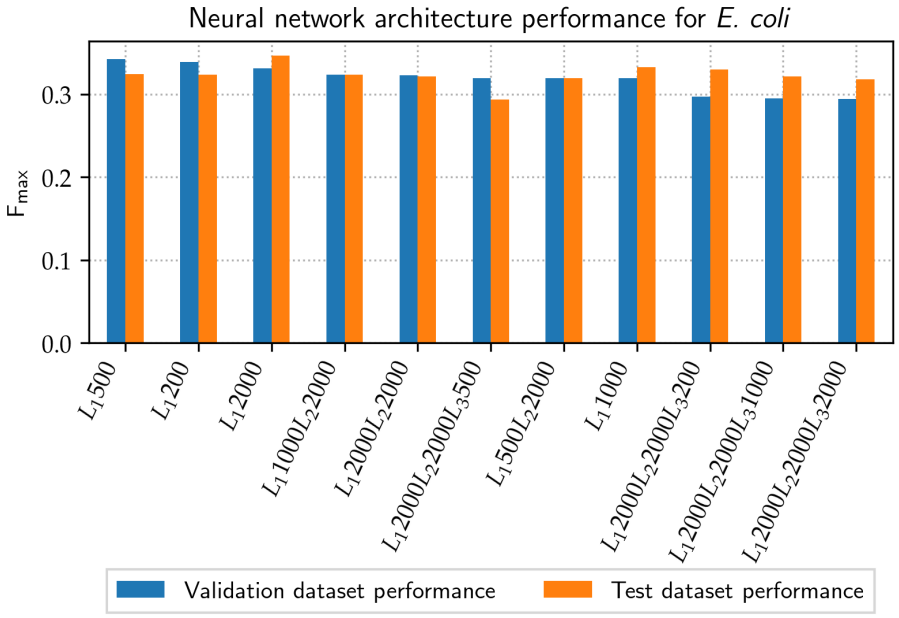


Figure 4.11: F_{max} of different neural network architectures on validation and testing dataset *E. coli*.

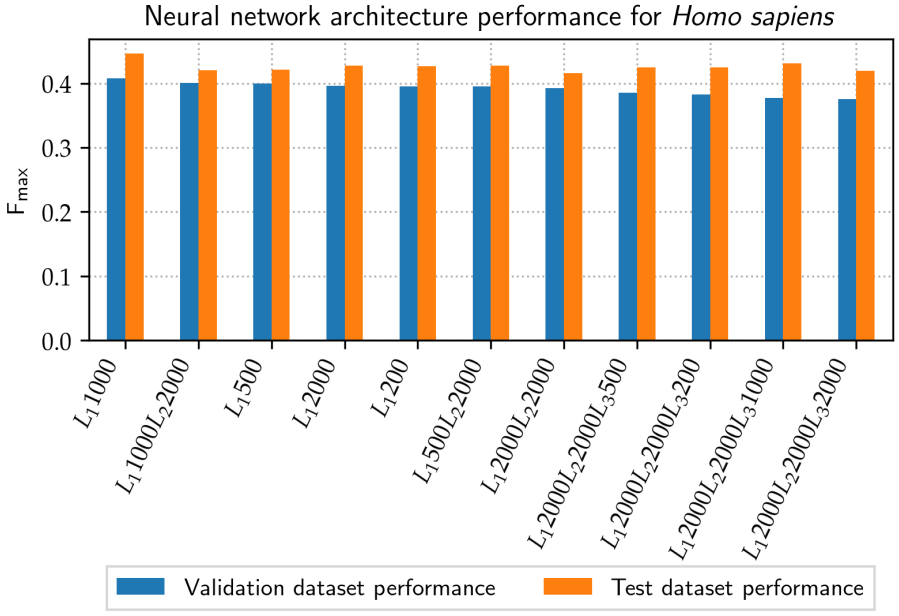


Figure 4.12: F_{max} of different neural network architectures on validation and testing dataset *Homo sapiens*.

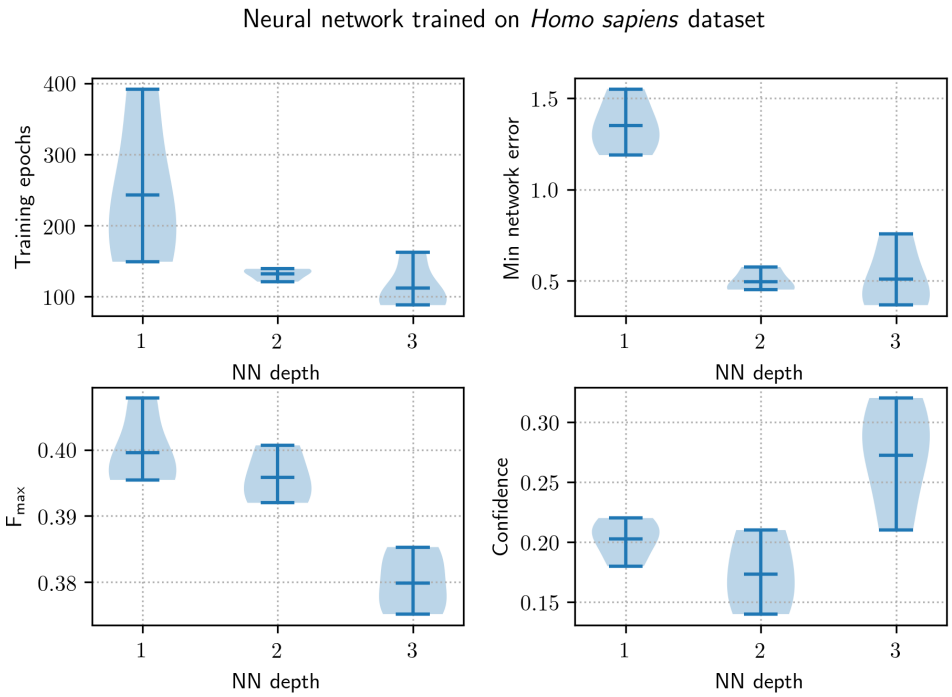


Figure 4.13: Analysis of NN on *Homo sapiens* dataset.

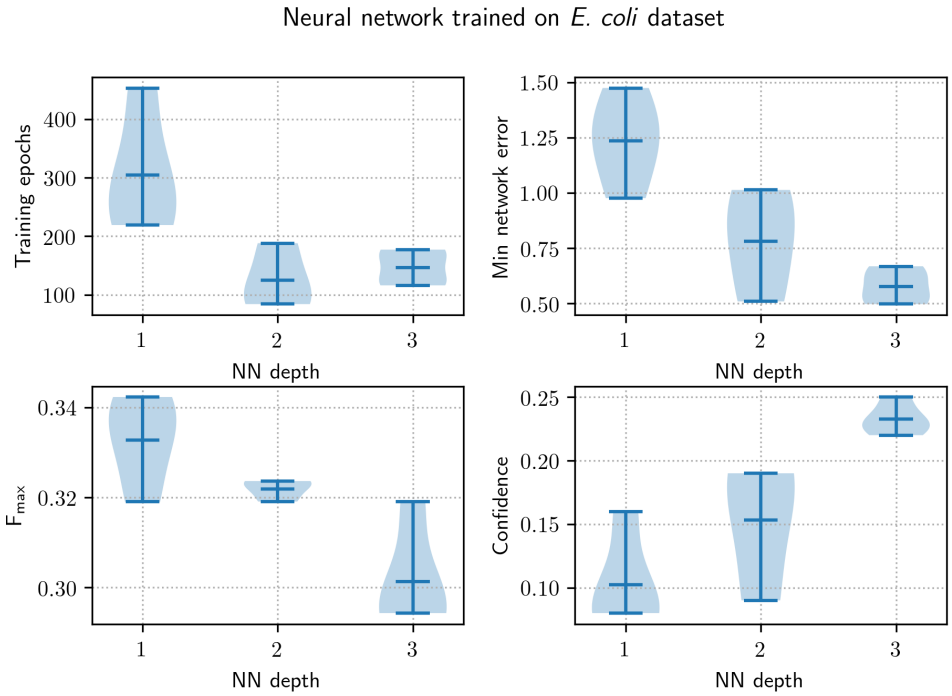


Figure 4.14: Analysis of NN on *E. coli* dataset.

architectures were $L_1 2000 L_2 2000 L_3 500$ and $L_1 2000 L_2 2000 L_3 200$. These two architectures have slightly less F_{max}, but the prediction confidence is 0.250 and 0.230 respectively.

The outcome of the tests performed on the *Homo sapiens* species is reported in Figure 4.10. The model with the least error loss was loaded and used to generate the predictions for the validation dataset. The predictions were evaluated using CAFA metric and Figure 4.10 shows the result. Similar to the *E. coli* taxon the F_{max} difference between the NN architectures was 0.040. From the confidence perspective, simpler networks tend to have lower confidence levels. The best confidence was obtained using the $L_1 2000 L_2 2000 L_3 1000$ architecture, followed by $L_1 2000 L_2 2000 L_3 500$.

Figures 4.9 and 4.10 show the performance of the different networking prioritising the prediction confidence. Analysing Figure 4.9 shows that *E. coli* the deeper networks have high confidence. Whilst for *Homo sapiens* this pattern has not emerged.

The performance of the NN architectures was tested on the test dataset to check the generalisation performance. The different NN architectures were trained on the training dataset of the specific taxon. Predictions were generated for the test dataset and the performance was evaluated using the CAFA metrics. The F_{max} on the validation dataset and on the testing dataset is reported for *E. coli* in Figure 4.11, while and for *Homo sapiens* the F_{max} is reported in Figure 4.12. Both species generalise well as the F_{max} obtained of the validation and on the test dataset are similar. In the case of *Homo sapiens* all the NN architectures had a better F_{max} on the test dataset. Although the *E. coli* NN generalise well, the small variations of the F_{max} between the validation and the testing dataset can be attributed to the small training dataset size.

The number of hidden layers affects different aspects of the neural network, related to training and predictive performance. For this purpose, the number of training epochs, minimum network error, F_{max} and prediction confidence were analysed from the NN depth perspective. Figures 4.13 and 4.14 show the method information for the *Homo sapiens* and *E. coli* trained neural networks. During training the deeper networks require less training epochs and after training, the networks have less network error. From the prediction performance perspective, the deeper networks have an F_{max} lower by 0.020 and 0.030 for *Homo sapiens* and *E. coli* respectively when compared to the one-layer networks. Deeper networks have a superior prediction confidence when compared to the one- or two-layer networks.

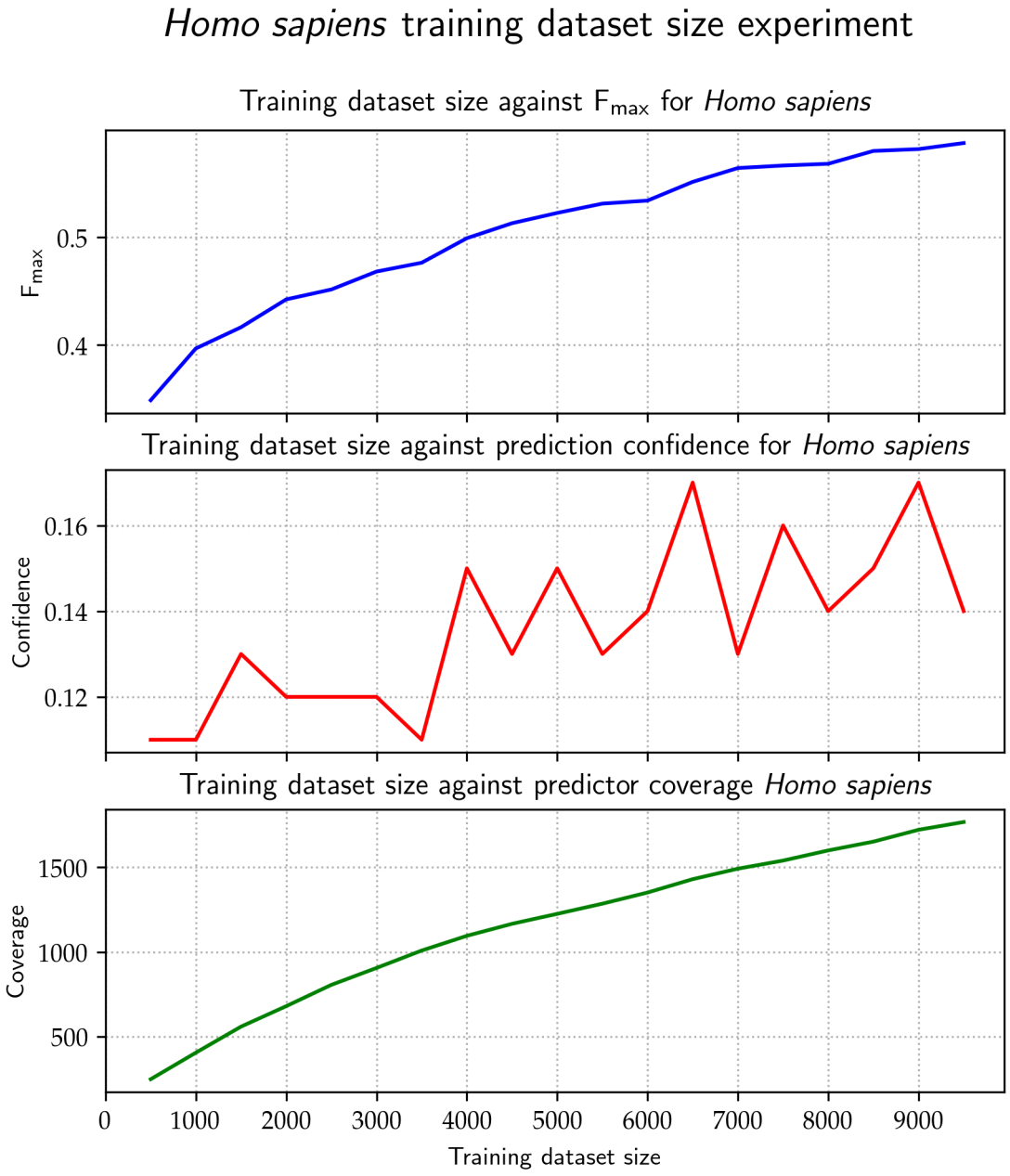


Figure 4.15: Performance of the RF-Gini classifier trained on with different amounts of *Homo sapiens* training data.

Figure 4.16: Performance of the RF-Gini classifier trained on with different amounts of *E. coli* training data.

4.4 | Determine the Amount of Training Data Needed

This experiment aims to determine the amount of training data required by the classifier. For this purpose, the best performing chromosome for each taxonomy identified by the automatic feature selection was selected. For each taxon, the training dataset is increased in steps of 500 samples and the testing dataset is kept constant. Figure 4.16 shows the experiment findings for *Homo sapiens*, while Figure 4.15 shows the findings for *E. coli*.

Each figure plots the classifier performance in terms of F_{max} and prediction confidence as computed by the CAFA evaluation metrics and training coverage against the training dataset size. Coverage is the number of unique GO terms that the classifier is able to predict.

The predictor's performance improves as training samples increases as shown in the F_{max} subplot of Figures 4.15 and 4.16. The increase of training samples increases the number of different GO terms (coverage) that can be predicted. Coverage increases linearly with the number of training samples. The increase in coverage enables the prediction of more GO terms, however the lack of sufficient training data of the new GO terms decrease the prediction confidence. This is graphically illustrated in the probability subplot in Figures 4.15 and 4.16.

The *Homo sapiens* predictor achieved an F_{max} of 0.575 with 9,500 training samples, whilst the *E. coli* predictor scored an F_{max} of 0.390 with 2,500 training samples. However, with 2,500 training samples, the *Homo sapiens* predictor had an F_{max} of 0.450. The relation between F_{max} and classifier is analysed. At coverage equal 1,000, the *E. coli* classifier required 2,400 samples available, whilst the *Homo sapiens* classifier required 3,500. With this coverage, the F_{max} for *E. coli* was 0.380 and that for *Homo sapiens* is 0.425. This illustrates that additional training samples can further improve the predictor performance. This is also further reinforced by the gradient of the F_{max} line within both plots. For both taxonomies, the gradient of the F_{max} line decreased slightly but is still away from approaching a plateau. This highlights that for both taxonomies additional training samples can potentially improve classifier performance.

4.5 | Cross Species Model Experiments

This control experiment applied two species trained classifiers to proteins originating from different taxonomies. For this experiment the most performant ML technique, RF using Gini as splitting criterion was utilised (RF-Gini). The ML classifier was trained

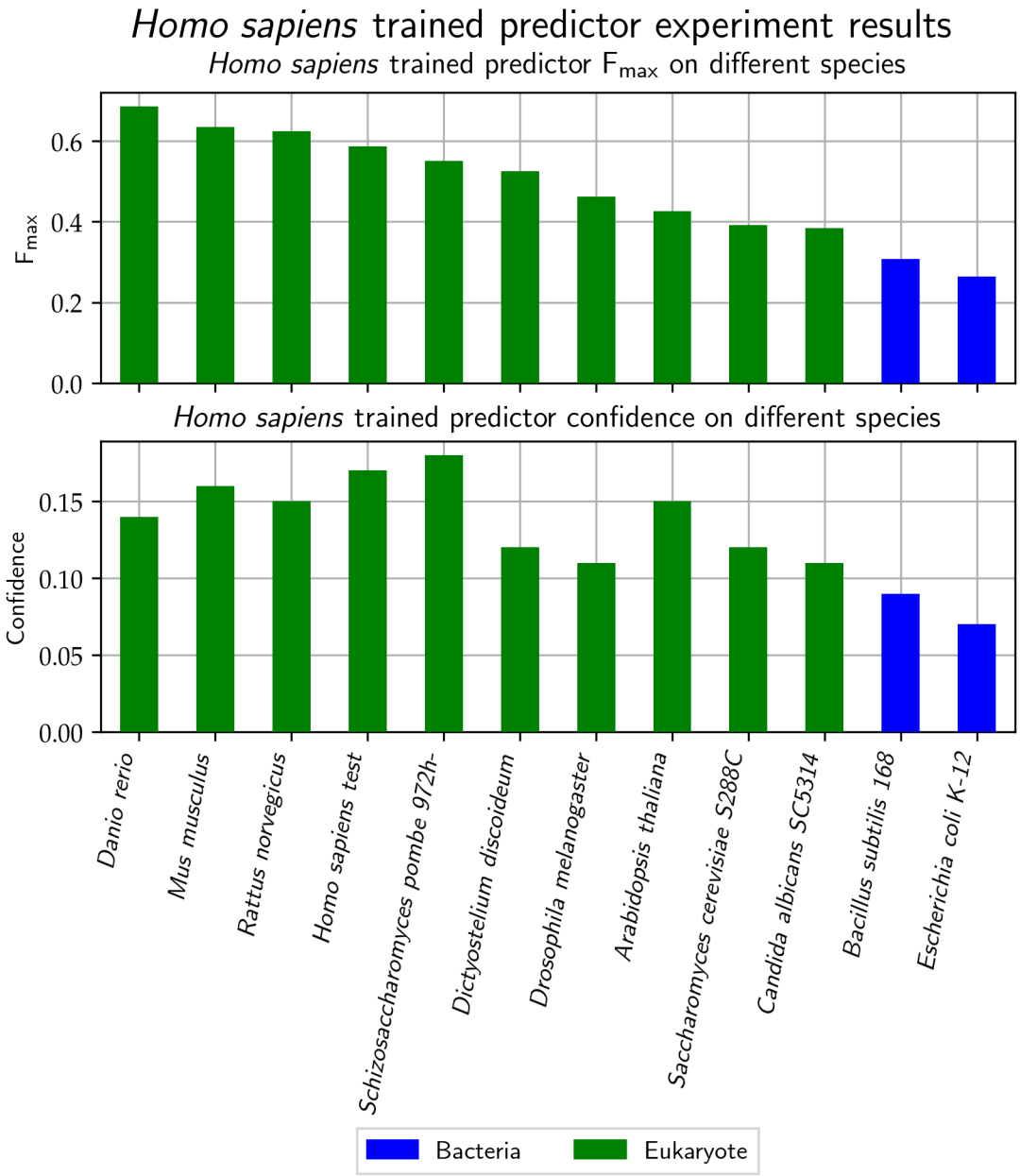


Figure 4.17: F_{max} performance of *Homo sapiens* trained classifier applied on different taxa.

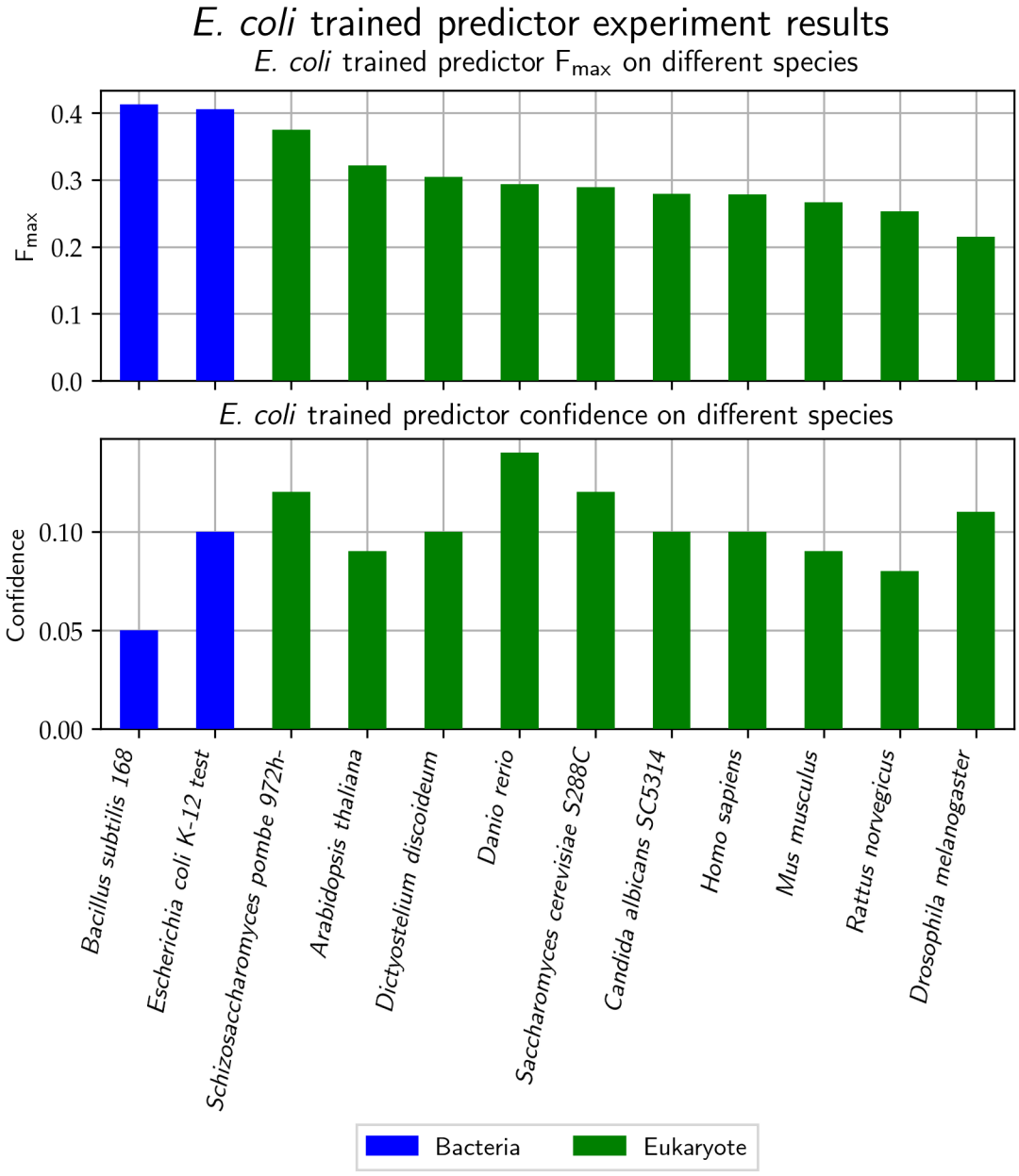


Figure 4.18: F_{max} performance of *E. coli* trained classifier applied on different taxa.

on the training data using the top performing GA chromosome. The trained classifier was used to perform cross species prediction. The expected outcome of the experiment is that the classifier performance will decrease as the distance between taxonomies increases.

Figure 4.17 illustrates the performance of the classifier trained on *Homo sapiens* data and applied to other species. The classifier performs well on Eukaryote taxonomies that are evolutionary near to *Homo sapiens*. The performance of the classifier starts to decay as the evolutionary distance increases. The lowest F_{max} on the Eukaryote is 0.390, whilst the highest F_{max} on Bacteria is 0.300.

The *E. coli* trained classifier was trained and applied on the dataset for other taxonomies. The performance is reported in Figure 4.18. The classifier's best performance was on Bacteria domains, with an F_{max} of 0.410. The performance of the classifier decreased on Eukaryote organisms with F_{max} ranging from 0.360 to 0.210.

4.6 | Summary

This chapter reported and discussed the findings of the different experiments. The first section reported the results of the feature selection that was executed on the two taxa, namely *Homo sapiens* and *E. coli*. The results include the performance metric on the validation dataset and as information related to execution time. Different aspects of the techniques such as execution time were reported. The next section applied GA within the cross validation loop to measure the classifier performance when trained with different data. The next section tackled the application of NN on the two species specific datasets. Different architectures were experimented on both datasets. For each NN the loss error is reported during the training epochs. Neural network training was configured to stop training if the loss error fails to reduce after three successive epochs. Consequently, the third epoch from the last was the epoch with least loss error. This epoch was selected to benchmark the network using the validation dataset. The next section investigated the effect of the training dataset size on the classifier performance. The training dataset size was increased in steps of five hundred and the classifier F_{max}, F_{max} confidence and coverage were reported. The final experiment is a controlled experiment, whether the species-specific classifier is applied to other species. The next chapter reports the work carried out to evaluate the work against CAFA submissions.

5

Evaluation

The outcomes of the defined experiments were reported and discussed in the previous chapter. In order to gauge the effectiveness of the work performed it must be benchmark against other work using the same methodology. This chapter details the evaluation protocol and evaluates the work against other CAFA submissions.

5.1 | Evaluation Protocol

Protein annotations are represented in biological databases using GO terms defined in the GO DAG. An important aspect of protein annotations is the concept of hierarchy, that enables description of generic and specific functionality. Laboratory experiments add annotations to proteins that were proven in the lab using specific protocol. In some cases, experiments evidence lack of functionality, thus remove annotations that are not observed in the laboratory. Further different experiments on the same protein add functionality that was not observed before using more specific annotation term from with the GO DAG, possibly extending the GO DAG to cover the functionality observed in the experiments.

The motivation of the CAFA shared task protocol, is to gauge the status of Protein Function Prediction (PFP). To measure the effectiveness of PFP methods, their performance must be measured against verified ground truths. The CAFA shared task is set up to replicate the environment whereby a classifier is used to predict the protein function that is not yet experimentally annotated. The performance of the method is measured after the experimental curation confirms the function or the lack thereof. The CAFA evaluation framework is described in section 2.9.

5.2 | CAFA Evaluation Metrics

The CAFA evaluation uses three metrics defined in section 2.9.2 to benchmark submissions. For each submission, CAFA reports three metrics, the F_{max}, the semantic distance S_{min} and predictor coverage. Using these three figures the performance of the predictors can be compared.

$$pr(\tau) = \frac{1}{m(\tau)} \sum_{i=1}^{m(\tau)} \frac{\sum_f \mathbb{1}(f \in P_i(\tau) \wedge f \in T_i)}{\sum_f \mathbb{1}(f \in P_i(\tau))} \tag{2.26}$$

$$rc(\tau) = \frac{1}{n_e} \sum_{i=1}^{n_e} \frac{\sum_f \mathbb{1}(f \in P_i(\tau) \wedge f \in T_i(t))}{\sum_f \mathbb{1}(f \in T_i(t))} \tag{2.27}$$

$$F_{max} = \max_{\tau} \frac{2.pr(\tau).rc(\tau)}{pr(\tau) + rc(\tau)} \tag{2.28}$$

$$ru(\tau) = \frac{1}{n_e} \sum_{i=1}^{n_e} \sum_f ic(f).\mathbb{1}.(f \notin P_i(\tau) \wedge f \in T_i) \tag{2.29}$$

$$mi(\tau) = \frac{1}{n_e} \sum_{i=1}^{n_e} \sum_f ic(f).\mathbb{1}.(f \in P_i(\tau) \wedge f \notin T_i) \tag{2.30}$$

$$S_{min} = \min_{\tau} \sqrt{ru(\tau)^2 + mi(\tau)^2} \tag{2.31}$$

where:

$P_i(\tau)$ = the set predicted with score equal or greater than τ for a protein sequence.
T_i = the true positive set the sequence.
$m(\tau)$ = the number of sequences with at least one score greater than or equal to τ.
$\mathbb{1}$ = the indicator function that returns 1 when the condition is true otherwise 0.
n_e = the number of targets in evaluation.
$ic(f)$ = the information content for term f.
τ = a value for 0.01 to 1.00 incremented by 0.01.

The precision metric defined in equation 2.26 assesses the performance of the classifier in terms of correctly predicted terms in respect to all predictions. Whilst recall metric defined in equation 2.27 assesses the performance in terms of correctly predicted terms in respect of ground truths. The F_{max} metric defined in equation 2.28 is the harmonic mean of precision and recall metrics. The value of F_{max} ranges between zero and one, with one being the F_{max} value of the ideal classifier and zero for a poor classifier.

CAFA reports the F_{max} results using two visualisations a bar chart or using a precision-recall curve. The precision-recall curve is more informative as it illustrates the precision

and recall values of the classifier and the respective F_{max} value as a contour. During evaluation the predictor annotations are filtered based on the confidence interval generating a series of precision and recall values. These values are plotted on the precision-recall curve, highlighting the best F_{max} value. For evaluation purposes, the predictor F_{max} value is the highest F_{max} at the highest prediction confidence (Jiang et al., 2016).

The semantic distance S_{min} computes an information theoretic metric based on the GO DAG structure and a Bayesian probabilities. The semantic distance S_{min} metric is based on the misinformation metric defined in equation 2.30 and remaining uncertainty metric defined in equation 2.29. These two metrics assess the performance of the classifier in terms of excessive information predicted (to highlight predictor over prediction) and missing annotations (to highlight lack of predictions). The semantic distance S_{min} is the euclidean distance of the misinformation and remaining uncertainty metrics from the origin as defined in equation 2.31. The value of semantic distance is unbounded, however the lower the value the better the classifier performance.

CAFA computes the performance of the classifier at different predictor confidence intervals generating multiple misinformation and remaining uncertainty values. CAFA reports S_{min} using two visualisations types, a bar chart and a remaining uncertainty-misinformation (RUMI) curve. The RUMI curve provides a better picture as the Misinformation and Remaining Uncertainty are plotting and the S_{min} is shown as a contour. The semantic distance considered for the evaluation is the minimum value of semantic distance value at the highest prediction confidence (Jiang et al., 2016).

The third aspect tackled by the CAFA evaluation is predictor coverage. The benchmark set is the set of annotations that will be used for evaluation. The predictor performance will be evaluated on the benchmark set annotations. The coverage is the ratio of the benchmark set the predictor can predict. The ideal predictor will have a ratio of one, implying that all the terms in the benchmark set can be predicted by the classifier (Jiang et al., 2016).

5.2.1 | Approach

Within the CAFA shared task, the time elapsed between the prediction generation (by the participants) and the execution of the evaluation is around six months. The aim was to evaluate this work against CAFA3 shared task, however this option was not possible as the ML features computed for the CAFA3 targets were not available. Following communication with Dr Lees, a dataset with the protein features for CAFA3 targets was made available. This dataset was used to generate species-specific datasets, namely *Homo sapiens* and *E. coli* to evaluate the different models. The selected methods for each

ML were used to generate predictions for the CAFA3 targets. The CAFA performance metrics were generated for each model.

The paper reporting the results of the CAFA3 shared task was not available from the CAFA shared task website[1]. An enquiry for the CAFA3 results was sent via the CAFA website. Prof. Iddo Friedberg (the contact person for CAFA) replied stating that the paper is currently works in progress and currently aiming to publish a pre-print format in a couple of months. Email exchange with Prof Iddo Friedberg is detailed in Appendix E.

In view of these constraints, the work will be evaluated using the prediction of the CAFA3 targets against the CAFA2 published results per taxonomy. This option offers a good indication of the performance of the models.

5.2.2 | Evaluation Protocol Used

To replicate the CAFA evaluation protocol, the annotations of the CAFA3 targets used in the prediction and evaluation phase must originate from a more recent version of QuickGO. Different versions of QuickGO are available from EMBL-EBI FTP site[2]. The evaluation protocol used was:

1. Use the species-specific datasets generated in the experimentation phase for training. These datasets are based on annotations of QuickGO version 124.
2. Train the ML methods using dataset features identified by the GA and the NN architectures identified during the experimentation phase with the training dataset identified in step 1.
3. Using the process outlined in section 3.5, generate two species-specific datasets for the CAFA3 targets, based on annotations of QuickGO version 152.
4. Using the methods trained in steps 2 generate the predictions for the datasets generated in step 3.
5. Generate the true positive set for the CAFA3 targets for each species based on QuickGO version 161. The outcome of this step is a true positive set for each species.
6. Evaluate the predictor performance using the predictions generated in step 4 and true positive set generated in step 5.

[1] `https://biofunctionprediction.org/cafa/` (Accessed 2018-12-30)
[2] `ftp://ftp.ebi.ac.uk/pub/databases/GO/goa/old/UNIPROT/` (Accessed 2018-12-28)

Dataset usage	Ground Truths (GT)	GT GO DAG
Training	QuickGO version 124	2013-10-15
Prediction	QuickGO version 152	2016-06-01
Evaluation	QuickGO version 161	2017-01-01

Table 5.1: Data sources used in evaluation.

The GO DAG provides the vocabulary used to describe protein functionality. A consequence of this is that functionality not described in the GO DAG cannot be described. For this purpose, the GO DAG is updated through a pre-defined protocol to include new functionality as it is observed in the lab. The different GO DAG versions are available from Gene Ontology FTP site[3]. The QuickGO annotations of the CAFA3 targets in the evaluation dataset require a more current version of GO DAG as per Table 5.1.

The functionality described in more recent version of QuickGO, requires a more current GO DAG. The rate of change within the GO DAG is low. The GO DAG dated 2016-06-01 has 10,236 GO terms and 672 GO term synonyms for the molecular function ontology. Whilst version 2017-01-01 of GO DAG released six months later has 10,543 GO terms and 687 GO term synonyms. Over a period of six months, 307 (0.030%) new GO terms were added to the GO DAG.

Taxon	QuickGO	Sequences	Unique GO terms	Missing Terms
E. coli	161	339	478	0
Homo sapiens	161	1,051	1,159	5

Table 5.2: True positive set GO terms existence in GO DAG 2016-06-01.

The functionality described in the more recent versions of GO DAG is inexistent in older GO DAG versions. In order to determine whether the true positive set generated based on QuickGO version 161 can be represented using the GO DAG as at prediction date. Table 5.2 reports the findings.

The GO terms in the true positive set used to describe the functionality of the *E. coli* taxon (based on QuickGO 161) can be represented in the GO DAG 2016-06-01. This is not the case for *Homo sapiens*, whereby five GO terms used to describe the function in QuickGO 161 does not exist in the GO DAG version aligned with QuickGO 152.

[3] `ftp://ftp.geneontology.org/go/ontology-archive/` (Accessed 2018-12-28)

GO term in true positive set	Replaced with parent GO term
GO:0002950	GO:0016780
GO:0097726	GO:0019904
GO:0102077	GO:0016811
GO:0102549	GO:0052689

Table 5.3: GO terms replaced in the true positive

The evaluation process needs to factor in the new annotation information acquired during the time lapse. For this purpose, the true positive set must be modified to retrofit the new information using the training version of the GO DAG. Whilst generating the true positive dataset to be used for evaluation from the QuickGO dataset, GO terms that do not exist in the training GO DAG are replaced with their immediate parent (more generic term). Table 5.3 lists the GO terms replaced in the true positive set.

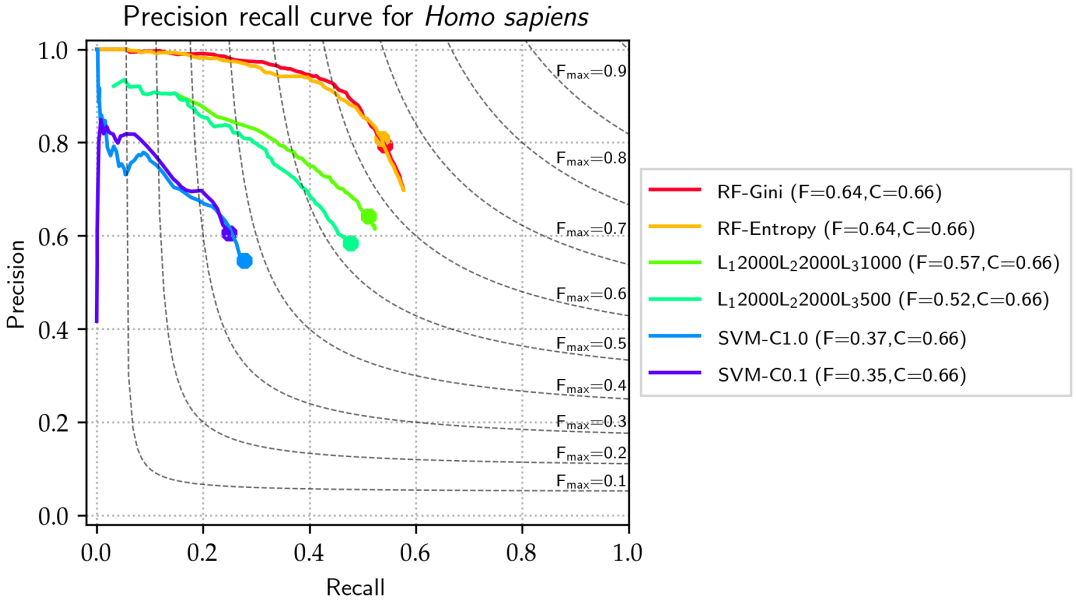
In view of the rate of change of the GO DAG which is 0.030% in six months, the amount of error introduced by substituting GO terms with a more generic parent is negligible. Further in the evaluation phase, the error is minimised as all the terms to the ontology root will be part of the sub graph through term enrichment. Only the more specific term will be omitted as it is inexistent on the GO DAG.

5.3 | Model Performance Assessment

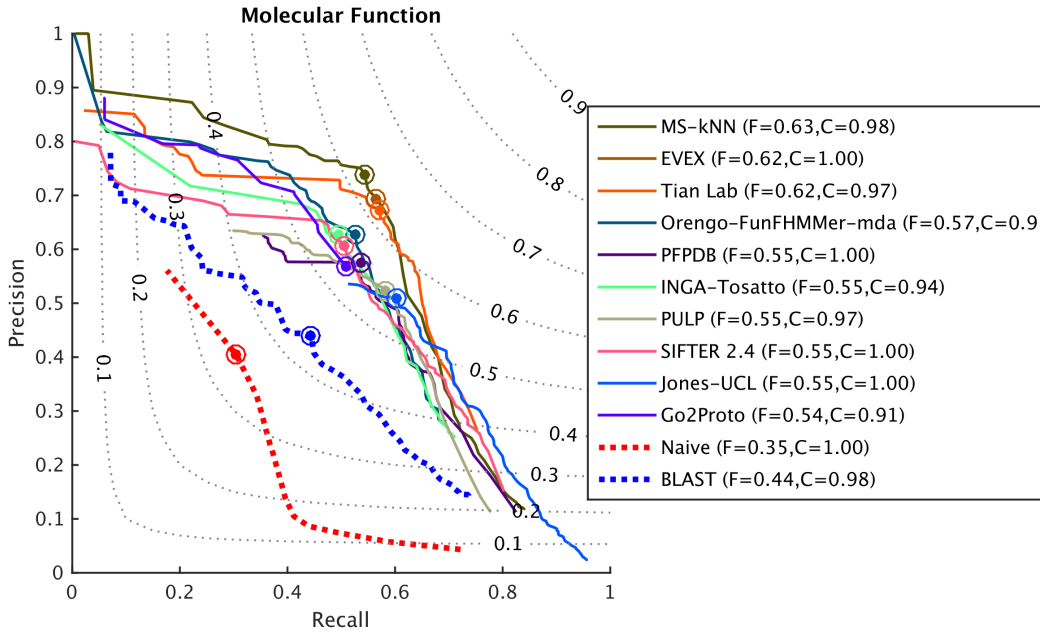
The ML models for *Homo sapiens* were evaluated on the CAFA3 evaluation data using the annotations present in QuickGO 161. The performance of the *Homo sapiens* models is reported as precision-recall curves in Figure 5.1(a) and as RUMI curves in Figure 5.2(a).

The *Homo sapiens* precision-recall curves show that the best F_{max} was achieved by the RF-Gini classifier. An interesting aspect of the performance of the models is that the two RF models score the same F_{max}. Whilst the two SVM methods are separated by an F_{max} of 0.02. This highlights that the ML tuning performance, had a minor impact on the classifier performance. In the case of NN, the different architecture separated the two methods by an F_{max} of 0.05. In the precision-recall curve, the methods are grouped per ML technique, with RF being the best, followed by NN and SVM.

The CAFA evaluation uses semantic distance (S_{min}) to report the performance of the classifier through RUMI curves. Figure 5.2(a) shows the RUMI curve of the *Homo sapiens* methods. The best methods have the lowest S_{min}. Similar to the precision-recall curve,

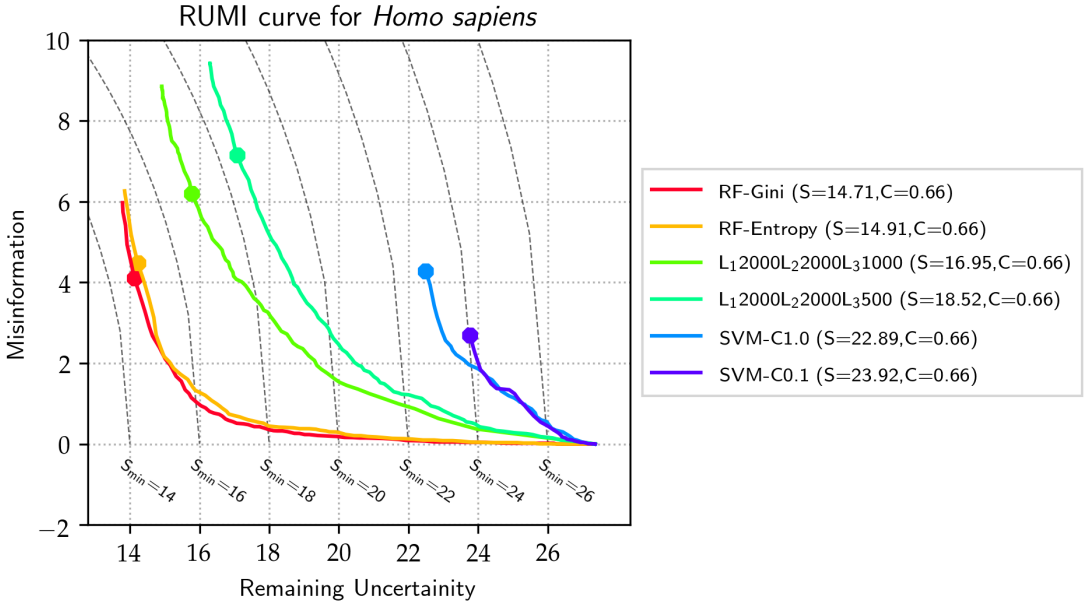


(a) Precision-recall curve for *Homo sapiens* trained methods.

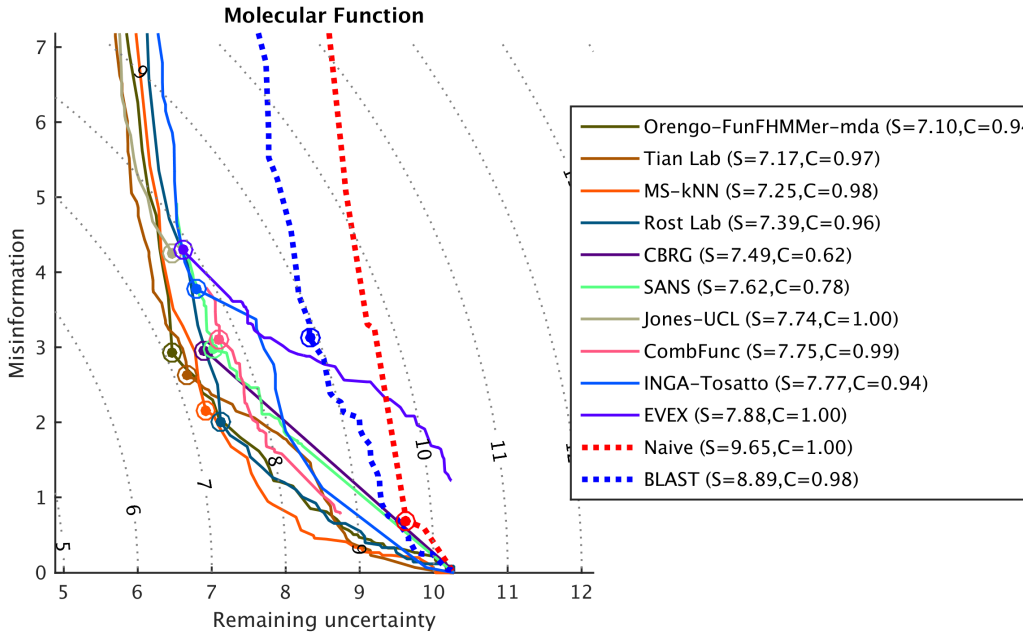


(b) Precision-recall curve for *Homo sapiens* CAFA2 submissions, reproduced from Friedberg et al. (2016).

Figure 5.1: Precision-recall curve for *Homo sapiens* trained methods and CAFA2 submissions. The best F_{max} is illustrated with a dot. The legend shows method description, best F_{max} (F) and predictor coverage (C).

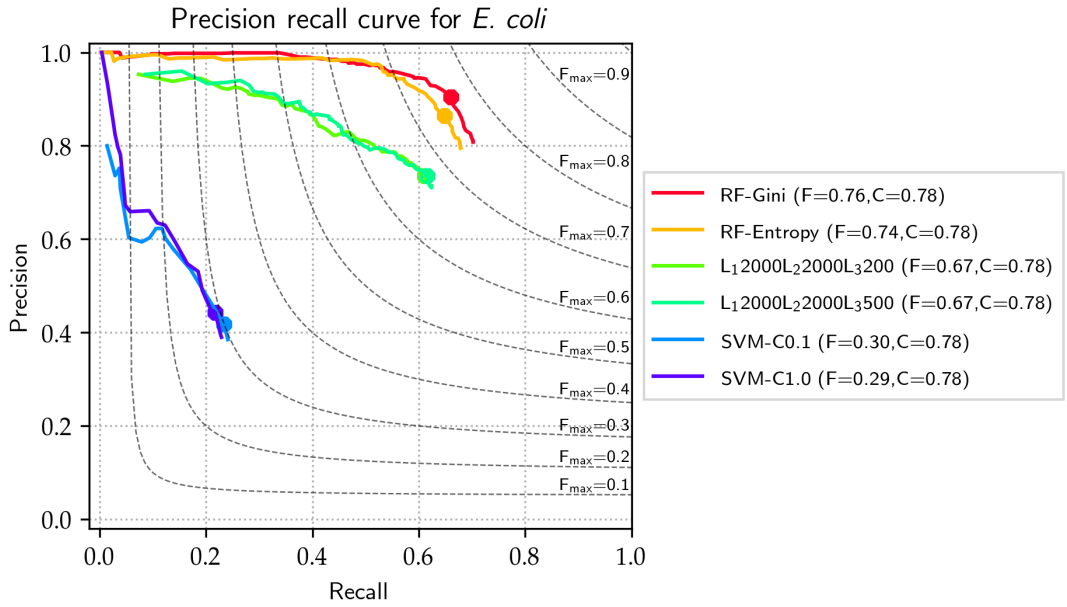


(a) RUMI curve for *Homo sapiens* trained methods.

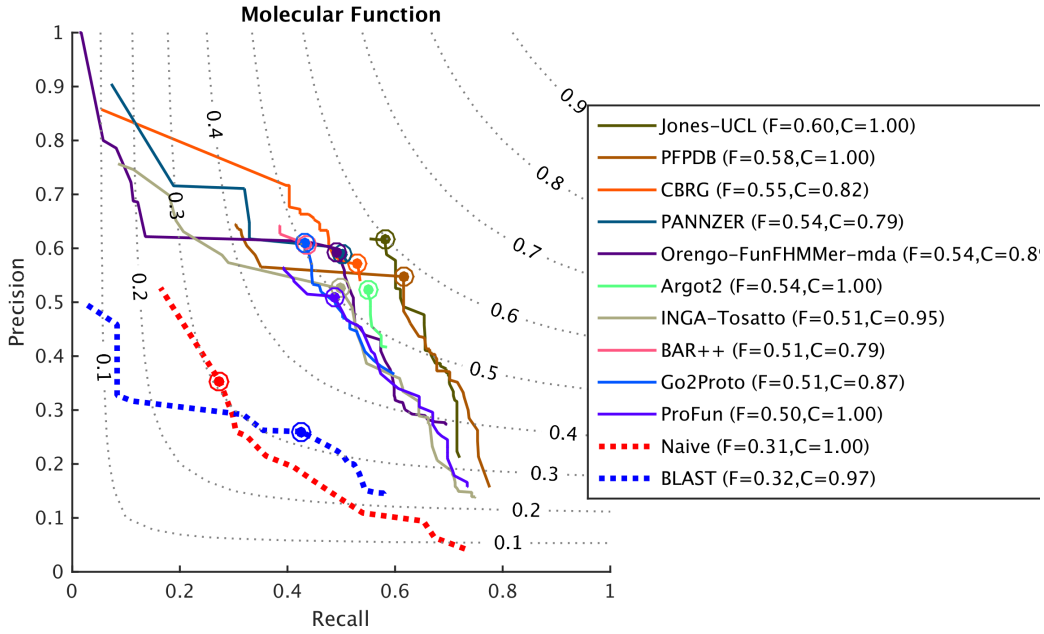


(b) RUMI curve for *Homo sapiens* CAFA2 submissions, reproduced from Friedberg et al. (2016).

Figure 5.2: RUMI curve for *Homo sapiens* trained methods and CAFA2 submissions. The best F_{max} is illustrated with a dot. The legend shows method description, best S_{min} (S) and predictor coverage (C).

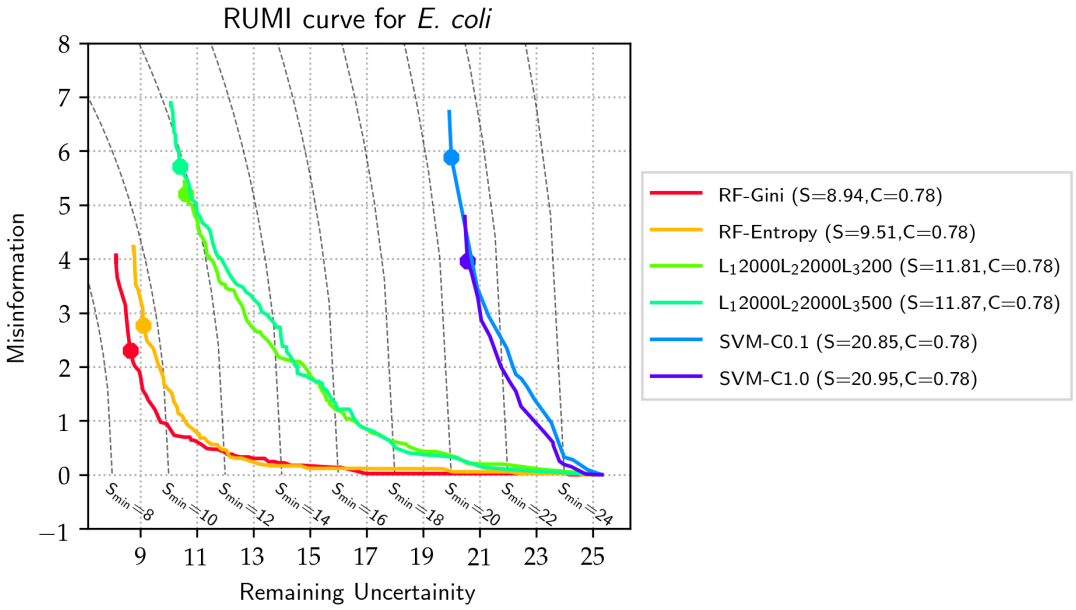


(a) Precision-recall curve for *E. coli* trained methods.

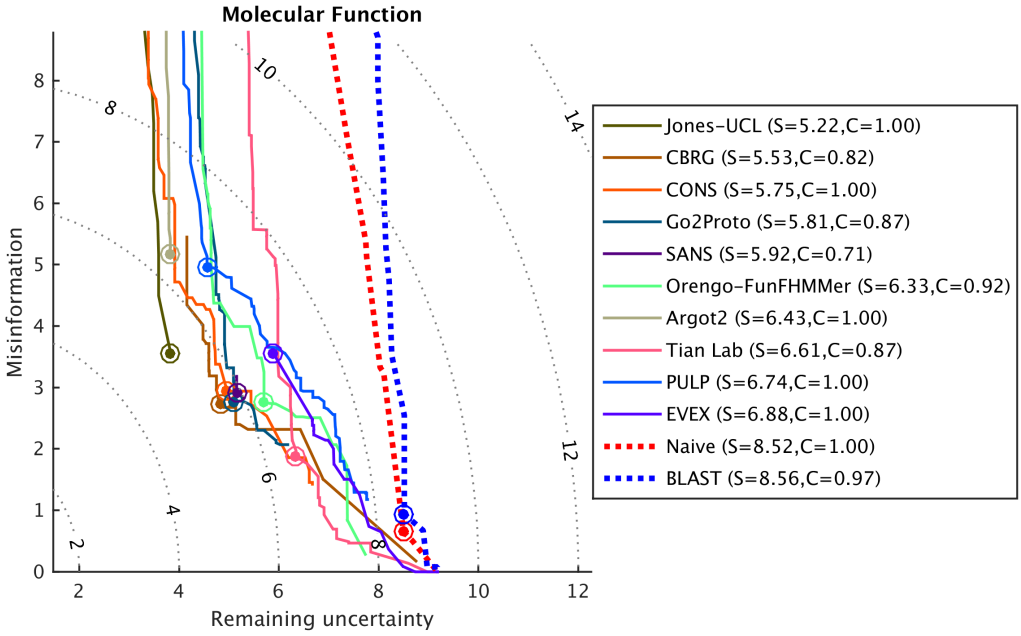


(b) Precision-recall curve for *E. coli* CAFA2 submissions, reproduced from Friedberg et al. (2016).

Figure 5.3: Precision-recall curve for *E. coli* trained methods and CAFA2 submissions. The best F_{max} is illustrated with a dot. The legend shows method description, best F_{max} (F) and predictor coverage (C).



(a) RUMI curve for *E. coli* trained methods.



(b) RUMI curve for *E. coli* CAFA2 submissions, reproduced from Friedberg et al. (2016).

Figure 5.4: RUMI curve for *E. coli* trained methods and CAFA2 submissions. The best F_{max} is illustrated with a dot. The legend shows method description, best S_{min} (S) and predictor coverage (C).

the methods are group by ML methods, whereby the best performance was that of RF, followed by NN and SVM. From the RUMI curve it can be noted that the different classifiers have a low misinformation rate indicating that the predictions done were close to the true positive set. However, the high remaining uncertainty, shows that the classifiers are missing predictions. An interesting aspect illustrated by the RUMI curve is that RF has the lowest remaining uncertainty, followed by NN and SVM. On the other hand, the SVM with penalty set to 1.0 (SVM-C1.0) had the lowest misinformation whilst the NN had the highest misinformation.

The coverage of the *Homo sapiens* models is 0.66. This implies that the classifiers did not have knowledge of 33% of the GO terms present in the evaluation dataset.

The performance of the *E. coli* classifier is reported as precision-recall curve in Figure 5.3(a) and as RUMI curve in Figure 5.4(a). The best performing model on *E. coli* is the F_{max} of 0.76. From the precision-recall curve the different ML methods are grouped tightly within an F_{max} of 0.02. The best method was the RF, followed by NN with an F_{max} difference of 0.04 and trailing behind is the SVM method with an F_{max} difference of 0.39. The RF method has a precision of 0.9, implying that most of the predictions are correct.

The performance of the methods is evaluated using the RUMI curves is reported in Figure 5.4(a). From the RUMI curve the performance difference of the models confirmed with RF being the best method, followed by NN and SVM. The good prediction performance of the RF is confirmed by a low misinformation. Similar to the *Homo sapiens* models the predictors have a high remain uncertainty.

The coverage of the *E. coli* classifiers is 0.76, improves the likelihood of better prediction performance. The classifiers have higher precision in the precision-recall curve and lower misinformation in the RUMI curve, when compared to the *Homo sapiens* counterpart.

The *Homo sapiens* and *E. coli* RF models, have superior F_{max} and S_{min} when compared to the respective NN and SVM models. The performance advantage of the RF models may originate from a better fitting of the data being used in this study.

5.4 | Performance Assessment in CAFA

In the CAFA shared task evaluation, community submissions are evaluated against the proteins that acquired manually curated annotations. These annotations are used to generate the true positive sets (ground truths) for the evaluation process. Each CAFA submission consists of a file that contains the predictions in CAFA format. The for-

mat mandates three fields: the CAFA target sequence identifier, the predicted GO term and the prediction confidence. The prediction confidence is generated by the technique utilised to generate the predictions.

The CAFA organisers report the performance of the models for each of the three GO ontologies. The prediction performance is taxon specific, thus the CAFA shared task evaluation generates performance metrics per taxon such as *E. coli* and *Rattus* and per domain such as Eukaryote, Bacteria and Archaea.

The CAFA results include two baseline methods, Naïve and Blast. The Naïve method is based on term frequency, whilst Blast assigns GO term using sequence similarity computed based on sequence alignment. The baseline methods are included in the CAFA figures together with the top 10 methods performing submission for the target taxon or domain. In the CAFA visualisations, the twelve methods are listed ranked from best performing down to the tenth best performance and the two baseline methods. For each submission the visualisation shows the name of the submission is shown together with the best performance and the coverage of the method.

The precision-recall curve for the top 10 *Homo sapiens* is reported in Figure 5.1(b) and the remaining uncertainty-misinformation curve is reported in Figure 5.2(b). Comparing the performance of the top 10 CAFA2 models reported in Figure 5.1(b), the RF models reported in Figure 5.1(a) would rank first by an F_{max} improvement of 0.01. The best model in the CAFA2 evaluation is the MS-kNN, that achieved an F_{max} of 0.63 with a coverage of 0.98. The slightly higher F_{max} of the RF-Gini method is attributed to an improvement of both precision and recall. The best NN model would have ranked fourth achieving an F_{max} of 0.57 which is at par with the Orengo-FunFHMMer-mda submission. The best SVM based method achieved an F_{max} of 0.37 which was comparable to the Naïve method.

The RUMI curves for the trained predictors and the CAFA2 submission are reported in Figure 5.2(a) and Figure 5.2(b) respectively, show that the CAFA2 submissions are superior. The semantic distance of the CAFA2 submissions is between 7.10 and 9.65, whilst the 6 models evaluated had a semantic distance between 14.71 and 23.92. From the RUMI curve, the misinformation for the RF-Gini model is 4.1, which is on the upper bound of the top 10 submissions of the CAFA2. The remaining uncertainty of all the models is high ranging from 14 to 24, whilst the values top 10 CAFA2 models range from 7 and 10. This shows that although the classifier is comparatively good in making predictions, the predictors are missing predictions.

The precision-recall curve for the top 10 CAFA2 submissions for *E. coli* is reported in Figure 5.3(b). Comparing the performance of the models against CAFA2 submissions, shows that the best model with an F_{max} of 0.76 is superior to the best method Jones-

UCL that has an F_{max} of 0.60. Analysing the classifier performance, it shows that the RF classifiers have a precision of 0.90 and 0.86, which is 0.25 higher than the best CAFA2 submission. The best recall was achieved by RF-Entropy 0.63, which is at par with the best CAFA submission of the *E. coli* species.

The RUMI curve for the top 10 CAFA2 *E. coli* submissions is reported in Figure 5.4(b). The CAFA2 submissions have a lower S_{min} when compared to the six methods. The 10th best performing submission has an S_{min} of 6.88, whilst the best performing method RF-Gini has an S_{min} of 8.94. The six models have a high semantic distance ranging from 8.94 to 20.952. Analysing the RUMI curve of the six methods, highlights that the misinformation is lowest for the RF based method with values ranging from 2.3 to 2.8, which is in line with the performance of the top 4 CAFA2 methods. The remaining uncertainty for the six method ranges from 8 to 21, whilst that of the CAFA2 submission ranges from 4 to 8.5. This highlights that the six classifiers are missing protein annotations.

The high values of remaining uncertainty warranted further investigation to determine the cause to such high discrepancy in the RUMI curve. For this purpose, the dataset for the CAFA3 targets and the true positive set generated on QuickGO 161 were analysed, as follows:

1. For each sequence, in the true positive set and in the CAFA3 prediction target dataset:

    a) For each annotation in the CAFA3 prediction target dataset, the enriched subgraph is extracted from the GO DAG version 2016-06-01. In case multiple annotations exist for the protein sequence, the subgraphs, are merged into one subgraph.

    b) For each annotation in the true positive set, the enriched subgraph is extracted from the GO DAG version 2016-06-01. In case multiple annotations exist for the protein sequence, the subgraphs, are merged into one subgraph.

    c) Subtract from the true positive set sub graph the CAFA3 prediction target subgraph. The resulting set of GO terms are missing from the CAFA3 prediction target dataset.

From the results reported in Table 5.4, 38.9% of the subgraphs for *Homo sapiens* are different, whilst for *E. coli* 29.5% of the subgraphs are different. The subgraph analysis reports the number of missing GO terms per taxon. These missing annotations originate from different levels within the GO DAG as per Figure 5.5.

The remaining uncertainty is high for both taxonomies as per Figure 5.2(a) for *Homo sapiens* and Figure 5.4(a) for *E. coli*. The fact that the evaluation dataset did not have

Taxon	Same subgraphs	Different subgraphs	Missing GO terms
Homo sapiens	642	409	1,946
E. coli	239	100	489

Table 5.4: During evaluation, two subgraphs are constructed, one for the test dataset and another of the true positive set. This tables analyses the subgraph differences per sequences and counts the GO terms that are missing in the testing subgraph. These terms are contributing to the high remaining uncertainty value in the RUMI curves.

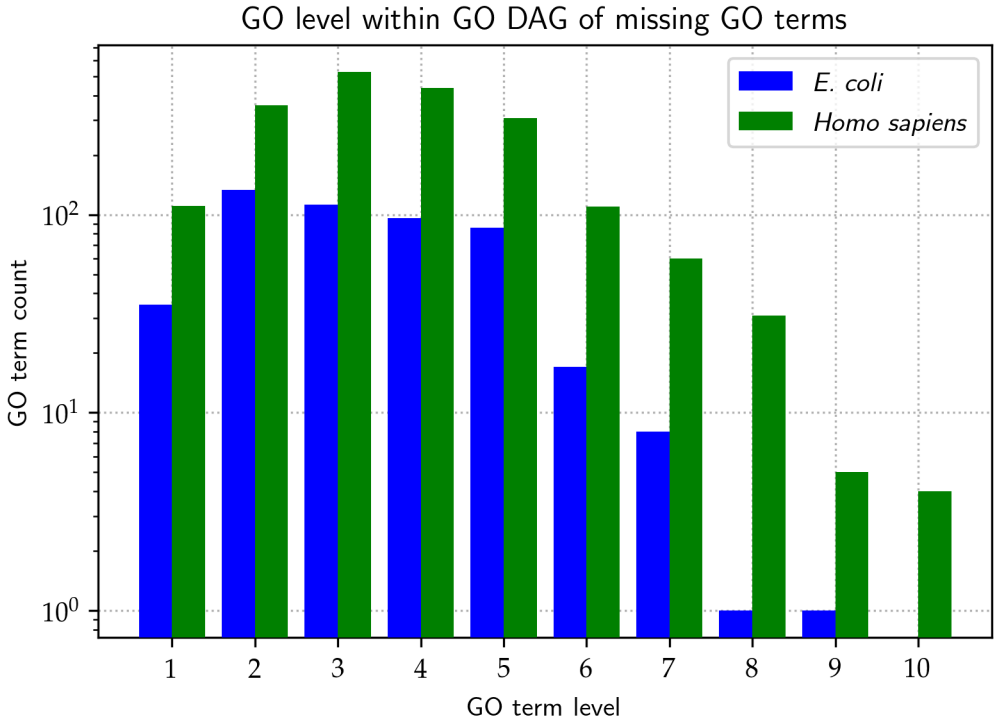


Figure 5.5: The GO subgraphs of the true positive set and evaluation datasets were generated and enriched. To identify missing GO term, the evaluation dataset was subtracted from the true positive set. The figure illustrates the GO term count and the GO level of terms in the GO DAG.

prediction data caused the predictor to skip annotations, which in turn increased the remaining uncertainty.

The evaluation of the models against CAFA2 submission show that the features selected by the GA have a strong correlation with the GO term. Prediction performance of *E. coli* and *Homo sapiens* species using species-specific classifiers achieved an F_{max} above the top 10 CAFA submissions for the species. An interesting aspect of the results is that the top *Homo sapiens* chromosome reported in Table 4.2 has eleven features of which eight originate from structural protein databases. The top *E. coli* GA chromosome reported in Table 4.3 has ten features, of which seven originate structural protein databases. From the results obtained, structural protein databases are effective is capturing protein information. The performance achieved by the *E. coli* classifier highlights that data for the Bacteria domain is more informative when compared to *Homo sapiens* because the best *E. coli* model improved F_{max} of the best CAFA2 submission by 0.16.

5.4.1 | Evaluation Limitation

The work was evaluated the CAFA3 targets against the CAFA2 submissions. This approach was utilised due to unavailability of the ML data for the CAFA2 targets and the CAFA3 results. The evaluation performed followed the CAFA evaluation protocol and provides an indicative benchmark for the work performed.

The CAFA evaluation assesses submissions using the annotations acquired during the annotation phase. The benchmark set used to evaluate the proteins might contain annotations that are described in the evaluation GO DAG but were missing in the GO DAG during the prediction phase. This can penalise submissions as the models are unable to predict a GO term that does not exist.

5.5 | Summary

This chapter presented the evaluation protocol used to evaluate the work performed. The protocol included information on training, testing and true positive datasets, evaluation procedure and presentation of the results. The results obtained were analysed and critically compared to CAFA2 submissions. The concluding chapter of this dissertation discusses the results in terms of the objectives set in the outset.

6

Conclusions

This chapter analyses the outcomes in view of the defined aims and objectives. The first section analyses each objective and how the work performed addresses the objectives and subsequently the aim of this dissertation. The second section criticises the approach used and highlights limitations of the work. The third section reports further improvement to the work performed. The fourth section gives a high level statement of the work performed highlighting the main contribution.

6.1 | Achieved Aims and Objectives

This study investigated the application of ML techniques to improve protein function prediction. The work carried out in this study will be analysed to identify the contributions.

The research uses automatic feature selection on the training dataset to identify strongly correlated features with functionality labels (GO terms). Automatic feature selection was applied on two species-specific datasets, namely *Homo sapiens* and *E. coli*. Automatic feature selection evaluated the selected feature sets using a fitness function. In turn the fitness function used two ML techniques to evaluate performance of the selected features. The first ML technique used was RF that was executed with two different node splitting criterion, namely Entropy and Gini. The second ML technique used was SVM that was executed with two penalty values (C) 0.1 and 1.0. The four runs were executed for each species-specific dataset.

The top performing ML technique for both taxonomies was RF with Gini as node splitting criteria (RF-Gini). The features selected for the two species are different. Analysis of the top six feature sets selected by the GA show common features exist amongst the two species. These common features can be used as the basis on PFP classifiers for

other species. The domain is a biological classification level that classifies organisms based on cell structure. The interesting aspect of the identified feature set is that the two species tackled originate from two different domains, Eukaryote for *Homo sapiens* taxon and Bacteria for *E. coli* taxon.

The feature selection process was subjected to cross validation to determine the sensitivity of feature selection process to dataset changes. The experiment entailed splitting the training dataset into training and validation datasets using k-fold cross validation with k=5. For each fold, the automatic feature selection was executed and the F_{max} of the fold was determined. The cross validation F_{max} was the mean of the F_{max} of each fold. The experiment was executed for the two species specific datasets. For each dataset the following four ML runs were executed: RF using Entropy and Gini as node splitting criterion and SVM with to penalty values (C) 0.1 and 1.0. The experiment results showed that the *Homo sapiens* runs had less sensitivity to the dataset changes when compared to *E. coli*. The results of the automatic feature selection were compared to the cross validation runs, to determine if the value obtained was within the cross validation values. The *E. coli* automatic feature selection was within the range cross validation runs for both the RF and SVM runs. In the case of *Homo sapiens*, the RF cross validation runs performed slightly better results when compared to the automatic feature selection performance. Analysis of the features evaluated on both runs showed that the variations in the datasets cause different feature sets to the evaluated.

The training data size experiment was executed on *Homo sapiens* and *E. coli* using the features selected by automatic feature selection. For both species the best ML technique was RF using Gini as node splitting criterion. This experiment varied the size of the training dataset and kept the testing dataset constant. The training dataset started from five hundred annotations and increased in increments of five hundred. The experiments of the training dataset size showed that both classifiers exhibit similar performance with the same amount of training data. The *Homo sapiens* has superior performance as more data was available. For both classifiers the performance curve did not plateau with additional training data, thus are most likely to benefit with more training data.

The applicability of species specific ML models across different species was experimented. The experiment entailed training a species-specific model and use it to perform predictions of other species. This experiment was executed for the two species, namely *Homo sapiens* and *E. coli*. The outcome of the experiment confirmed the intuition that *Homo sapiens* ML model performed better on Eukaryote domain, whilst *E. coli* ML model perform better on Bacteria domain. The classifier's performance decreases as the evolutionary distance between species increases.

The ML models were evaluated on predictions of the CAFA3 targets against the

CAFA2 submissions. The performance of the methods varied per ML technique, the RF has the best performance followed by NN and SVM for both species. In the case of *Homo sapiens*, the RF classifier ranked at par with the best submission, whilst the *E. coli* classifier, out-performed the best CAFA2 submission. The results highlight the richness of features originating from structural protein databases.

The objectives defined in the outset were thoroughly investigated and fulfilled for both *Homo sapiens* and *E. coli* species, achieving the pre-defined aim of improving protein function prediction through ML techniques. The results obtained in the experimentation phase, confirmed that applying the RF technique on the provided datasets for the *Homo sapiens* and *E. coli* species achieved reliable PFP at par and better than CAFA2 submissions respectively.

6.2 | Critique and Limitations

The work in this dissertation was evaluated, with limitations, against CAFA shared task. Evaluation against CAFA2 shared task was not possible as the ML data for the CAFA2 targets was unavailable. In case of CAFA3 evaluation, the ML data for CAFA3 targets was made available by Dr Lees a member of Orengo Group at University College London (UCL). However, the CAFA3 results paper was not available and will be available in preprint format within months as per communication in Appendix E. In view of these constraints, the results reported for CAFA3 targets were paper evaluated against CAFA2 shared task results.

The dataset provided by Dr Lees, consists in protein identifier, GO terms and a set of features describing the protein. Proteins are associated with a number of GO terms that describe the protein functionality. Some of the features in the dataset are linked to the protein sequence whilst others are linked to the domains. The dataset includes protein identifier but lacks CATH and PFAM domain identifiers. The missing domain information hindered the inclusion of other sources such as PPI and CATH FunFam.

In order to perform PFP for a new protein, the dataset features for the new protein must be computed. Assuming that the protein is originating from *Homo sapiens* or *E. coli*, the feature set to use in ML has already been identified by automatic feature selection. However, a limitation of the current system is that it lacks the pre-processing pipeline necessary to create the features from the protein sequence. Due to this limitation the system is unable to perform PFP to an unknown protein. This issue can be tackled by creating the necessary infrastructure to generate the datasets for this work.

A potential issue in handling new proteins is that the dataset contains features com-

puted on protein families with PFAM and CATH. The family membership can be identified using a PFAM scan available from PFAM FTP site[1] and CATH Genomescan tool available from UCLOrengoGroup Github[2]. Both tools utilise HMM profiles to identify potential family membership using protein sequences. Even though the coverage of the protein databases is growing, there exists the chance that matches are not found inhibiting the feature computation.

In this work, the training data was used to train ML models to perform PFP. The performance of the ML models is dependent on the data used for training. The coverage of the ML classifier is determined by the distinct labels in the training data. This implies that the ML predictor would be unable to predict GO terms that are not present in the training dataset. Thus, the proposed solution can be used to predict GO terms from within the training dataset coverage. The ability to predict functionality that is yet to be discovered is further hindered by the inclusion of the GO term in the GO DAG.

This work builds a species-specific predictor to perform PFP. From the experiments performed it was shown that sufficient data for the given taxonomy is required to perform reliable PFP. In the case of *Homo sapiens* even through the classier performance did not plateau as the training data was increased, the classifier performed at par on the F_{max} with the top CAFA2 submissions. On the other hand, in case of *E. coli*, experiments showed that classifier may benefit from additional data. The experiment of cross species models showed that the model works well as long as the species are evolutionary close. Although the model can be utilised to perform cross species predictions, the work does not perform checks on the applicability of the GO term to the target species. A consequence of this is that it may predict invalid GO terms for the target species.

6.3 | Future Work

Part of this work executed feature selection using a fitness function that evaluates the GA chromosome fitness using RF and SVM ML techniques. For each ML technique two parameters were experimented to achieve better predictor performance. In case of SVM more work is required to experiment with different penalty values. Within this dissertation, limited hyperparameter experimentation was performed due to time and resource constraints.

This work is based on the dataset provided by Dr Jon Lees. This dataset enabled this research to get to speed and focus on the implementation of the necessary infrastructure.

[1] `ftp://ftp.ebi.ac.uk/pub/databases/Pfam/Tools/` (Accessed 2018-12-30)
[2] `https://github.com/UCLOrengoGroup/cath-tools-genomescan` (Accessed 2018-12-30)

Given enough time and resources, it would have been beneficial to build the dataset as part of this work. The main rationale being that more data could have been used and thus enabling this research to tackle different species and run evaluation against all the annotated CAFA targets.

The system developed operates in batch mode, whereby the ML models are trained and used to generate predictions. In order to improve the accessibility of the system, further work is required to change the operating mode to be web enabled. Through the online web portal, the user can perform PFP for *Homo sapiens* and *E. coli* species from the CAFA3 targets. The portal will utilise pre-trained models and generate predictions. The results can be presented to the user as: a visualisation, a table or in CAFA format. To facilitate environment deployment, the prediction environment should be made available as a Docker[3] image. The docker image should be made public to enable researchers to download to the image and make predictions offline. To public web portal can be offered by deploying the docker image on a public hosting service such as Amazon Elastic Container Service[4].

Automatic feature selection was used in this work. This area can be explored further by using different feature selection approaches such as recursive feature elimination. Exploring multiple feature selection methods enables comparison of feature-sets and PFP method performance.

This work used the datasets provided by Dr Lees from UCL. The datasets contain features computed from protein sequence and proportions from protein databases. Another approach to PFP is to use a protein's sequence. Work can be tackled by converting protein sequences into vectors using Prot2Vec (Asgari and Mofrad, 2015) and linked to GO term. The labelled dataset can be used to train an LSTM network and subsequently, the trained LSTM can be used to perform PFP.

The concept of splitting a protein's sequence to k-mers has been applied on protein to tackle secondary structure prediction (Madera et al., 2010). Within DL, the protein sequences can also be represented as k-mers associated with GO terms. The k-mers can be used to train LSTM to perform PFP. This approach would require exploring different k-mer sizes to identify the best performing k-mer size.

[3] `https://www.docker.com/` (Accessed 2018-12-30)
[4] `https://aws.amazon.com/ecs/` (Accessed 2018-12-30)

6.4 | Final Words

This dissertation investigated the application of machine learning techniques to achieve reliable PFP. The main contribution of this work is that it identified species specific feature sets and the ML technique to enable reliable PFP. The feature sets comprise of proportions from structural protein databases and protein measurement extracted from the protein sequence using different tools. The identified feature sets and methods were limitedly evaluated against CAFA2 share task methods and performance exceeded the top performing method.

A

Media Contents

The experiment setup used in this dissertation comprised different artefacts, source code, ML datasets, configuration files and additional data. Figure A.1, details the directory structure used for the different artefacts in the attached media.

```
/
├── dataset (datasets used in this work)
│   ├── generated datasets (datasets used in experiments)
│   └── provided dataset (provided dataset files)
├── experimentevaldata (dump of experiment data)
├── mariadbdata (dump of downloaded/processed data)
├── otherdata (provided information content data and GO DAG OBO files)
└── src (sources directory)
    ├── bioml (ML wrapper code)
    ├── dsgeneratorcafa2 (dataset generator code)
    ├── dsgeneratorcafa3 (dataset generator code)
    ├── evalmetrics (CAFA evaluation code)
    ├── experimentconfigs (experiment configuration files)
    ├── ga (GA code)
    ├── misc (various utility code and figuregenerator)
    ├── nn (NN wrapper code)
    └── pipeline (integration code)
```

Figure A.1: Media contents directory structure.

The Python source of the different modules used in this dissertation are provided under *src* directory. Each Python module is located in a different directory.

This work relies on a number of data artefacts to generate the ML datasets. The *dataset* contains the datasets provided by Dr Lees and the generated ML datasets.

The *otherdata* directory contains the three files. Two versions of the GO DAG in OBO file format and the information content data in Python pickle format.

The data used in this dissertation is stored in MariaDB tables to facilitate storage and retrieval. The compressed MariaDB dump located under *mariadbdata* generated using *mysqldump* contains the following tables:

- *cafa2quickgoannotations124* - populated QuickGo data using GOA 124 data.
- *cafa3quickgoannotations2016* - populated QuickGo data using GOA 2016 data.
- *cafa3quickgoannotations161* - populated QuickGo data using GOA 161 data.
- *cafa3completemlds* - Dr Lees CAFA3 dataset populated from HDF files.
- *taxonid* - Taxon ID to string mapping populated from NCBI TaxID.
- *cafa3targetsfasta* - CAFA3 target information loaded from CAFA3 FASTA files.
- *cafa3mappings* - gene name to UniProt ID mapping data populated using Uniprot mapping service.

The experiment performance data is stored MariaDB tables. Under directory *experimentevaldata* four dump files were generating using *mysqldump* each one containing the following tables:

- *evaluationruns* - run information and the computed metrics.
- *evaluationdatafmax* - run information and precision recall data to plot curve.
- *evaluationdatarumi* - run information and remaining uncertainty misinformation data to plot curve.
- *garuns* - details of the GA execution.
- *gaperformance* - chromosome scoring linked to specific GA execution.
- *gageneration* - tracks the epochs of the GA runs.
- *gacvdesc* - tracks cross validation GA runs.

The directory *experimentevaldata* also contains two *SQLite* databases. The files contain the prediction results generated by the RF and SVM models for the CAFA3 targets. The database file *nnperf_cafa3.sqlite* contains the prediction results for the NN models store in tables prefixed with *nn*. The tables in the databases are:

- *runs* - run information and the computed metrics.
- *runoutput* - the predicted GO terms and prediction confidence for the CAFA3 targets.

B

Dataset Overview

The dataset provided training dataset includes a number of features. All the features describe GO terms originating from the "molecular function" ontology of the GO DAG. Table B.1 describes the dataset in terms of its contents, the minimum, maximum and number of unique values for each feature.

Column name	Minimum value	Maximum value	Unique values
GO	GO:0000006	GO:2001227	6,388
all_go_prop_cath_funfam	0.0	1.0	4,077
all_go_prop_dc	0.0	1.0	6,028
all_go_prop_pfam	0.0	1.0	11,899
all_go_prop_pfam_funfam	0.0	1.0	4,626
ancestor_Bilateria	0	1	2
ancestor_Eukaryote	0	1	2
ancestor_gn	0	1	2
cider_FCR	0.0	0.68	17,520
cider_hydropathy	0.0	6.389655172413794	30,992
cider_kappa	0.0	0.6465477004447107	30,341
disorder	0.0	1.0	5,441
go_freq_score	0.0	165.89	398
go_level	0.1	1.1	11
go_with_domain	0	1	2
go_with_mmseq	0	1	2
mmseq_bit_score	0.0	126.26	3,294
mmseq_ident	0.0	1.0	887
mmseq_prox	0.0	10.0	2,153
number_pfam_domains	0.0	2.70805020110221	13
pfam_to_go	0	1	2
seq_acc	A0A087X1C5	X2JGQ0	31,097
signalp	0	1	2
strict_go_prop_cath_funfam	0.0	1.0	1,020
strict_go_prop_dc	0.0	1.0	1,393
strict_go_prop_pfam	0.0	1.0	2,990
strict_go_prop_pfam_funfam	0.0	1.0	997
tmhmm	0	34	26
yval	-1.0	1.0	2

Table B.1: Training dataset description.

C

Dissertation Journey

This dissertation was completed in part time commitment over eighteen months.

Jun 2017 - Dataset exploration of the dataset using RapidMiner.

Jul 2017 - Started code for GO term visualiser based on D3 Javascript library. Loaded GO DAG in Neo4J graph database to familiarise with structure using Cypher queries.

Aug 2017 - Rewrote ML code using Scikit-learn. Researched visualisation library and evaluated Cytoscape. Cytoscape provide various layout plug-ins such as Dagre. Re-implemented GO term visualiser based on Node JS and Cytoscape. Continued work to understand the dataset provided by Dr Lees at UCL. Switched performance metric from accuracy to F_{max}. Generated visualisations for results obtained using Matplotlib.

Sep 2017 - Continued dataset analysis in coordination with Dr Lees at UCL. Added prediction confidence to ML results. Ongoing work to improve specificity of the work and improve feature selection. Initial work on implementing the information theoretic metrics. Started work to generate visual map of the GO ontology.

Oct 2017 - Updated GO term visualiser to be based on Bootstrap, retrieve data through REST endpoints and to support mouseover events. Improved dataset generation to ensure that the proteins are not split between training and testing datasets. Started work to develop pipeline that orchestrates the different modules to perform experiments, prototyped idea with evaluation code integration. Started work on progress report.

Nov 2017 - Finalise Progress report. Started developed on feature selection code based on literature read. Investigated issues related to high annotation count of proteins dataset.

Dec 2017 - Ongoing dataset analysis related to high annotation count for proteins. Worked with multiple databases to verify protein annotations. Continued work on in-

formation theoretic implementation based on journal paper. Worked on CAFA2 Matlab implementation to prepare groundwork to test evaluation code implementation.

Jan 2018 - Continued work on the information theoretic implementation to ensure correct results. Worked on GA to ensure proper integration with evaluation code. Code was implemented as python modules and fitness function was externalised to other module to keep GA independent from chromosome evaluation. Re-wrote critical parts of the code to improve performance. Fixed MariaDB connection timeout issues.

Feb 2018 - Tackled pipeline performance issues through parallelisation, data structure optimisation, re-implementation of critical code and used Cython to speed up code execution. Analysis of the data resulted that in most cases GO term is already enriched causing classifier to predict shallow terms in GO DAG. Developed code to identify leaf node and remove intermediate annotations. Wrote updated abstract for external examiner. Due to family circumstances work on the dissertation was paused.

Mar 2018 - Investigated different ML techniques including Deep Learning. Discussed various options to enrich the dataset, including function families data. Work was carried out to get sequence data and extract FunFam features using appropriate tools. Data enrichment options were considered including PFAM, CATH FunFam and PPI data.

Apr 2018 - Executed GA and extracted "best chromosome". Test were performed using different ML techniques, such as SVM, RF and Logistic Regression. Performed cross species testing as a control experiment. Work on figure generation was started to automate figure generation and ensure consistency. Organised Skype call with Prof. Orengo UCL to tackle dataset issues. Dr Lees UCL proposed to use CAFA3 dataset as more data is available. True annotations of the sequence could be downloaded from QuickGO. CAFA3 was made available and code updated to reflect data changes. Initial work focused on *Homo sapiens* and *E. coli*.

May 2018 - Mapped CAFA3 target to UniProt ID. Initial attempts did not work as web service does not proved ranked search results. QuickGO data of all sequences were downloaded from QuickGO WS. Taxonomy data was downloaded from NCBI and leaf nodes annotation were marked to facilitate ML dataset generation. Initial investigation of NN based on Scikit-learn was carried out. Worked to test out SVM, one vs one, one vs rest and SVM ensemble. SVM ensemble performed poorly. Code was amended to test SVM kernel estimators, however this was not used due to time constraints.

Jun 2018 - Continued work on NN, implementation options investigated were MLP-Classifer and Pytorch. Python code was developed to implement the NN experimental code using Pytorch with ADAM optimiser and mini batches. Started work on to investigate NN stopping criteria.

Jul 2018 - Implemented NN training stopping criteria based on error reduction. Started work on the dissertation report and executed experiments.

Aug 2018 - Ongoing work on the dissertation report.

Sep 2018 - Work continued on ML tuning and cross validation exploration. A number of experiments were executed. Ongoing work on the dissertation report.

Oct 2018 - During some experiments a data anomaly was identified. Further investigation revealed that during dataset creation annotations were retrieved from a more recent version of QuickGO. In some cases the GO terms used in QuickGO were not found in the GO DAG. Issue was solved by swapping QuickGO WS with GOA DS. The dataset generation was modified to read older QuickGO version. This change required re-execution to all the experiments. Implemented GA cross validation and started experiment execution.

Nov 2018 - Re-executed all the experiments. The GA cross validation code required some minor fixes in the database connection logic to fix race conditions. Ongoing work on the dissertation report and figure fixes to improve clarity.

Dec 2018 - Whilst discussing the results it was noted that the training dataset used was not the right one. For this purpose, the proper training dataset was generated and all the experiments re-executed. Ongoing work on the dissertation report and figure fixes to improve clarity.

D

Additional Experiment Results

The GA was executed on two species specific datasets, namely *Homo sapiens* and *E. coli* using different ML techniques to evaluate GA chromosomes. This appendix reports the GA chromosomes for:

- RF with Entropy as the node splitting criterion, denoted as RF-Entropy.
- SVM with the misclassification penalty value set to 1.0, denoted as SVM-C1.0.
- SVM with the misclassification penalty value set to 0.1, denoted as SVM-C0.1.

The top 6 GA chromosomes for RF-Entropy are reported for *E. coli* in Table D.1 and for *Homo sapiens* in Table D.2. Analysis of the top 6 GA chromosomes, illustrates that using RF-Entropy as ML technique, the GA selects more the ratios of protein databases than protein measurements. The two protein measurements that were selected are: *tmhmm* (number of predicted transmembrane helices) and protein disorder information (prefixed *cider*.

The GA chromosomes selected using SVM-C0.1 vary significantly between taxa. For the *E. coli* as per Table D.3, the database ratios with strict evidence codes and protein disorder information were selected. In case of *Homo sapiens* as illustrated in Table D.4, chromosomes include more ratios considering all evidence codes and protein structural information such as protein disorder information, *tmhmm* and *signalp* (presence of signal peptide cleavage sites).

The feature set selected by the GA using SVM-C1.0 vary between taxa. In case of *E. coli* as illustrated in Table D.5, the strict database ratios, protein disorder information and *tmhmm* were selected. For *Homo sapiens* taxon as illustrated in Table D.6, the GA selected, protein measurements derived from sequences such as disorder information, sequence match information (prefixed *mmseq*) and *tmhmm*.

Chromosome	all_go_prop_cath_funfam	all_go_prop_dc	all_go_prop_pfam	all_go_prop_pfam_funfam	cider_FCR	cider_hydropathy	cider_kappa	disorder	mmseq_bit_score	mmseq_ident	mmseq_prox	number_pfam_domains	pfam_to_go	signalp	strict_go_prop_cath_funfam	strict_go_prop_dc	strict_go_prop_pfam	strict_go_prop_pfam_funfam	tmhmm
1		✓	✓	✓						✓		✓	✓		✓	✓	✓	✓	
2		✓											✓		✓				✓
3		✓										✓	✓			✓			
4	✓	✓			✓	✓			✓		✓				✓	✓			✓
5		✓													✓				✓
6		✓													✓	✓			

Table D.1: Top 6 GA chromosomes for *E. coli* taxon using RF-Entropy as the ML technique.

Chromosome	all_go_prop_cath_funfam	all_go_prop_dc	all_go_prop_pfam	all_go_prop_pfam_funfam	cider_FCR	cider_hydropathy	cider_kappa	disorder	mmseq_bit_score	mmseq_ident	mmseq_prox	number_pfam_domains	pfam_to_go	signalp	strict_go_prop_cath_funfam	strict_go_prop_dc	strict_go_prop_pfam	strict_go_prop_pfam_funfam	tmhmm
1		✓	✓	✓				✓			✓	✓		✓	✓	✓			✓
2		✓	✓					✓			✓		✓		✓	✓	✓	✓	✓
3		✓	✓	✓	✓					✓	✓		✓		✓	✓	✓	✓	✓
4		✓	✓	✓	✓				✓		✓				✓	✓			✓
5	✓	✓	✓									✓			✓	✓	✓	✓	✓
6	✓	✓	✓								✓				✓	✓	✓	✓	✓

Table D.2: Top 6 GA chromosomes for *Homo sapiens* taxon using RF-Entropy as the ML technique.

Chromosome	all_go_prop_cath_funfam	all_go_prop_dc	all_go_prop_pfam	all_go_prop_pfam_funfam	cider_FCR	cider_hydropathy	cider_kappa	disorder	mmseq_bit_score	mmseq_ident	mmseq_prox	number_pfam_domains	pfam_to_go	signalp	strict_go_prop_cath_funfam	strict_go_prop_dc	strict_go_prop_pfam	strict_go_prop_pfam_funfam	tmhmm
1		✓		✓		✓	✓				✓	✓	✓			✓	✓		
2				✓	✓		✓		✓		✓	✓			✓	✓			✓
3		✓	✓	✓		✓		✓			✓	✓	✓			✓	✓		
4		✓	✓			✓	✓		✓			✓	✓		✓	✓	✓	✓	✓
5				✓		✓					✓	✓			✓	✓	✓		
6	✓			✓	✓	✓			✓	✓	✓	✓		✓	✓	✓			✓

Table D.3: Top 6 GA chromosomes for *E. coli* taxon using SVM-C0.1 as the ML technique.

Chromosome	all_go_prop_cath_funfam	all_go_prop_dc	all_go_prop_pfam	all_go_prop_pfam_funfam	cider_FCR	cider_hydropathy	cider_kappa	disorder	mmseq_bit_score	mmseq_ident	mmseq_prox	number_pfam_domains	pfam_to_go	signalp	strict_go_prop_cath_funfam	strict_go_prop_dc	strict_go_prop_pfam	strict_go_prop_pfam_funfam	tmhmm
1	✓	✓		✓	✓	✓	✓	✓			✓	✓	✓	✓			✓		✓
2	✓	✓		✓	✓	✓		✓	✓		✓	✓	✓	✓			✓		✓
3	✓		✓		✓		✓	✓			✓	✓	✓	✓			✓		✓
4	✓	✓		✓	✓			✓	✓		✓	✓	✓	✓			✓	✓	✓
5	✓	✓		✓	✓	✓	✓	✓				✓	✓	✓	✓		✓	✓	✓
6	✓			✓	✓	✓	✓			✓	✓	✓	✓	✓			✓		✓

Table D.4: Top 6 GA chromosomes for *Homo sapiens* taxon using SVM-C0.1 as the ML technique.

Chromosome	all_go_prop_cath_funfam	all_go_prop_dc	all_go_prop_pfam	all_go_prop_pfam_funfam	cider_FCR	cider_hydropathy	cider_kappa	disorder	mmseq_bit_score	mmseq_ident	mmseq_prox	number_pfam_domains	pfam_to_go	signalp	strict_go_prop_cath_funfam	strict_go_prop_dc	strict_go_prop_pfam	strict_go_prop_pfam_funfam	tmhmm
1	✓				✓							✓	✓		✓	✓	✓	✓	
2	✓		✓	✓	✓	✓			✓	✓	✓	✓	✓	✓	✓	✓			✓
3		✓	✓	✓	✓	✓	✓					✓	✓		✓	✓	✓	✓	✓
4		✓		✓		✓	✓	✓			✓	✓			✓		✓		
5	✓				✓	✓	✓					✓	✓		✓	✓	✓	✓	
6		✓	✓	✓	✓	✓	✓	✓			✓	✓	✓		✓		✓	✓	

Table D.5: Top 6 GA chromosomes for *E. coli* taxon using SVM-C1.0 as the ML technique.

Chromosome	all_go_prop_cath_funfam	all_go_prop_dc	all_go_prop_pfam	all_go_prop_pfam_funfam	cider_FCR	cider_hydropathy	cider_kappa	disorder	mmseq_bit_score	mmseq_ident	mmseq_prox	number_pfam_domains	pfam_to_go	signalp	strict_go_prop_cath_funfam	strict_go_prop_dc	strict_go_prop_pfam	strict_go_prop_pfam_funfam	tmhmm
1	✓	✓		✓	✓	✓	✓	✓		✓	✓	✓	✓			✓			✓
2	✓		✓	✓		✓	✓		✓		✓	✓	✓	✓		✓			✓
3	✓		✓	✓			✓	✓	✓		✓	✓	✓	✓		✓			✓
4		✓			✓	✓		✓		✓	✓	✓	✓	✓		✓			✓
5	✓		✓	✓			✓	✓	✓		✓	✓	✓			✓	✓		✓
6	✓	✓			✓		✓			✓	✓	✓	✓	✓		✓			✓

Table D.6: Top 6 GA chromosomes for *Homo sapiens* taxon using SVM-C1.0 as the ML technique.

E

CAFA3 Results Enquiry

This appendix reports email communication with Prof Iddo Friedberg regarding the CAFA3 results. From the email communication the CAFA3 evaluation paper is works in progress and it will take at least a couple of months for a preprint version of the paper to be available.

L-Università ta' Malta

Kenneth Penza <kenneth.penza.16@um.edu.mt>

Re: New Message From Bio Function Prediction - Contact: Iddo Friedberg

Friedberg, Iddo [V MPM] <idoerg@iastate.edu> 18 December 2018 at 18:50
To: Kenneth Penza <kenneth.penza.16@um.edu.mt>

Hi Kenneth,

We do not have a CAFA3 paper yet. It will take a couple of months before we even have a preprint. Would evaluating yourself against CAFA2 methods work?

Iddo

On Tue, Dec 18, 2018 at 1:04 AM Kenneth Penza <mail@biofunctionprediction.org> wrote:

> Good morning, First of all, hope this email finds you well. I am Kenneth Penza a M. Sc. Artificial Intelligence student at the University of Malta. In my dissertation I am applying different machine learning techniques on bioinformatics data. My work, will be evaluated against CAFA submissions as it this provides the robustness of evaluating against a solid benchmark. From the site (https://biofunctionprediction.org/), I cannot find the CAFA3 results paper. Is there a version of the paper I can use to perform the paper evaluation? Thank you for your time and consideration Kenneth

--
Iddo Friedberg, PhD
Associate Professor, Computational Biology
Associate Chair, Graduate Program in Bioinformatics and Computational Biology
Dpt. of Veterinary Microbiology and Preventive Medicine
2118 Vet Med
1800 Christensen Dr.
Iowa State University
Ames, IA 50011-1134
USA
+1 (515) 294 5959
http://iddo-friedberg.net/contact.html
++++++++++[>+++>++++++>++++++++>++++++++++>++++++++++<<<<<-]>>>>++++.>
++++++.----.<<<<+++++++++++++++++++++++++++++.----------.>>>+.-----.
.>-.<<<<---.>>>++.>+++.<+++.----.-.<++++++++++++++++++.>+.>.<++.<<<+.>>
>>----.<---.>++++++.<<<<----------.

References

Stephen F. Altschul, Warren Gish, Webb Miller, Eugene W. Myers, and David J. Lipman. Basic local alignment search tool. *Journal of Molecular Biology*, 215(3):403–410, 1990.

Ehsaneddin Asgari and Mohammad R. K. Mofrad. Continuous Distributed Representation of Biological Sequences for Deep Proteomics and Genomics. *PLOS ONE*, 10(11):1–15, 11 2015.

Michael Ashburner, Catherine A. Ball, Judith A. Blake, David Botstein, Heather Butler, J. Michael Cherry, Allan P. Davis, Kara Dolinski, Selina S. Dwight, Janan T. Eppig, et al. Gene Ontology: tool for the unification of biology. *Nature genetics*, 25(1):25, 2000.

Lisa Bartoli, Ludovica Montanucci, Raffaele Fronza, Pier Luigi Martelli, Piero Fariselli, Luciana Carota, Giacinto Donvito, Giorgio P. Maggi, and Rita Casadio. The Bologna Annotation Resource: a Non Hierarchical Method for the Functional and Structural Annotation of Protein Sequences Relying on a Comparative Large-Scale Genome Analysis. *Journal of Proteome Research*, 8(9):4362–4371, 2009.

Alex Bateman, Ewan Birney, Richard Durbin, Sean R. Eddy, Robert D. Finn, and Erik L. L. Sonnhammer. Pfam 3.1: 1313 multiple alignments and profile HMMs match the majority of proteins. *Nucleic acids research*, 27(1):260–262, 1999.

Alex Bateman, Ewan Birney, Richard Durbin, Sean R. Eddy, Kevin L. Howe, and Erik L. L. Sonnhammer. The Pfam protein families database. *Nucleic Acids Research*, 28(1):263–266, 2000.

Andreas D. Baxevanis and B. F. Francis Ouellette. *Bioinformatics: A Practical Guide to the Analysis of Genes and Proteins*, volume 43. John Wiley & Sons, second edition, 2004. ISBN 0471383910.

Seth I. Berger and Ravi Iyengar. Network analyses in systems pharmacology. *Bioinformatics*, 25(19):2466–2472, 2009.

Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Springer-Verlag New York, 2009. ISBN 9780387310732.

Verónica Bolón-Canedo, Noelia Sánchez-Maroño, and Amparo Alonso-Betanzos. *Feature Selection for High-Dimensional Data*. Artificial Intelligence: Foundations, Theory, and Algorithms. Springer International Publishing, 2015. ISBN 978-3-319-21858-8.

Boyan Bonev. *Feature selection based on information theory*. PhD thesis, University of Alicante, 2010.

Leo Breiman. Bagging predictors. *Machine Learning*, 24(2):123–140, 1996.

Leo Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001.

Leo Breiman, Jerome H. Friedman, Richard A. Olshen, and Charles J. Stone. *Classification and regression trees.* Chapman and Hall, 1984. ISBN 9780412048418.

Ian Bruno, Saulius Gražulis, John R. Helliwell, Soorya N. Kabekkodu, Brian McMahon, and John Westbrook. Crystallography and Databases. *Data Science Journal*, 16(38):1–17, 2017.

Christiam Camacho, George Coulouris, Vahram Avagyan, Ning Ma, Jason Papadopoulos, Kevin Bealer, and Thomas L. Madden. BLAST+: architecture and applications. *BMC bioinformatics*, 10(1):421, 2009.

Mengfei Cao, Hao Zhang, Jisoo Park, Noah M. Daniels, Mark E. Crovella, Lenore J. Cowen, and Benjamin Hescott. Going the Distance for Protein Function Prediction: A New Distance Metric for Protein Interaction Networks. *PLOS ONE*, 8:1–12, 10 2013.

Renzhi Cao and Jianlin Cheng. Integrated protein function prediction by mining function associations, sequences, and protein–protein and gene–gene interaction networks. *Methods*, 93:84–91, 2016.

Renzhi Cao, Colton Freitas, Leong Chan, Miao Sun, Haiqing Jiang, and Zhangxin Chen. ProLanGO: protein function prediction using neural machine translation based on a recurrent neural network. *Molecules*, 22 (10):1732, 2017.

Marcus C. Chibucos, Christopher J. Mungall, Rama Balakrishnan, Karen R. Christie, Rachael P. Huntley, Owen White, Judith A. Blake, Suzanna E. Lewis, and Michelle Giglio. Standardized description of scientific evidence using the Evidence Ontology (ECO). *Database*, 2014:bau075, 2014.

Davide Chicco, Peter Sadowski, and Pierre Baldi. Deep Autoencoder Neural Networks for Gene Ontology Annotation Predictions. In *Proceedings of the 5th ACM Conference on Bioinformatics, Computational Biology, and Health Informatics*, pages 533–540. ACM, 2014.

Wyatt T. Clark and Predrag Radivojac. Analysis of protein function and its prediction from amino acid sequence. *Proteins: Structure, Function, and Bioinformatics*, 79(7):2086–2096, 2011.

Wyatt T. Clark and Predrag Radivojac. Information-theoretic evaluation of predicted ontological annotations. *Bioinformatics*, 29(13):i53–i61, 2013.

Francesc Comellas and Javier Ozón. Graph Coloring Algorithms for Assignment Problems in Radio Networks. In *Proceedings of the International Workshop on Applications of Neural Networks to Telecommunications2*, pages 46–56. Psychology Press, 1995. ISBN 0-8058-2084-1.

Gene Ontology Consortium. The Gene Ontology (GO) database and informatics resource. *Nucleic Acids Research*, 32(suppl_1):D258–D261, 2004.

Gene Ontology Consortium et al. Creating the gene ontology resource: design and implementation. *Genome research*, 11(8):1425–1433, 2001.

Gene Ontology Consortium et al. The Gene Ontology in 2010: extensions and refinements. *Nucleic acids research*, 38(suppl_1):D331–D335, 2010.

Domenico Cozzetto, Federico Minneci, Hannah Currant, and David T. Jones. FFPred 3: feature-based function prediction for all Gene Ontology domains. *Scientific Reports*, 6:31865, 2016.

Sayoni Das, Natalie L. Dawson, and Christine A. Orengo. Diversity in protein domain superfamilies. *Current opinion in genetics & development*, 35:40–49, 2015a.

Sayoni Das, David Lee, Ian Sillitoe, Natalie L. Dawson, Jonathan G. Lees, and Christine A. Orengo. Functional classification of CATH superfamilies: a domain-based approach for protein function annotation. *Bioinformatics*, 31(21):3460–3467, 2015b.

Minghua Deng, Kui Zhang, Shipra Mehta, Ting Chen, and Fengzhu Sun. Prediction of protein function using protein–protein interaction data. *Journal of computational biology*, 10(6):947–960, 2003.

Joel T. Dudley, Tarangini Deshpande, and Atul J. Butte. Exploiting drug–disease relationships for computational drug repositioning. *Briefings in Bioinformatics*, 12(4):303–311, 2011.

Robert D. Finn, Jaina Mistry, Benjamin Schuster-Böckler, Sam Griffiths-Jones, Volker Hollich, Timo Lassmann, Simon Moxon, Mhairi Marshall, Ajay Khanna, Richard Durbin, Sean R. Eddy, Erik L. L. Sonnhammer, and Alex Bateman. Pfam: clans, web tools and services. *Nucleic acids research*, 34(suppl_1):D247–D251, 2006.

Valerio Freschi. Protein function prediction from interaction networks using a random walk ranking algorithm. In *2007 IEEE 7th International Symposium on BioInformatics and BioEngineering*, pages 42–48. IEEE, Oct 2007.

Iddo Friedberg and Predrag Radivojac. Community-wide evaluation of computational function prediction. In *The Gene Ontology Handbook*, pages 133–146. Springer, 2017. ISBN 978-1-4939-3743-1.

Iddo Friedberg, Yuxiang Jiang, and Predrag Radivojac. Supplementary Data for CAFA2, Jan 2016. URL `https://figshare.com/articles/Supplementary_Data_for_CAFA2/2059944/1`.

Vladimir Gligorijević, Meet Barot, and Richard Bonneau. deepNF: deep network fusion for protein function prediction. *Bioinformatics*, 34(22):3873–3881, 2018.

Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`.

Marianne A. Grant. Integrating computational protein function prediction into drug discovery initiatives. *Drug development research*, 72(1):4–16, 2011.

Yuanfang Guan, Chad L. Myers, David C. Hess, Zafer Barutcuoglu, and Olga G. Caudy, Amy A.and Troyanskaya. Predicting gene function in a hierarchical context with an ensemble of classifiers. *Genome Biology*, 9(1):S3, Jun 2008.

Tobias Hamp, Rebecca Kassner, Stefan Seemayer, Esmeralda Vicedo, Christian Schaefer, Dominik Achten, Florian Auer, Ariane Boehm, Tatjana Braun, Maximilian Hecht, et al. Homology-based inference sets the bar high for protein function prediction. *BMC bioinformatics*, 14(suppl_3):S7, 2013.

Trevor Hastie. *The elements of statistical learning: data mining, inference, and prediction*. Springer series in statistics. Springer, second edition, 2009. ISBN 9780387848570.

Randy L. Haupt and Sue Ellen Haupt. *Practical genetic algorithms*, volume 2. John Wiley & Sons, second edition, 2004.

Troy Hawkins, Meghana Chitale, Stanislav Luban, and Daisuke Kihara. PFP: Automated prediction of gene ontology functional annotations with confidence scores using protein sequence data. *Proteins: Structure, Function, and Bioinformatics*, 74(3):566–582, 2009.

Haretsugu Hishigaki, Kenta Nakai, Toshihide Ono, Akira Tanigami, and Toshihisa Takagi. Assessment of prediction accuracy of protein function from protein–protein interaction data. *Yeast*, 18(6):523–531, 2001.

Badr Hssina, Abdelkarim Merbouha, Hanane Ezzikouri, and Mohammed Erritali. A comparative study of decision tree ID3 and C4.5. *International Journal of Advanced Computer Science and Applications*, 4(2), 2014.

Daphne HEW Huberts and Ida J van der Klei. Moonlighting proteins: an intriguing mode of multitasking. *Biochimica et Biophysica Acta (BBA) - Molecular Cell Research*, 1803(4):520–525, 2010.

J. D. Hunter. Matplotlib: A 2D graphics environment. *Computing In Science & Engineering*, 9(3):90–95, 2007. doi: 10.1109/MCSE.2007.55.

Lawrence Hunter. *Molecular Biology for Computer Scientists*. AAAI Press, 1993. ISBN 0-262-58115-9.

Constance J. Jeffery. Why study moonlighting proteins? *Frontiers in genetics*, 6, 2015.

Lars Juhl Jensen, Ramneek Gupta, H.-H. Staerfeldt, and Søren Brunak. Prediction of human protein function according to Gene Ontology categories. *Bioinformatics*, 19(5):635–642, 2003.

Biaobin Jiang, Kyle Kloster, David F. Gleich, and Michael Gribskov. AptRank: an adaptive PageRank model for protein function prediction on bi-relational graphs. *Bioinformatics*, 33(12):1829–1836, 2017.

Yuxiang Jiang, Tal Ronnen Oron, Wyatt T. Clark, Asma R. Bankapur, Daniel D'Andrea, Rosalba Lepore, Christopher S. Funk, Indika Kahanda, Karin M. Verspoor, Asa Ben-Hur, et al. An expanded evaluation of protein function prediction methods shows an improvement in accuracy. *Genome Biology*, 17(1):184, 2016.

Craig E. Jones, Julian Schwerdt, Tessa Arwen Bretag, Ute Baumann, and Alfred L. Brown. GOSLING: a rule-based protein annotator using BLAST and GO. *Bioinformatics*, 24(22):2628–2629, 2008.

Indika Kahanda and Asa Ben-Hur. GOstruct 2.0: Automated Protein Function Prediction for Annotated Proteins. In *Proceedings of the 8th ACM International Conference on Bioinformatics, Computational Biology, and Health Informatics*, pages 60–66. ACM, 2017. ISBN 978-1-4503-4722-8.

Kazutaka Katoh, Kei-ichi Kuma, Hiroyuki Toh, and Takashi Miyata. MAFFT version 5: improvement in accuracy of multiple sequence alignment. *Nucleic acids research*, 33(2):511–518, 2005.

Diederik P. Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. *CoRR*, abs/1412.6980, 2014.

Koutroumbas Konstantinos and Theodoridis Sergios. *Pattern Recognition, 4th Edition*. Academic Press, 2008. ISBN 9781597492720.

Patrik Koskinen, Petri Törönen, Jussi Nokso-Koivisto, and Liisa Holm. PANNZER: high-throughput functional annotation of uncharacterized proteins in an error-prone environment. *Bioinformatics*, 31(10):1544–1552, 2015.

Miroslav Kubat. *An Introduction to Machine Learning*. Springer International Publishing, 2015. ISBN 978-3-319-20009-5.

Max Kuhn and Kjell Johnson. *Applied predictive modeling*, volume 26. Springer, 2013. ISBN 978-1-4614-6849-3.

Aleix Lafita, Spencer Bliven, Andriy Kryshtafovych, Martino Bertoni, Bohdan Monastyrskyy, Jose M. Duarte, Torsten Schwede, and Guido Capitani. Assessment of protein assembly prediction in CASP12. *Proteins: Structure, Function, and Bioinformatics*, 86(S1):247–256, 2018.

David A. Lee, Robert Rentzsch, and Christine Orengo. GeMMA: functional subfamily classification within superfamilies of predicted protein structural domains. *Nucleic acids research*, 38(3):720–737, 2009.

Rasko Leinonen, Federico Garcia Diez, David Binns, Wolfgang Fleischmann, Rodrigo Lopez, and Rolf Apweiler. UniProt archive. *Bioinformatics*, 20(17):3236–3237, 2004.

Arthur Lesk. *Introduction to bioinformatics*. Oxford University Press, 2013. ISBN 9780199251964.

Jundong Li, Kewei Cheng, Suhang Wang, Fred Morstatter, Robert P. Trevino, Jiliang Tang, and Huan Liu. Feature selection: A data perspective. *ACM Computing Surveys (CSUR)*, 50(6):94:1–94:45, 2017a.

Shumin Li, Junjie Chen, and Bin Liu. Protein remote homology detection based on bidirectional long short-term memory. *BMC Bioinformatics*, 18(1):443, Oct 2017b.

Xueliang Liu. Deep recurrent neural network for protein function prediction from sequence. *arXiv preprint arXiv:1701.08318*, 2017.

Anna E. Lobley, Timothy Nugent, Christine A. Orengo, and David T.' Jones. FFPred: an integrated feature-based function prediction server for vertebrate proteomes. *Nucleic acids research*, 36(suppl_2):W297–W302, 2008.

Martin Madera, Ryan Calmus, Grant Thiltgen, Kevin Karplus, and Julian Gough. Improving protein secondary structure prediction using a simple k-mer model. *Bioinformatics*, 26(5):596–602, 02 2010.

Gaston K. Mazandu, Emile R. Chimusa, and Nicola J. Mulder. Gene Ontology semantic similarity tools: survey on features and challenges for biological knowledge discovery. *Briefings in Bioinformatics*, 18(5): 886–901, 2017.

Wes McKinney. Data Structures for Statistical Computing in Python. In Stéfan van der Walt and Jarrod Millman, editors, *Proceedings of the 9th Python in Science Conference*, volume 445, pages 51–56. Austin, TX, 2010.

Federico Minneci, Damiano Piovesan, Domenico Cozzetto, and David T. Jones. FFPred 2.0: Improved Homology-Independent Prediction of Gene Ontology Terms for Eukaryotic Protein Sequences. *PLOS ONE*, 8(5):1–10, 05 2013.

Dmytro Mishkin, Nikolay Sergievskiy, and Jiri Matas. Systematic evaluation of CNN advances on the ImageNet. *arXiv e-prints*, page arXiv:1606.02228, Jun 2016.

Tom M. Mitchel. *Machine Learning*. McGraw-Hill Science, 1997. ISBN 0070428077.

Sobhan Moosavi, Masoud Rahgozar, and Amir Rahimi. Protein function prediction using neighbor relativity in protein–protein interaction network. *Computational biology and chemistry*, 43:11–16, 2013.

John Moult, Krzysztof Fidelis, Andriy Kryshtafovych, Torsten Schwede, and Anna Tramontano. Critical assessment of methods of protein structure prediction (CASP)-Round XII. *Proteins: Structure, Function, and Bioinformatics*, 86:7–15, 2018.

Kevin P. Murphy. *Machine learning : a probabilistic perspective*. Adaptive computation and machine learning series. MIT Press, 2012. ISBN 978-0-262-01802-9.

Saul B. Needleman and Christian D. Wunsch. A general method applicable to the search for similarities in the amino acid sequence of two proteins. *Journal of molecular biology*, 48(3):443–453, 1970.

Christine A. Orengo, A. D. Michie, S. Jones, David T. Jones, M. B. Swindells, and Janet M. Thornton. CATH–a hierarchic classification of protein domain structures. *Structure*, 5(8):1093–1109, 1997.

Genevieve B. Orr and Klaus-Robert Müller. *Neural Networks: Tricks of the Trade*, volume 1524 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, Berlin, Heidelberg, 1998. ISBN 978-3-540-65311-0.

Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. The PageRank Citation Ranking: Bringing Order to the Web. Technical Report 1999-66, Stanford InfoLab, November 1999.

Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in PyTorch. In *NIPS-W*, 2017.

William R. Pearson. An introduction to sequence similarity ("homology") searching. *Current protocols in bioinformatics*, 42(1):3.1.1–3.1.8, 2013.

William R. Pearson. Finding Protein and Nucleotide Similarities with FASTA. *Current Protocols in Bioinformatics*, 53(1):3.9.1–3.9.25, 2016.

Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

Damiano Piovesan, Pier Luigi Martelli, Piero Fariselli, Andrea Zauli, Ivan Rossi, and Rita Casadio. BAR-PLUS: the Bologna Annotation Resource Plus for functional and structural annotation of protein sequences. *Nucleic Acids Research*, 39(suppl_2):W197–W202, 2011.

Chris P. Ponting and Robert R. Russell. The Natural History of Protein Domains. *Annual Review of Biophysics and Biomolecular Structure*, 31(1):45–71, 2002.

Giuseppe Profiti, Pier Luigi Martelli, and Rita Casadio. The Bologna Annotation Resource (BAR 3.0): improving protein functional annotation. *Nucleic Acids Research*, 45(W1):W285–W290, 2017.

J. Ross Quinlan. Induction of decision trees. *Machine Learning*, 1(1):81–106, 1986.

Robert Rentzsch and Christine A. Orengo. Protein function prediction using domain families. *BMC Bioinformatics*, 14(3):S5, Feb 2013.

Sebastian Ruder. An overview of gradient descent optimization algorithms. *arXiv preprint arXiv:1609.04747*, 2016.

Stephen John Sammut, Robert D. Finn, and Alex Bateman. Pfam 10 years on: 10 000 families and still growing. *Briefings in Bioinformatics*, 9(3):210–219, 2008.

Ganapathi Varma Saripella, Erik L. L. Sonnhammer, and Kristoffer Forslund. Benchmarking the next generation of homology inference tools. *Bioinformatics*, 32(17):2636–2641, 2016.

Kumara Sastry, David E. Goldberg, and Graham Kendall. Genetic algorithms. In *Search Methodologies: Introductory Tutorials in Optimization and Decision Support Techniques*, pages 93–117. Springer, 2014. ISBN 978-1-4614-6940-7.

Andrea Scaiewicz and Michael Levitt. The language of the protein universe. *Current opinion in genetics & development*, 35:50–56, 2015.

Benno Schwikowski, Peter Uetz, and Stanley Fields. A network of protein–protein interactions in yeast. *Nature biotechnology*, 18(12):1257, 2000.

John Shawe-Taylor and Nello Cristianini. *Kernel methods for pattern analysis*. Cambridge university press, 2004. ISBN 978-0-511-21060-0.

Ian Sillitoe, Tony E. Lewis, Alison Cuff, Sayoni Das, Paul Ashford, Natalie L. Dawson, Nicholas Furnham, Roman A. Laskowski, David Lee, Jonathan G. Lees, et al. CATH: comprehensive structural and functional annotations for genome sequences. *Nucleic acids research*, 43(D1):D376–D381, 2015.

Richard Simon. *Resampling Strategies for Model Assessment and Selection*, pages 173–186. Springer, 2007. ISBN 978-0-387-47509-7.

Temple F. Smith and Michael S. Waterman. Comparison of biosequences. *Advances in Applied Mathematics*, 2(4):482–489, 1981.

Artem A. Sokolov and Asa A. Ben-Hur. Hierarchical classification of gene ontology terms using the GOstruct method. *Journal of bioinformatics and computational biology*, 8(02):357–376, 2010.

Erik L. L. Sonnhammer, Sean R. Eddy, and Richard Durbin. Pfam: A Comprehensive Database of Protein Domain Families Based on Seed Alignments. *Proteins: Structure, Function, and Bioinformatics*, 28(3):405–420, 1997.

Erik L. L. Sonnhammer, Sean R. Eddy, Ewan Birney, Alex Bateman, and Richard Durbin. Pfam: multiple sequence alignments and HMM-profiles of protein domains. *Nucleic acids research*, 26(1):320–322, 1998.

Pingping Sun, Xian Tan, Sijia Guo, Jingbo Zhang, Bojian Sun, Ning Du, Han Wang, and Hui Sun. Protein Function Prediction Using Function Associations in Protein–Protein Interaction Network. *IEEE Access*, 6:30892–30902, 2018.

Baris E. Suzek, Yuqi Wang, Hongzhan Huang, Peter B. McGarvey, Cathy H. Wu, and UniProt Consortium. UniRef clusters: a comprehensive and scalable alternative for improving sequence similarity searches. *Bioinformatics*, 31(6):926–932, 2015.

Tatiana A. Tatusova and Thomas L. Madden. BLAST 2 Sequences, a new tool for comparing protein and nucleotide sequences. *FEMS Microbiology Letters*, 174(2):247–250, 1999.

Günter Theißen. Orthology: Secret life of genes. *Nature*, 415(6873):741–741, 2002.

Janet M. Thornton, Christine A. Orengo, Annabel E. Todd, and Frances M. G. Pearl. Protein folds, functions and evolution. *Journal of Molecular Biology*, 293(2):333–342, 1999.

UniProt Consortium. UniProt: the universal protein knowledgebase. *Nucleic acids research*, 45(D1):D158–D169, 2017.

Sheng Wang, Hyunghoon Cho, ChengXiang Zhai, Bonnie Berger, and Jian Peng. Exploiting ontology graph for predicting sparsely annotated gene function. *Bioinformatics*, 31(12):i357–i364, 2015.

Sheng Wang, Meng Qu, and Jian Peng. *PROSNET: INTEGRATING HOMOLOGY WITH MOLECULAR NETWORKS FOR PROTEIN FUNCTION PREDICTION*, pages 27–38. WORLD SCIENTIFIC, 2017.

Mark N. Wass and Michael J. E. Sternberg. ConFunc—functional annotation in the twilight zone. *Bioinformatics*, 24(6):798–806, 2008.

Wei Xiong, Luyu Xie, Shuigeng Zhou, and Jihong Guan. Active learning for protein function prediction in protein–protein interaction networks. *Neurocomputing*, 145:44–52, 2014.

Guoxian Yu, Hailong Zhu, Carlotta Domeniconi, and Jiming Liu. Predicting protein function via downward random walks on a gene ontology. *BMC bioinformatics*, 16(1):271, Aug 2015.

Xianchun Zou, Guijun Wang, and Guoxian Yu. Protein Function Prediction Using Deep Restricted Boltzmann Machines. *BioMed research international*, 2017, 2017.