Application of Amino Acid Property Grouping Technique to Predict Protein Function Using Natural Language Processing Algorithms

By

Ritwik Katiyar, B.S. Bioinformatics

A thesis submitted to the Graduate Committee of

Ramapo College of New Jersey in partial fulfillment

of the requirements for the degree of

Master of Science in Data Science

Fall, 2023

Committee Members:

Dr. Osai Tweneboah, Advisor

Dr. Scott Frees, Reader

Dr. Sourav Dutta, Reader

COPYRIGHT

Dedication

To my mom who always believed in me.

Thank you for all that you did.

I hope that I made you proud.

Acknowledgements

I would like to acknowledge and give thanks to my thesis advisor Dr. Osai Tweneboah who assisted and provided guidance and direction for this thesis. Additionally, for guiding my research and aiding with machine learning algorithms, sampling techniques as well as data visualization and the overall writing of this report.

Furthermore, I would like to thank the committee members Dr. Scott Frees and Dr. Sourav Dutta for providing valuable input and support. A special thanks to Dr. Amanda Beecher and Dr. Ashley Stuart for providing support and advisement on data visualizations and on the biological nature of this study respectively. Finally, I would like to thank Ramapo College for providing me with the opportunity to pursue this avenue of research

Table of Contents

List of Tables

List of Figures

Abstract

Diverse biological processes are orchestrated by proteins, the fundamental building blocks of life, utilizing distinctive amino acid sequences. Traditional methods for determining protein function are often labor-intensive and time-consuming. To address this challenge, computational tools, particularly those rooted in Natural Language Processing (NLP), offer a promising alternative for efficient protein function prediction. This paper reviews the information-based and common NLP algorithms and techniques along with amino acid property grouping technique.

A significant focus is placed on the application of NLP techniques, acknowledging the complexity of deciphering the intricate language of proteins. Despite the multifunctionality of proteins and evolutionary nuances, NLP-based approaches show potential for accelerating protein function identification, with direct implications for drug discovery, personalized medicine, protein engineering, and bio-manufacturing.

The paper presents a detailed analysis of the amino acid property grouping technique's impact on predictive capabilities across different machine learning algorithms. Results indicate that this technique, when integrated with specific algorithms, holds promise in enhancing predictive accuracy.

The study concludes by proposing future directions, emphasizing the need for further analyses, alternative sampling techniques, and leveraging high-performance computing. It suggests exploring the transformative potential of the Transformer algorithm in protein function prediction, with the possibility of integrating a specialized encoder based on amino acid property grouping. This comprehensive exploration aims to contribute to the evolving landscape of computational methodologies in deciphering and predicting protein functions, holding immense promise for advancements in biological research, medicine, and biotechnology.

Introduction

Proteins are the building blocks of life, orchestrating various biological processes within living organisms. Their diverse functions, ranging from catalyzing biochemical reactions to providing structural support, are fundamentally determined by their unique sequences of amino acids. Deciphering the intricate relationship between a protein's sequence and its function has long been a focal point in biological research.[7]

Before delving into the various techniques and methodologies for predicting protein function, it is essential to establish a clear understanding of the term 'function' in this context. From an evolutionary perspective, a protein's function is to enhance overall organismal processes. However, within the scope of this study, function specifically denotes the distinct biological role a protein plays within a cell. Proteins can serve various functions, such as providing structural support, facilitating molecular transport, acting as receptors or inhibitors, among others. While it is impractical to enumerate all possible protein functions in this paper, Figure 2.1 provides a brief overview of protein classification in humans. Having established a comprehensive understanding of protein functions, we can now further explore the applications and techniques employed. This discussion will shed light on the methodologies that were specifically utilized to unravel the intricacies of protein functionality, providing valuable insights into the innovative approaches applied in protein research.[21]

Conventional methods for determining protein function, such as experimental assays and structural analysis, are labor-intensive, time-consuming, and often limited in scope.[23][22] Herein lies the motivation for exploring alternative approaches that harness the power of computational

tools, to sift through vast amounts of biological data and predict protein functions more efficiently.

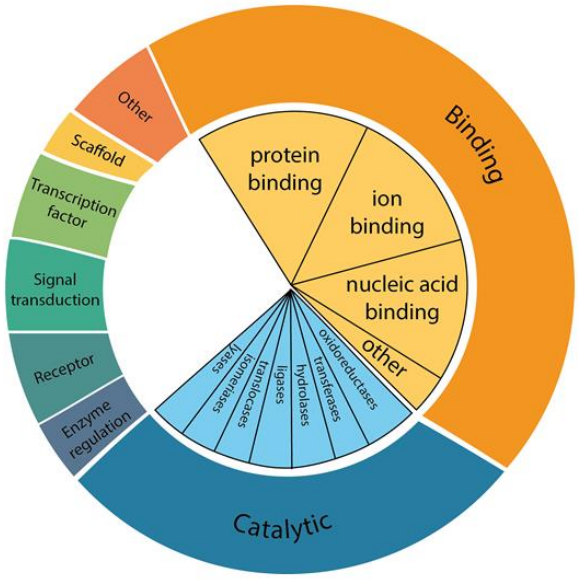


Figure 1.1 illustrates the classification of proteins based on their biological functions in humans. These groups can further be subdivided into specific categories. Enzymes, categorized as catalytic proteins, play a role in accelerating chemical reactions, with subgroupings based on the reactions they are involved in. Similarly, binding proteins can be subcategorized based on the molecules they interact with. [21]

Some of the computational tools and methods used in the field vary based upon a specific approach. For example, one such approach depends on finding proteins homologues proteins with the use of Basic Local Alignment Search Tool (BLAST) algorithm. This algorithm is then used to search a query sequence against a pre-existing protein database that contains experimentally determined protein function information. [7]

Yet another approach relies upon the use of networks. One such network is the protein – to – protein interactions network which assumes that proteins that interact with one another generally share a similar function. In another instance, protein information is used to generate predictions. [7]

The information approach relies on the use of various predicted information about the protein such as pKa (acidity of an amino acid [14]), Hydrophobicity, Charge, Secondary structure, etc. to ascertain the function of a protein[8] . Such approaches generally use similarity-based algorithms (SVM, Random Forest, Apriori, and naïve Bayes classifiers etc.) and focus entirely on obtaining as much information as possible on the protein prior to prediction. However, this method relies heavily upon the use of predicted information, which can or cannot be true. Additionally, this information-based approach is where many of the NLP techniques were first applied with the assumption that a protein sequence may be a complex language that can be decoded and understood. However, that is easier said than done; proteins often exhibit multifunctionality, where a single protein may have diverse roles within different cellular contexts. Moreover, the evolutionary relationships between proteins can result in functional similarities despite differences in sequences. Deciphering these nuances using NLP methods requires even more sophisticated algorithms capable of understanding and contextualizing this complex biological language [7].

Despite these challenges, the field offers promising opportunities. NLP-based approaches have the potential to accelerate the identification of protein functions, facilitate drug discovery by identifying potential drug targets, and enhance our understanding of biological systems [13]. NLP based protein function prediction has direct implications in drug discovery. By identifying potential drug targets more efficiently, these methods streamline the process of drug development [15]. Additionally, understanding protein functions at a deeper level allows for the design of more targeted and effective therapeutic interventions [18]. For instance, personalized medicine could benefit significantly from these advancements, tailoring treatments based on an individual's unique protein functionalities and variations. Furthermore, the integration of

multiple data sources and the continuous advancement of machine learning techniques are expanding the horizons of protein function prediction [7].

However, the benefits are not limited to Drug discovery, predicting protein functions has implementation in protein engineering, and bio-manufacturing as well. NLP assisted approaches contribute to designing enzymes (catalysts) for specific industrial applications, optimizing bioproduction processes and creating novel protein-based materials with highly tailored functionalities [21].

The convergence of NLP and protein function prediction presents a compelling avenue in biological research. By harnessing the power of language processing techniques, researchers aim to decode the intricate language of proteins and unravel their functions more efficiently. As advancements in computational methodologies continue to evolve, the prospect of unlocking the full potential of NLP in predicting protein function holds immense promise for revolutionizing biological research and its practical applications in medicine and biotechnology.

The proceeding chapter explores completed research, algorithms, and techniques that have been developed or are currently in use. The methodology chapter explores the application of amino acid property grouping and the use of nonconventional algorithms and techniques that were explored in this study. Which leads to a chapter on the evaluation, and discussion of results. The conclusion section summarizes the overarching results and the verdict that can be drawn. The future works chapter outlines what can be accomplished moving forward from this study and the viability of the amino acid property grouping technique moving forward.

Background

In the preceding chapter, the employment of predicted information in early machine learning algorithms inadvertently led to the application of Natural Language Processing (NLP) techniques for protein sequences, a technique commonly referred to as feature embedding. Noteworthy features, such as predicted secondary and tertiary structures, pKa, and hydrophobicity, are commonly gathered for analysis [6]. This perspective conceptualizes protein sequences akin to language, where proteins, composed of amino acids, exhibit diverse chemical compositions. There are approximately 20 to 22 amino acids constituting proteins that vary in their properties, influencing acidity, charge, hydrophobicity, and other chemical characteristics, as detailed in Table-A1 in the appendix. Amino acids are interconnected through peptide bonds, and their distinct properties contribute to the formation of secondary structures. The three predominant types of secondary structures are elongated coils, helixes, and beta sheets [4]. As we delve into the intricate world of proteins, akin to deciphering a language written in amino acid sequences, it becomes essential to employ advanced techniques such as the Term Frequency-Inverse Document Frequency (TFIDF) approach for a nuanced analysis of protein data. This method allows us to explore the unique chemical compositions and interconnectedness of amino acids, paving the way for a deeper understanding of the complex structures that govern diverse biological processes. [6]

Employing the Term Frequency-Inverse Document Frequency (TFIDF) technique, a natural language processing approach, enables the analysis of protein data. Mathematically, TFIDF is expressed as a product of term frequency (tf) and inverse document frequency (idf), calculated as the logarithm of the total number of documents divided by the number of documents containing the term.

$$\boldsymbol{tfidf(t,d,D) = tf(t,d) * idf(t,D)} \quad (2,1)$$

Where tf is the term frequency which can be defined as the frequency of t in d and idf is the inverse document frequency which is defined as log (total amount of documents / number of documents with t) [35]

This TFIDF approach is applicable to proteins, allowing the calculation of TFIDF for individual amino acids and secondary structures within a sequence, providing a relative outline of the protein sequence composition. Additionally, the identification of common tertiary structures or domains and motifs, often determined using BLAST, contributes valuable information to sequence analysis. However, for predicting protein functions, the focus remains primarily on secondary and tertiary structures, as quaternary structures are typically too large for analysis of many smaller proteins. [6] Figure 2.1 below as they relate to one another.

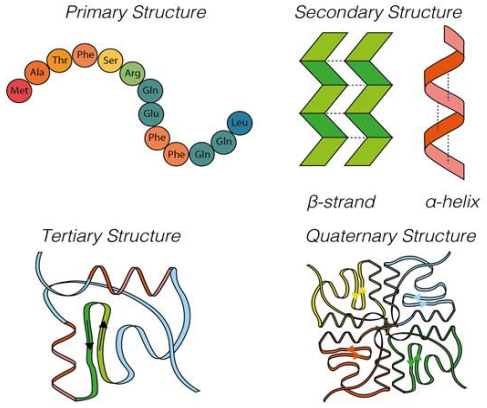


Figure 2.1: illustrates how a sequence of amino acids can be translated into secondary, tertiary, and quaternary structures. [22]

Leveraging information about individual amino acids and common structures, binary or multiclassification machine learning algorithms can be employed for protein function prediction. Commonly used algorithms include Support Vector Machines (SVMs), Decision Trees, Random

Forests, Deep Neural Networks (DNNs), Convolution Neural Networks (CNNs), and Recurrent Neural Networks (RNNs). [23][36][12]

Support Vector Machines (SVM)

SVM is a supervised machine learning algorithm that is used for binary classification. SVM attempts to classify the data by finding the best possible hyperplane that classifies the data points. To find the best possible hyperplane, the algorithm attempts to find the plane that has the maximum margin or the maximum distance between two data points of both classes. Figure 2.2 showcases the hyperplane selection by the SVM algorithm while it maintains the maximum margin. [27]

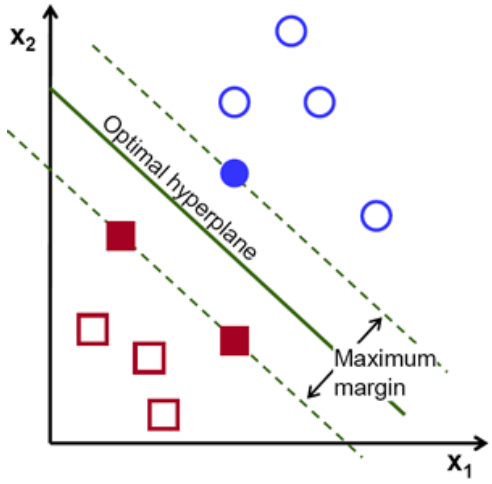


Figure 2.2: An example of SVM finding the best hyperplane by using the maximum distance [27]

A standard SVM is designed to work best with liner data. For nonlinear or protein data, the SVM is modified such that data gets mapped to a higher dimensional space. These modifications are referred to as kernels. [8]

There are a myriad of papers that discuss the application of the variety of different SVM kernels for protein function prediction. However, one of the most common kernel that is often deployed is the polynomial SVM kernel. The polynomial kernel uses a polynomial function to

redefine the data into a higher dimension. Mathematically a polynomial kernel can be defined as follows: [27]

$$\boldsymbol{K(x_1, x_2) = (x_1^T x_2 + c)^d} \qquad (2,2)$$

The equation above shows the mathematical representation of a polynomial kernel between two features. where c is some constant and x_1 and x_2 are vectors and d is the degree of polynomial. The degree of 3 is commonly used for larger datasets. [27]

Decision Tree and Random Forest

Decision trees are non-parametric supervised learning methods that are commonly used for classification and regression. The objective of a decision tree is to build a model that can predict the value of the target by learning simple decision rules from the features provided.

Most decision trees use the Gini index to determine the nodes it needs to select. The Gini score is a measure of the impurity of the node. A node is considered pure if all training instances it applies to belong to the same class. The score can be defined as: [16]

$$\boldsymbol{G_i = 1 - \sum_{k=1}^{n} P_{i,k}^2} \qquad (2,3)$$

Gini score impurity score; where $P_{i,k}$ is the ratio of class k instances in the ith node. Although with the implementation of this technique, if the same impurity score is shared between several features during a split, the split can be selected at random. [16]

Hence, Random Forest is largely considered over decision trees, as a random forest algorithm is simply a collection or an ensemble of decision trees. Each decision tree within the

ensemble is built upon a random subset of input features, and eventually the majority vote among the trees is taken to predict results.

Deep Neural Network (DNN)

A neural network aims to mimic the human brain and how the neurons interact with one another. Like neurons the neural networks are made up of perceptron's. Invented in 1957 by Frank Rosenblatt, perception uses a weighted sum to generate output from an input. Figure 2.3 below provides a visual representation of what the weighted sum appears as in the neural network algorithm [16].

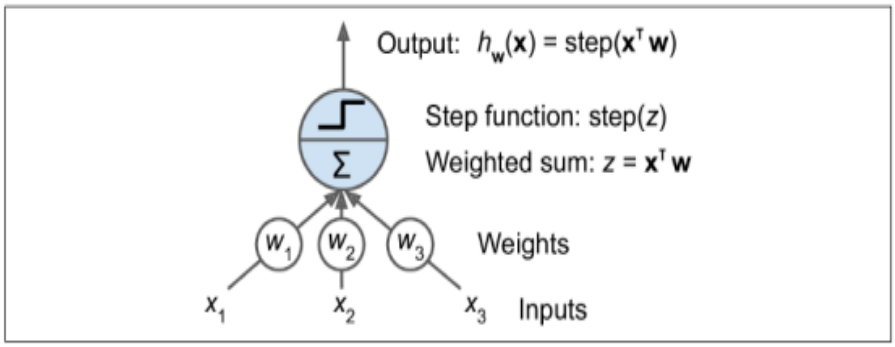
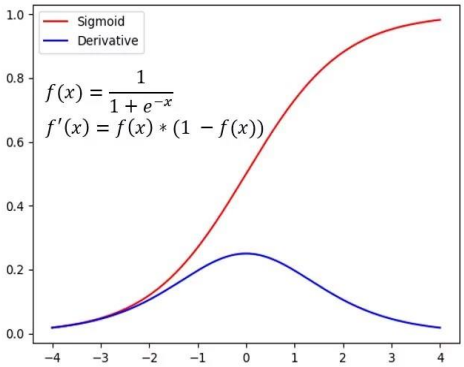


Figure 2.3: Shows the threshold logic unit perception which computes a weighted sum of its inputs and applies a step function. [16]

The perceptron can be used together side by side in an input and output layer to generate results based upon an activation function which determines the output of the perceptron. The most used activation function is the rectified linear unity (ReLU) function. The ReLU function activation function primarily assists with the performance of the neural network by replacing negative values with 0 and only keeping only the positive value. However, to get a meaningful output the output node is generally sent to have a logistic activation function. [29]



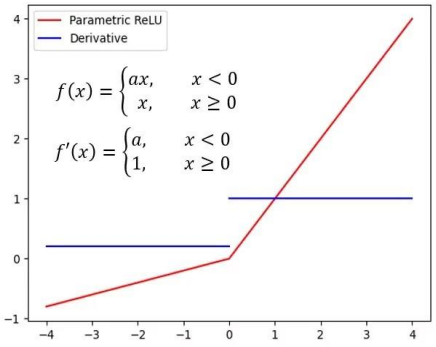


Figure 2.4a and 2.4b: The image on the right shows the ReLU activation function represented on a graph. Similarly, the image on the left shows the sigmoidal (logistic) function represented on a graph. [29]

To build upon the complexity of a simple perceptron input and output layer model, addition hidden layers of perceptrons can be applied between the input and output layers. This sort of neural network is what is generally referred to as a Deep Neural Network (DNN). Figure 2.4a and b highlight the graphical as well as equation difference between ReLU and the sigmoidal function. With the introduction of more sophisticated and complex neural networks the way natural language as well as protein sequences are processed has changed dramatically. [29]

Recurrent Neural Network (RNN)

The advent of Recurrent Neural Networks (RNN) has ushered in a transformative paradigm shift in algorithmic language processing. RNN's can account for the sequential nature of language and process language while accounting for context. In essence, the RNN processes a sentence's meaning based on the sentence itself as well as all the words within a sentence [7].

Fundamentally, an RNN comprises of Long Short-Term Memory (LSTM) layers. The LSTM layer upholds a cell state, ensuring the preservation of information gradients throughout sequence processing. At each iteration, the LSTM meticulously evaluates the current word, the

carry, and the cell state. It accomplishes all this via the use of 3 different channels or weighted vectors, the first vector is referred to as “forget” gate where irrelevant information gets removed, then the input and output gates hands the current input as well as producing predictions for each time step. With the use of these techniques RNN can process language as is without the need for additional information, thereby applying tokenization or bag-of-words technique rather than the feature embedding techniques that were employed previously. [23] Figure 2.5 shows the conceptual similarities between protein and English language processing for RNN which provides validity of applying RNN in similar vein to English language processing.

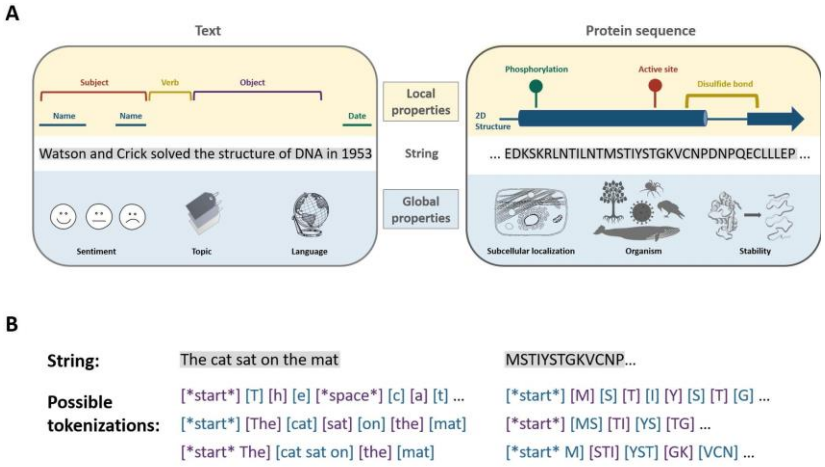


Figure 2.5: A side-by-side display of the common bag-of-words tokenization technique for the English language as well as the protein sequence. **[23]**

The processing of language by Recurrent Neural Networks (RNNs) involves the tokenization of language. In essence, a sentence is decomposed into tokens, resulting in a two-dimensional array where each row signifies a distinct step, and each column denotes words encoded into specific numeric values. Over recent years, several tokenization methodologies have been developed, with a focus on encoding protein layers based on specific criteria. [23]

The prevalent criterion has been the encoding of amino acids as unique words, adhering to the bag-of-words technique. However, this form of tokenization exhibits limitations in terms of

accuracy and processing complexity. Consequently, alternative tokenization approaches have been devised. ProLanGo, for instance, represents an RNN model that employs the k-mers algorithm, a bioinformatics-based probabilistic model. This model integrates the k-mers algorithm with the bag-of-words methodology, yielding superior results compared to the sole utilization of the bag-of-words technique. [7]

Convolution Neural Network (CNN)

Convolutional Neural Networks (CNNs) were originally tailored for image processing. Nevertheless, the application of CNNs to textual data becomes feasible through tokenization and the conversion of language into two-dimensional numerical arrays. CNN architectures encompass convolutional layers comprising specialized perceptrons, which can be either pooled or fully connected. Figure 2.6 shows the visual representation of the convolution image layers and how its processed. This algorithm is designed to emphasize crucial or distinctive features, such as colors or shapes, seeking common patterns for the identification or classification of objects [34].

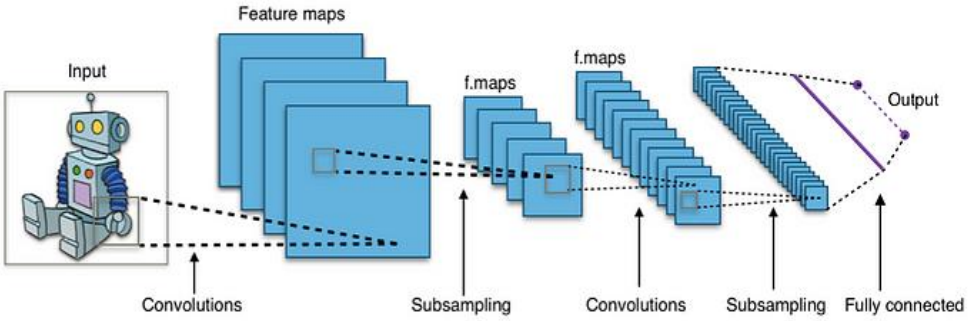


Figure 2.6: An example of a standard CNN architecture for images [21].

Within the domain of protein sequence-to-function prediction, CNNs find diverse applications. A notable example is ProtConv, which employs feature embedding or information techniques previously discussed to transform data into a two-dimensional array. By leveraging

reduction techniques, the data is condensed into a 28x28 heatmap image. This innovative approach essentially converts a protein sequence into an image, facilitating processing by CNNs for predictive analysis. Notably, ProtConv demonstrates an accuracy of up to 80% with their unique methodology [26].

It is noteworthy that the tokenization technique employed for Recurrent Neural Networks (RNNs) can also be adapted for CNNs. While this technique is commonly utilized in natural language processing, ongoing research is exploring its efficacy in the context of protein sequences.

Methodology

As discussed in the preceding chapter, amino acids possess distinctive properties inherent to them which can be used to differentiate one from another. Furthermore, these unique properties enable the categorization of amino acids into distinct groups based on their characteristic features. It is imperative to acknowledge that the categorization of amino acids is not universally straightforward, as certain amino acids exhibit shared properties. Consequently, a decision has been made to adopt the classification scheme provided by Sigma Aldrich for grouping specific amino acids [3]. In adherence to this scheme, amino acids are systematically categorized based on predetermined criteria, facilitating a more systematic and organized approach to their classification:

Amino Acids	Property
Alanine (A). Isoleucine (I), Leucine (L), Methionine (M), Valine (V)	Aliphatic (A)
Phenylalanine (F), Tryptophan (W), Tyrosine (Y)	Aromatic (R)
Asparagine (N), Cysteine (C), Glutamine (Q), Serine (S), Threonine (T)	Polar (P)
Aspartic Acid (D), Glutamic Acid (E)	Acidic (C)
Arginine (R), Histidine (H), Lysine (K)	Basic (B)
Glycine (G), Proline (P)	Unique (U)
Pyrrolysine (O), Selenocysteine (U), Glutamic acid or Glutamine (Z)	Super Rare (S)

Table 3.1: illustrates the grouping of amino acids based on their unique chemical properties.

This way of amino acid grouping serves as a foundation for manipulating sequences to compute grouping data. For instance, a sequence may be transformed as follows:

MMNAQKSKIA…. → AAPAPBPBAA….

Through this sequence alteration, the objective is to evaluate the efficacy of the technique using the aforementioned machine learning and deep learning algorithms.

Data Background

The CAFA5 dataset was procured from Kaggle, originating from The Gene Ontology Resource organization [5]. The Gene Ontology database contains information on human and machine-readable genes, established to facilitate the computational analysis of genes and proteins [11]. The dataset comprises taxonomic, Go-term (functional), and sequential data of proteins, totaling 142,246 sequences. Therefore, owing to the extensive number of sequences within the dataset, it became imperative to process and optimize the sequences suitably for utilization in machine learning and for exploratory data analysis. To achieve this, Sequential Query Language (SQL) was employed for effective data handling and validation. The objective was to discern whether the sequences are associated with proteins from a singular organism or from diverse organisms. Consequently, a comprehensive count of all unique taxonomic IDs within the dataset was conducted, revealing a total of 3,131 distinct organisms associated with these sequences. Additionally, it was discovered that the maximum sequence length was 35,375 amino acids, while the minimum was just three amino acids, with an average length of 554 amino acids across all sequences. The protein sequence of length three was obtained from a mushroom, where the protein is responsible for regulation or inhibition. Alternatively, the maximum length sequence originates from the Norway rat and the protein is responsible for development of muscle tissues. The taxonomic data's magnitude is illustrated in Figure B-1 of the appendix. Additionally, to observe

a more detailed distribution of sequences in terms of their taxonomic origins is shown in Figure 3.1.

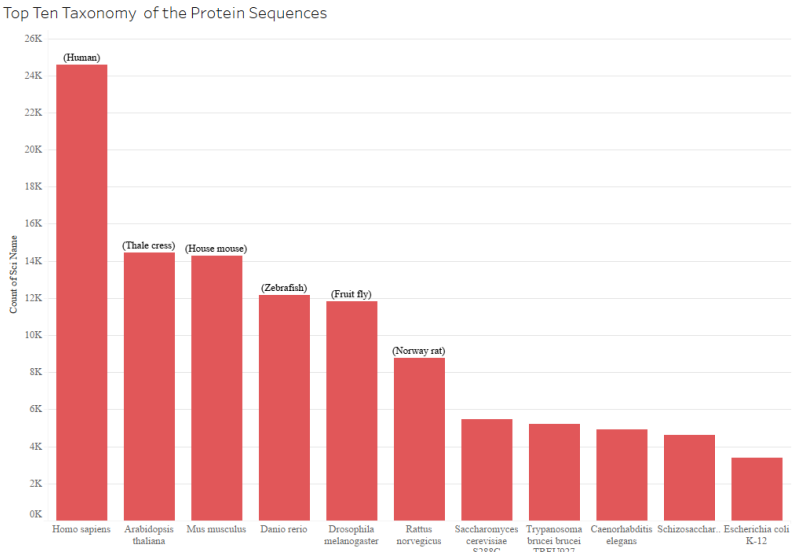


Figure 3.1: the plot displays the top ten protein sequences affiliated with a specific taxonomy. The nomenclature above the plot denotes the common names for the respective organisms.

Given the substantial variation in taxonomic and sequence data, the total number of unique functions mappable to a protein sequence is 43,248. Figure B-2 in the appendix provides an overview of all functions and the corresponding number of proteins associated with each function. The subsequent figure, Figure 3.2 specifically outlines the top 20 functions within the CAFA5 dataset, along with the corresponding number of amino acids mapped to each function.

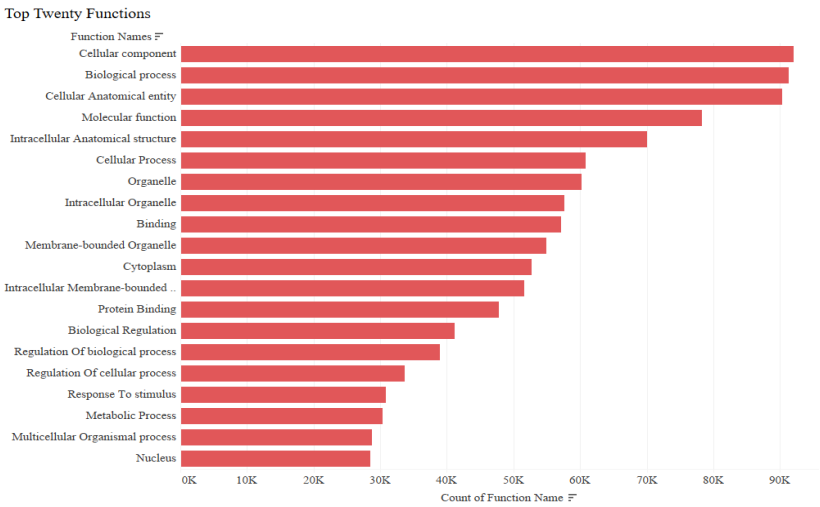


Figure 3.2: The top 20 functions within the CAFA5 dataset.

Upon evaluating the dataset, a significant challenge emerges—its sheer size renders it impractical for processing in its entirety without the use of specialized high-performance computers. To address this, a decision was made to extract a representative sample that captures the variation in protein sequences. The focus was on protein sequences from diverse taxonomies, aiming to encompass a high level of variation within the sequences. Leveraging the Synthetic Minority Oversampling Technique (SMOTE), one sequence was randomly sampled from each taxonomy, resulting in a reduction of the dataset from 142,246 to 3,131 protein sequences. Subsequently, an exploratory data analysis of the functions was conducted on the newly sampled data to ensure its adherence to a similar pattern to the original dataset.

A plot analogous to Figure B-3 was generated for the sampled data to assess the distribution of functions within the dataset. The distribution in the sample closely mirrors that of the original dataset; however, a substantial reduction in the number of functions is evident. The sampled dataset comprises only 5,206 unique functions compared to the 43,248 functions present in the CAFA5 dataset. This reduction is an anticipated outcome, and the loss of specific protein functions is not deemed a significant concern. Given the considerable computational demands associated with running a multiclassification algorithm over 5,206 classes, a decision was made to narrow the focus to the top ten functions represented within our sample. Additionally, it is noteworthy that the minimum sequence length within this dataset remains three (pertaining to sequences from the same fungal species discussed earlier). The maximum sequence length has extended to 6,872 amino acids, while the average sequence length stands at 474 amino acids.

The functions, however, are extremely general and are much more applicable to various proteins. Suggesting that further testing with all function classifications may be necessary in the future. In general, however, the ‘molecular function’ corresponds to molecular - level activities

performed by gene products such as catalysis or transportation as opposed to non-gene product activities. The term 'Biological Process' tag refers to a proteins involvement in a larger muti-molecular activity such as DNA repair or signal transduction. 'Binding' and 'catalytic activity' are both border terms used to describe proteins that perform those functions. 'Cellular Component' asserts that the protein belongs to a specific or a larger cellular component. 'Cellular Processes' is a term used to describe a protein that is involved in processes such as the replication and transcription of DNA, or even production, processing, or secretion of other proteins. 'Cellular Anatomical entities' implies gene product proteins involved in cellular structures (such as plasma membrane or the cytoskeleton), membrane-enclosed cellular compartments, and other molecular functions. 'Protein binding' and 'Metabolic Processes' are proteins which bind with other proteins or are involved with the metabolic pathways (such as cellular respiration, photosynthesis, etc.) [10]. Ultimately, the 'Orgain Substance Metabolic Processes' alludes to proteins that are involved in chemical reactions and pathways which require organic acids, any acidic compound containing carbon in a covalent linkage [11]. Figure 3.3 showcases the functions that will be tested by various algorithms.

It is easy to observe that the sample data is imbalanced hence, accuracy alone can no longer be used as a performance measure for the algorithms. A common performance measure for imbalanced data is the F1-score which can be defined as:

$$F1 - Score = \frac{TP}{TP + \frac{1}{2}(FP+FN)} \quad (3,1)$$

F1 Score equals true positive (TP) over the sum of TP, one half times the sum of false positive (FP) and false negatives (FN) [16]. Furthermore, to establish the complexity of our data a bipartite network was constructed that mapped each protein sequence within the sample to its respective

function. For a visual representation of this network, please refer to Appendix Section B, Figure B-4.

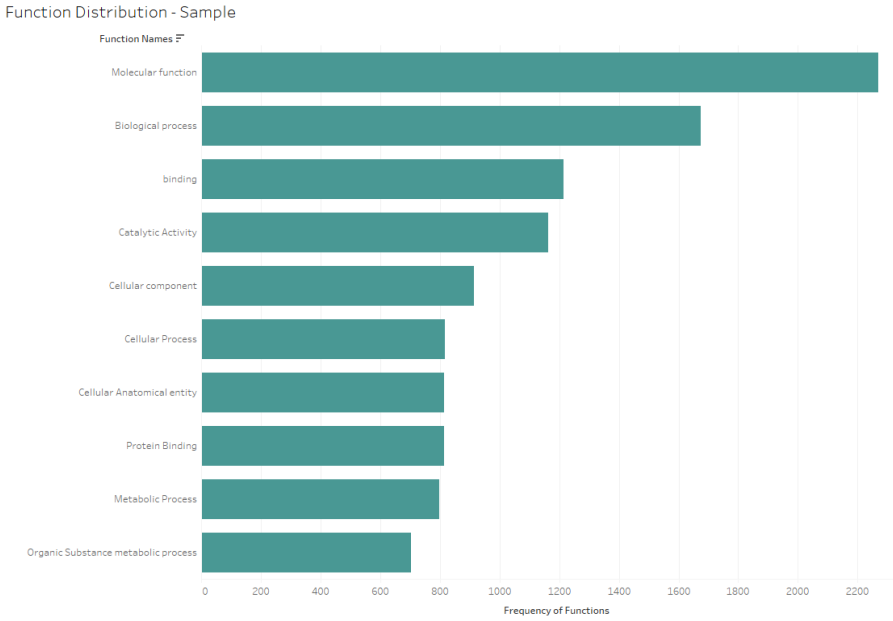


Figure 3.3: The top ten functions in the sample data set will be the primary focus of our prediction.

Additional Data Collection

As the primary objective of this study is to evaluate the viability of the amino acid property grouping technique for protein function prediction, we will employ the conventional approach of feature embedding, as previously outlined. This process necessitates the prediction of secondary and tertiary structures, hydrophobicity, and pKa.

Several well-documented methods are available for predicting tertiary and secondary structures in protein sequences. In this study, the Basic Local Alignment Search Tool (BLAST) was utilized to acquire predicted tertiary structure information. BLAST is widely acknowledged as a highly reliable algorithm in the field of bioinformatics. Leveraging the expansive database of The National Center for Biotechnology Information (NCBI), BLAST can identify sequences

within their database that closely align with a given protein sequence, thus providing accurate predictions [32].

To optimize the accuracy of our results, only the most robust methods were employed for predicting secondary and tertiary structures. BLAST was chosen for its established reliability and the comprehensive protein, DNA, and RNA information available through the NCBI database. This approach ensures that the predictions are based on sequences closely aligned with the studied protein sequence, enhancing the overall accuracy of the tertiary structure predictions. Moreover, the predictions obtained for BLAST were concise predictions that had an e-score of roughly 0.001 or less. A lesser e-score equates to a more accurate prediction by the algorithm [16].

In the context of secondary structure prediction, the S4PRED tool was employed to generate predictions from protein sequences. S4PRED is a non-homology-based technique that utilizes a Deep Neural Network for prediction, achieving an accuracy ranging from 76% to 85% [20]. The selection of S4PRED over homology-based models was primarily driven by time constraints. Homology modeling, although recognized for its accuracy, can be a time-consuming process. In contrast, S4PRED demonstrated efficiency by providing secondary structure predictions for all 3,131 sequences within the sample in under 3 hours. This expedited timeframe aligns with the study's objectives.

To calculate hydrophobicity and pKa values for the entire protein sequence, two tables of pre-determined amino acid pka and hydrophobicity values were summed. The tables can be found in section A of the appendix as Table A-1 and A-2 respectively.

After the compilation of all relevant information, the application of TFIDF (Term Frequency-Inverse Document Frequency) was performed on both the secondary structure, amino acid property grouping, and raw protein sequence (frequency data can be found in section C of the appendix as

Figures C-1, C-2 & C-3). This transformative step facilitated the preparation of the data for subsequent application of machine learning algorithms. SVM, decision tree, random forest, and Deep Neural Network algorithms were successfully applied to the processed data.

Application: SVM, Random Forest, DNN, Hard Voting & Soft Voting

Given our primary objective of prediction, we employed a train-test split methodology to evaluate the performance of our algorithms. This involved partitioning the data into training and testing sets, allocating 70% for training and 30% for testing purposes, respectively. After which, the Polynomial Kernel Support Vector Machine (SVM) was configured with a degree of three and a coefficient of one. The random forest underwent up to 100 iterations, as additional iterations yielded negligible improvements in algorithm performance. The Deep Neural Network (DNN) was structured with 100, 50, and 25 layers, sequentially, and an alpha value of 0.001. The Deep Neural Network utilized the scikit learn library as it is extensively documented and offers simple easy to use features for a basic neural network [9]. Furthermore, the maximum iterations were capped at 50. Moreover, the general model implementation structure is provided in Appendix E.

After executing the applications in isolation, the ensemble technique of hard and soft voting was applied. In hard voting, a majority voting mechanism among multiple algorithms is employed to determine the final predictions [2], as illustrated in Figure 3.4.

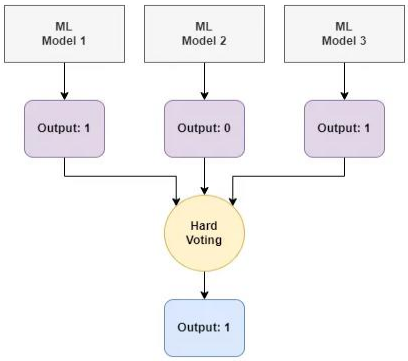


Figure 3.4: Depreciation of hard voting ensemble algorithm

On the other hand, soft voting leverages the probability estimates from each algorithm to formulate its predictions. Figure 3.5 presents a visual representation of the soft voting ensemble algorithm. [2]

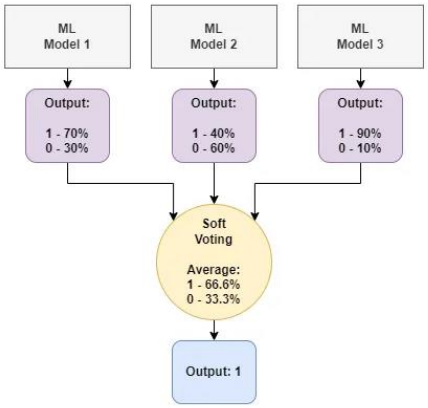


Figure 3.5: Depiction of the soft voting ensemble algorithm

Data Processing and Application: RNN

The processing of data for Recurrent Neural Network (RNN) involves the construction of a custom modified encoder that takes into consideration the properties of amino acids. To elucidate the structure of this encoder, we will utilize an example protein sequence comprising six amino acids: AILRST.

The encoder initially converts the amino acid string into numerical values, assigning a unique value to each amino acid. Consequently, the example sequence is translated into a list of numbers:

$$[1, 8, 10,14, 16, 17]$$

Once the string of amino acids is transformed into its respective numeric values, these values are then grouped based on the pre-determined amino acid property grouping:

$$[[1, 8, 10], [14], [16,17]]$$

With the grouped values established, attention is then directed to the sequential nature of the data. Padding is applied with zeros to transform the data from a list of lists to a 2-D array:

$$\begin{bmatrix} 0 & 0 & 0 & 1 & 8 & 10 \\ 0 & 0 & 1 & 8 & 10 & 14 \\ 1 & 8 & 10 & 14 & 16 & 17 \end{bmatrix}$$

However, for sequences exceeding 500 amino acids, truncation to the first 500 amino acids was necessary due to the computational intensity associated with processing and transforming arrays into 2D arrays beyond this length. Additionally, to enhance performance, the Singular Value Decomposition (SVD) dimensionality reduction algorithm was employed to further condense the array to a size of 28x28. The SVD is a matrix can be interpreted as the factorization of a matrix of three other matrices, such that: [19]

$$A = U \sum V^T \qquad (4.2)$$

Where, U is the *mxm* matrix of the orthonormal eigenvectors of AA^T. V^T is the transpose of *nxn* matrix containing the orthonormal eigenvectors of A^TA and Σ is the diagonal matrix with r elements equal to the root of the positive eigenvalues AA^T or A^TA. [19]

Following the factorization, the matrix undergoes a reduction process, isolating its fundamental components. This method proves particularly effective in addressing sparse matrices. Subsequently, the reduced matrix is integrated into the Recurrent Neural Network (RNN) model.

The implemented model comprises an input layer featuring a bidirectional Long Short-Term Memory (LSTM) layer, (Processes the input both forward and backwards) layer followed by a regular LSTM layer and two dense layers. This is succeeded by a regular LSTM layer and two dense layers. Notably, RNNs are computationally intensive, rendering them more efficient when executed on a Graphics Processing Unit (GPU). For GPU execution, the activation function is modified from Rectified Linear Unit (ReLU) to Hyperbolic Tangent (Tanh), the latter being akin to the sigmoidal activation function but with a slight variation in the equation [30]. Refer to Figure 3.6a for a graphical representation of the Tanh activation function. Moreover, a number of drop

out layers were introduce to prevent the algorithm from overfitting [1]. The Dropout layer insures that the model will be able to learn the broader differences and patterns rather than minute details that could hinder the results. A visual representation of how the dropout layer functions can be seen in figure 3.6b.

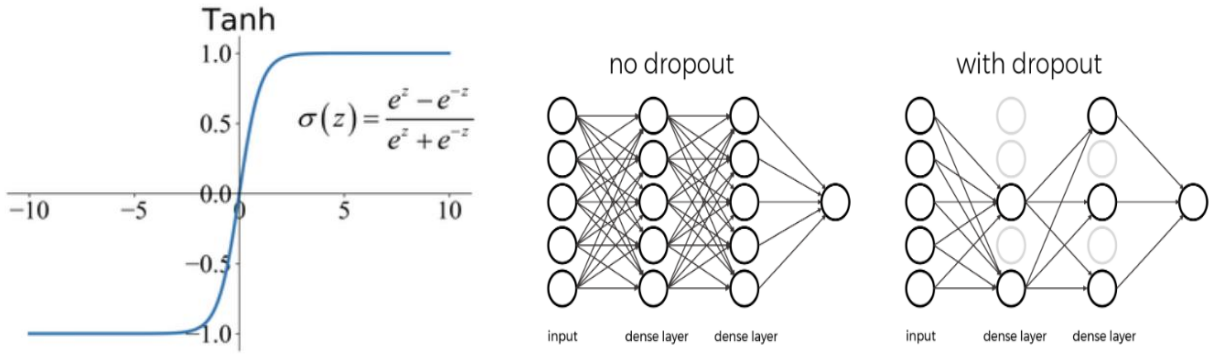


Figure 3.6a and 3.6b: Figure on the left (3.6a) shows the tanh activation function on a graph along with the Tanh equation. Figure on the right (3.6b) displays the dropout layer and how the layer drops neurons that do not meet a specified threshold to reduce overfitting. [29] [1]

Upon constructing the network, the final step involves processing the data for a Convolutional Neural Network (CNN), utilizing the matrix crafted for the RNN.

Data Processing and application: CNN

As elucidated in the study's background, RNN data applicability to CNNs is under examination. In this investigation, we evaluate efficiency by applying the amino property-based encoder matrix to the CNN. The data necessitates conversion into an image format for CNN processing.

To achieve this conversion, a heatmap represents the matrix as an image. Post-Singular Value Decomposition (SVD), the non-diagonal elements exhibit minimal values, potentially causing the heatmap to display a uniform color for low values, with only diagonal values containing meaningful data. To enhance data visualization, natural logarithm transformation is applied. Refer to Figure 3.7 for a heatmap image of an RNN matrix of a protein sequence converted for CNN

use. Moreover, the image configuration adopted RGB coloring, as investigated in the study titled *'Color-Net: Investigating the importance of color spaces for image classification.*' The research findings indicated that Convolutional Neural Networks (CNN) exhibit optimal performance when trained on RGB images, in contrast to grayscale (black or white) or other color formats [13]. Furthermore, pooling layers were introduced to the network. Pooling layers as opposed to convolution layers alter the dimensions of an image and reduces it by a scaling factor. In essence decreasing the resolution of an image. Which is meant to invoke a broader view of the image with the express purpose of reducing overfitting [1].

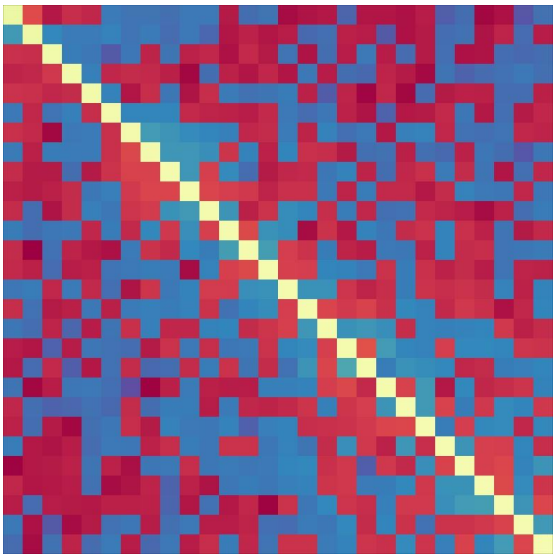


Figure 3.7: Shows the heatmap image of a RNN matrix of a protein sequence converted for the use of CNN. Blue represents low values; red consists of high values and yellow represents middle values.

With the processed image, the focus shifts to building the CNN. The network comprises three conventional 2D layers interspersed with two max pooling layers. Following the third layer, the output undergoes flattening and passes through two dense layers to yield results. Moreover, it is noteworthy that the utilization of the Ada-Delta optimizer yielded more favorable outcomes in comparison to the conventional Adam optimizer. Despite the shared categorization, the Ada-Delta optimizer exhibits a pronounced resemblance to the RMSProp optimizer [25]. Notably, empirical

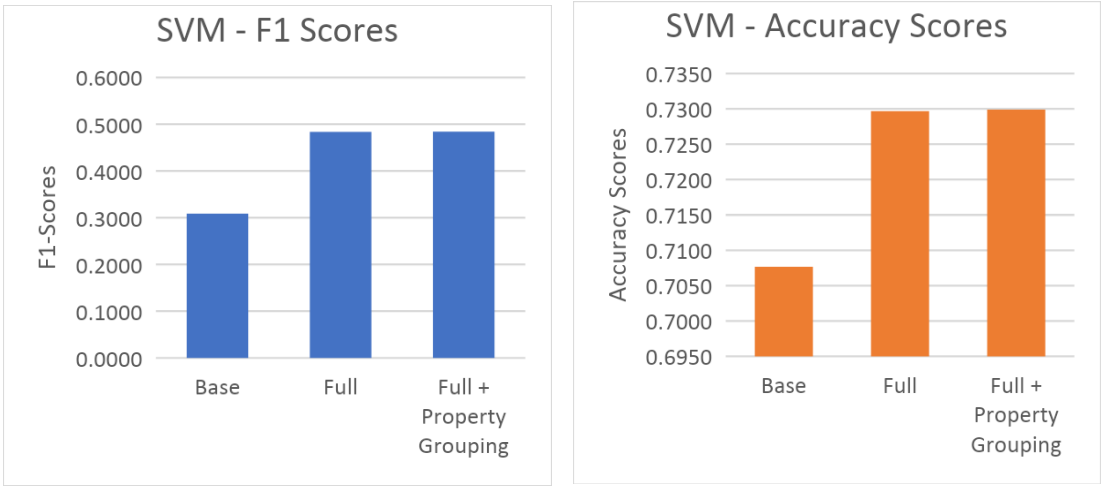
evidence underscores the efficacy of the RMSProp optimizer, particularly in scenarios where CNNs constitute a limited number of layers [17]. With our networks built and our data processed we can now observe the results that were obtained.

Results and Discussion

Feature Embedding Algorithms

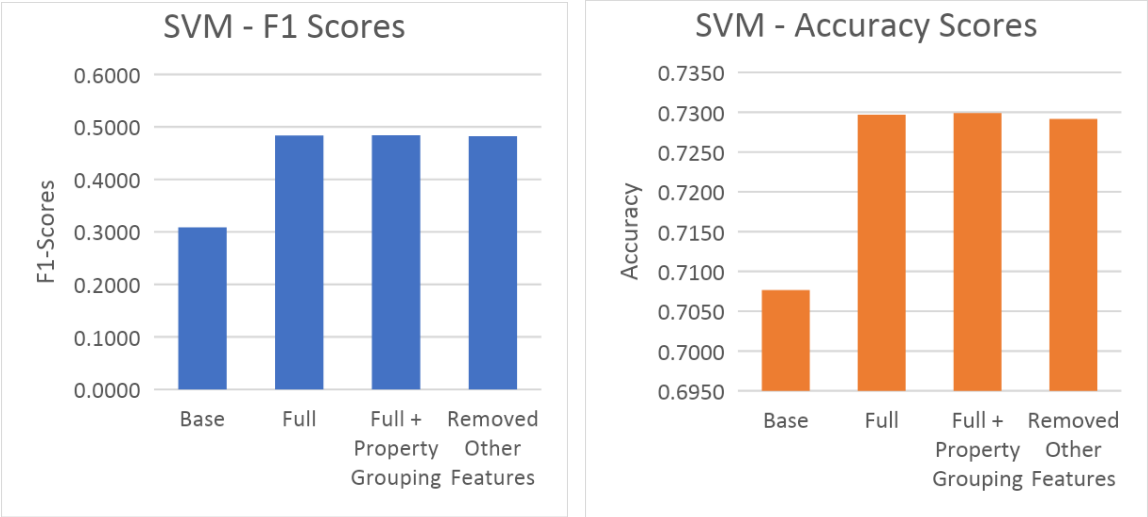
To assess the impact of amino acid property grouping on algorithmic performance, each algorithm utilizing feature embedding data underwent three separate runs with distinct datasets. The initial dataset comprised fundamental information, encompassing solely amino acid counts within sequences. The comprehensive dataset incorporated additional parameters such as Hydrophobicity, pKa, TFIDF of secondary sequence structures, tertiary structures, and TFIDF of amino acids. Subsequently, the TFIDF of the amino acid property grouping technique was introduced.

Traditionally, accuracy scores serve as the standard metric for algorithmic performance evaluation. However, owing to our specific sampling methodology and the inherent imbalances in our dataset, we also considered F1-Score. Figures 4.1 a and b highlight the scores obtained.



Figures 4.1a & 4.1b: Figures 16 and 17 illustrate the mean function prediction scores from SVM for various datasets.

The marginal disparity between the complete data and the data augmented with amino property grouping suggests minimal alteration. This could signify a potential conflict of information between the grouping technique and certain values within the dataset. To gain a more nuanced understanding, we eliminated pKa, Hydrophobicity, and secondary structure data to evaluate changes in scores, yielding the subsequent results:



Figures 4.2a and 4.2b: display the mean function prediction scores from SVM for diverse datasets, encompassing removed features

Observations from figure 4.2a and 4.2b reveal a subtle decrease in both F1-Score and Accuracy when certain features are excluded, suggesting the possibility of achieving comparable performance by removing these features from the model.

In the subsequent phases of our analysis, we extended our assessments to encompass all algorithms, culminating in hard and soft voting, yielding discernible results from each algorithm which can be observed on Figure 4.3. Furthermore, feature importance plots for each data set obtained from the Random Forest algorithm can be found in appendix section F. The plots highlight, how Random Forest valued the property grouping technique over other variables when the properties were introduced.

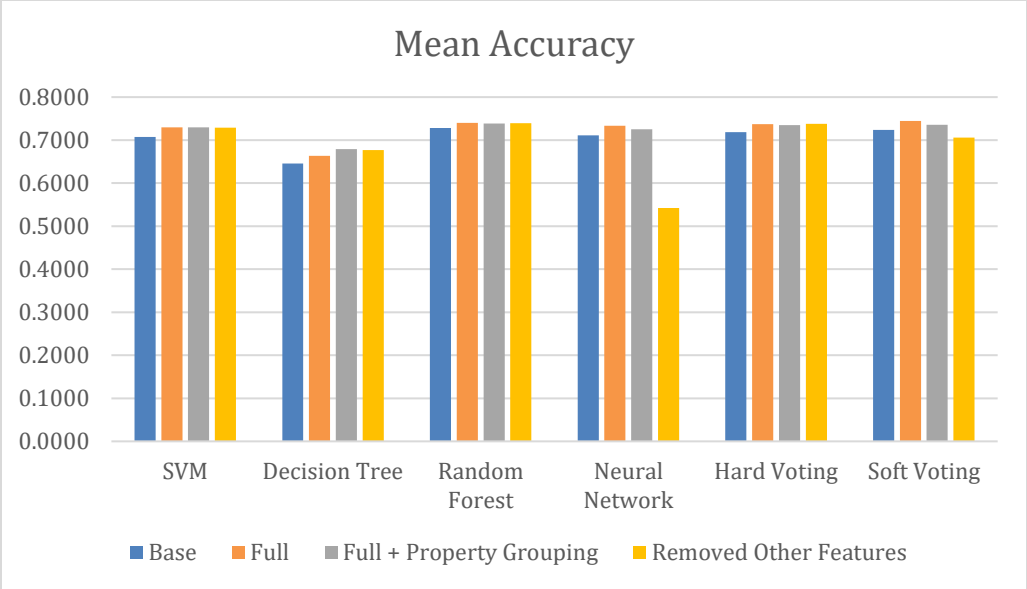


Figure 4.3: provides an overview of the mean accuracy derived from all functions across the algorithms we tested.

Given the challenge of discerning minute accuracy changes, min-max normalization was applied to the scores for each algorithm, facilitating a clearer visualization of score variations (Figure 4.4). The min-max normalization can be represented as follows: [28]

$$x_{scaled} = \frac{x - x_{min}}{x_{max} - x_{min}} \quad (4,1)$$

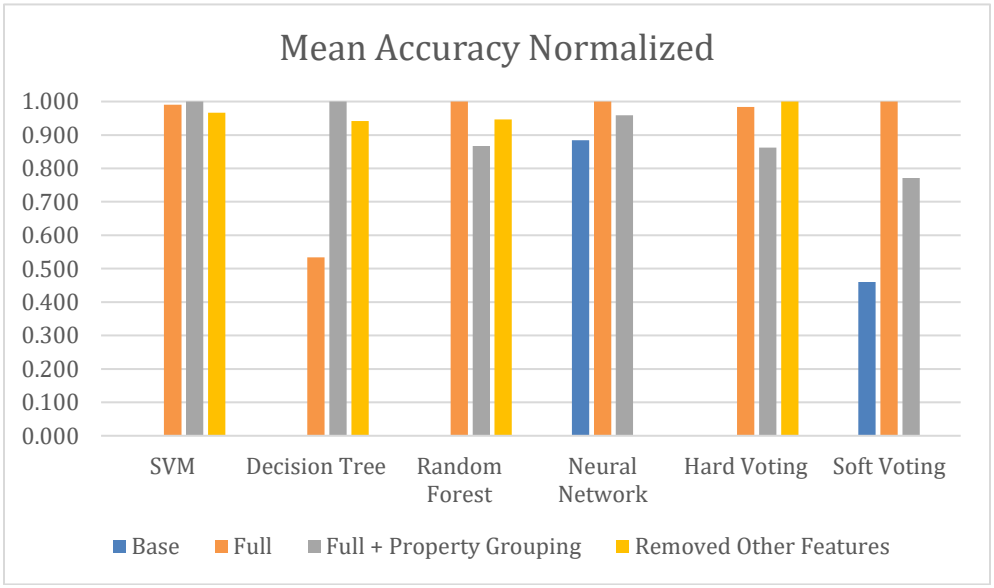


Figure 4.4: Shows the normalized scores for easier visualization of the scores. The minimum score for each algorithm has been zeroed.

Notably, SVM and Decision Tree exhibited optimal performance with the complete dataset, including the addition of property grouping. Conversely, Random Forest, Neural Network, and Soft Voting demonstrated superior performance solely with the complete data. Intriguingly, Hard Voting outperformed others when features were removed. Considering the imbalanced nature of our data, traditional accuracy metrics were complemented by F1-Scores, depicted in Figure 4.5. The normalized F1-Scores are presented in Figure 4.6.

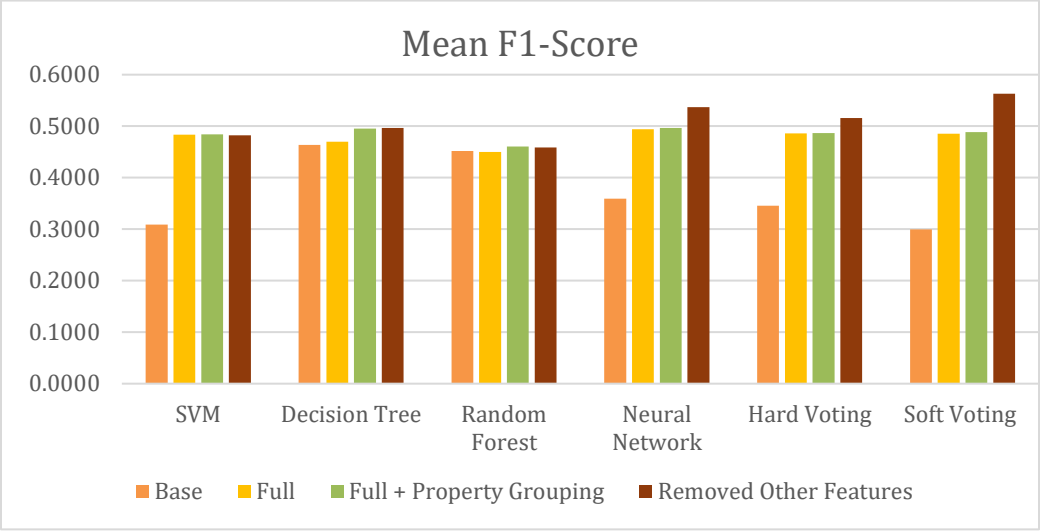


Figure 4.5: Shows the F1-Scores for all our functions from all the various algorithms that were tested.

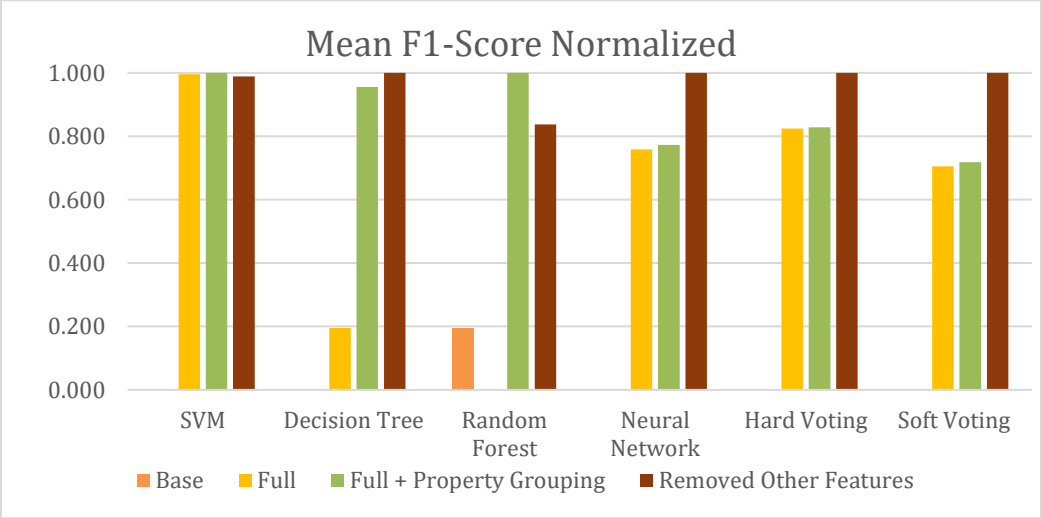


Figure 4.6: Shows the normalized scores for easier visualization. The minimum score for each algorithm has been zeroed.

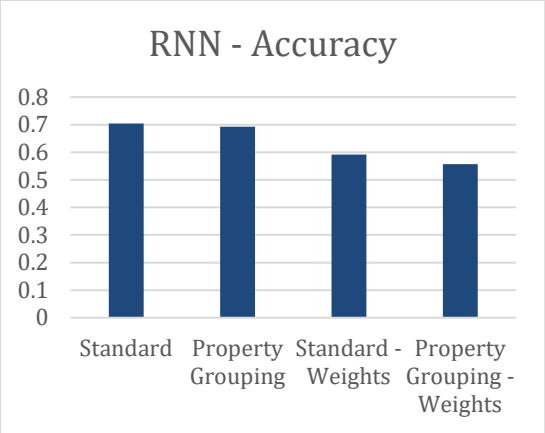
Contrary to the accuracy trends, SVM and Random Forest were the sole algorithms that exhibited improved performance with the full dataset and property grouping in terms of F1-Scores. Other algorithms exhibited a stronger preference for the removed features. These results suggest that the amino acid grouping technique may hold potential when employed with complete data and specific algorithms, though further testing is imperative for conclusive insights.

RNN and CNN

Preceding the assessment of RNN and CNN results, a comparative analysis was conducted between the amino acid property grouping encoder and the standard RNN language encoder. In the latter, each amino acid was treated as a word and tokenized to a specific number. For instance, the AILTST amino acid sequence was tokenized as [1, 8, 10, 14, 16, 17], resulting in the matrix below: [33]

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 8 \\ 0 & 0 & 0 & 1 & 8 & 10 \\ 0 & 0 & 1 & 8 & 10 & 14 \\ 0 & 1 & 8 & 10 & 14 & 16 \\ 1 & 8 & 10 & 14 & 16 & 17 \end{bmatrix}$$

Subsequently, the standard encoder was employed to evaluate the efficacy of the custom RNN encoder. Prior to application, class-weighting was introduced to augment the balance and consistency of RNN results. Figures 4.7a and 4.7b showcase the outcomes. Notably, the introduction of class weights significantly improved the F1-scores of the models. Additionally, in terms of accuracy, the property grouping encoder was surpassed. However, when focusing on F1-scores, the custom encoder exhibited unexpectedly superior performance. Given the study's reliance on the F1-score as the primary metric due to the imbalanced nature of the dataset, the decision was made to deploy the custom encoder for testing on the CNN model.



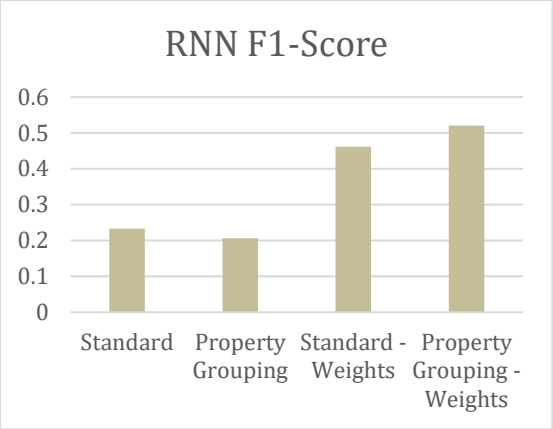


Figure 4.7a and 4.7b: Illustrates the standard encoder vs the property grouping encoder. Figure (4.7a) on the left displays the accuracy and the figure (4.7b) on the right shows the F1-Score of our models.

Upon initial execution of RNN and CNN, comparable accuracy to other tested algorithms was observed. However, F1-scores indicated suboptimal performance, potentially attributed to overgeneralization and insufficient data for neural network training. Figure 4.7 contrasts the accuracy scores of RNN and CNN against feature-embedded algorithms, showcasing competitive performance, especially when features were removed. However, the introduction of weights led to a substantial reduction in accuracy. Conversely, F1-scores exhibited an opposite trend as seen in figure 4.8, revealing an improvement with weighted classes. Furthermore, CNN consistently underperformed in comparison to RNN.

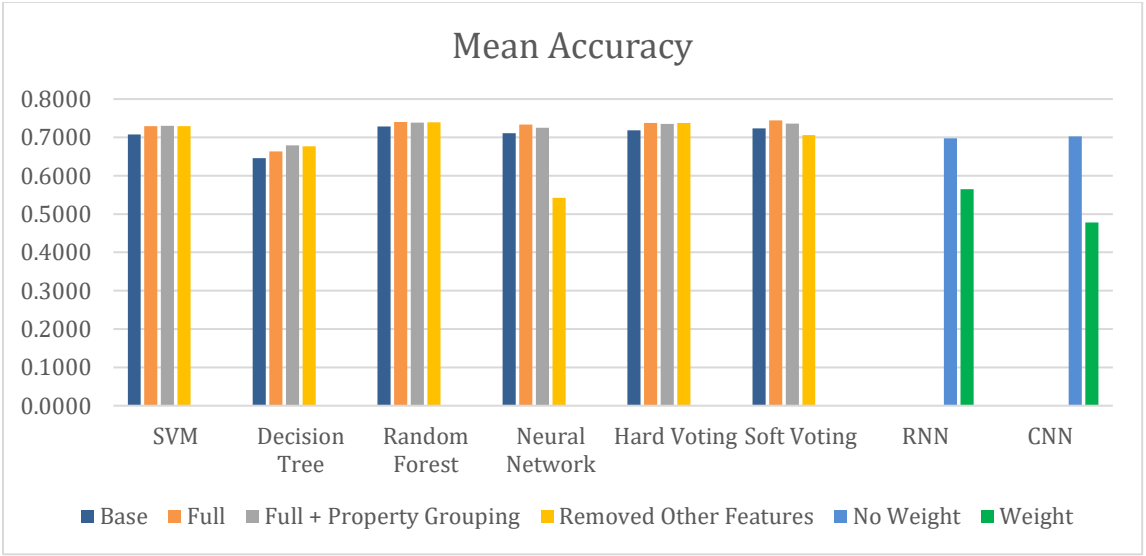


Figure 4.8: Shows the RNN and CNN accuracy Scores compared to the feature embedded algorithms.

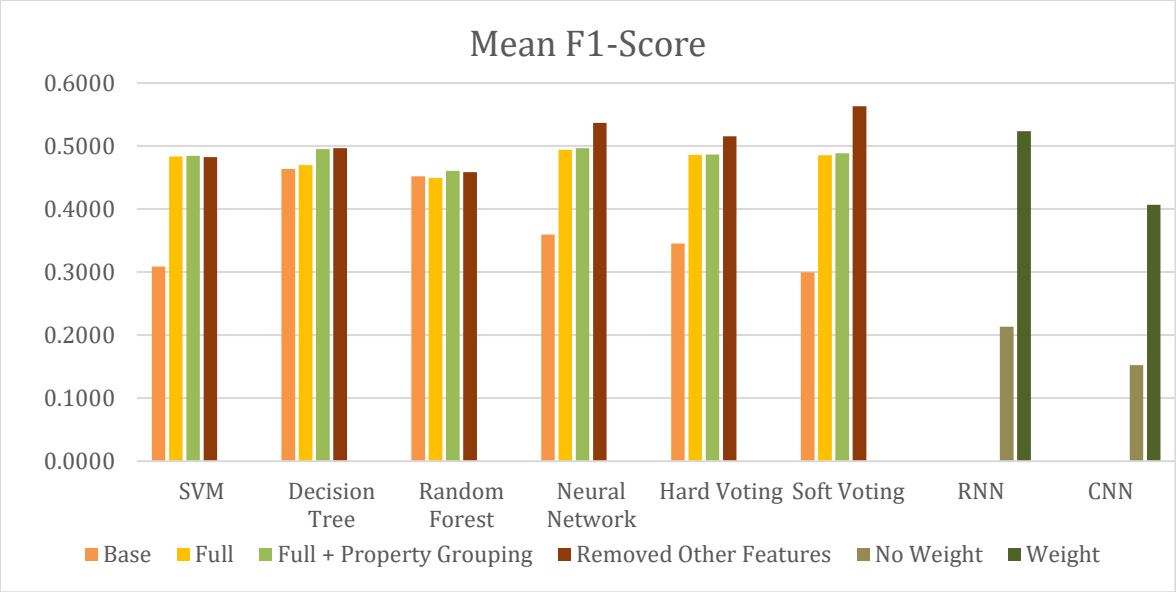


Figure 4.9: Shows the RNN and CNN F1-scores compared to the feature embedded algorithms.

The assertion that the imbalanced dataset contributed to performance disparities is supported by individual accuracy and F1-scores for each protein function. For example, Molecular function consistently achieved the highest accuracy and F1-scores across all algorithms. For instance, molecular function attained up to 0.76 accuracy and 0.86 F1-score, indicating effective prediction capabilities of a class that had the highest representation within the sampled data. Detailed individual scores and raw data can be found in Appendix section D.

Conclusions

Based on the obtained results, it appears that the amino acid property grouping technique holds promise for predicting the function of protein sequences. A crucial consideration in interpreting the lower F1-scores is the deliberate introduction of high variance in the sample data. This intentional variance ensures that each feature utilized carries significance, and the addition or removal of such features can markedly impact the scores. Despite this intentional variance, the property grouping technique demonstrated minimal loss in predictive power in certain instances and, in most other instances, resulted in better results.

Specifically, in the case of SVM, property grouping exhibited the capability to replace other features without a substantial decline in the algorithm's predictive capacity. Similar trends were observed with the decision tree algorithm, albeit random forest favored the utilization of the complete dataset with the addition of the property grouping technique. Notably however, for neural network and voting algorithms, the introduction of amino acid property grouping led to an improvement in scores.

In the context of recurrent neural networks (RNN) and convolutional neural networks (CNN), comparable F1-scores to feature-embedded algorithms were achieved when class weights were applied. Although, the property grouping method was successfully able to outperform a standard RNN encoder. Furthermore, it is essential to acknowledge that both RNN and CNN demand substantial data for optimal performance. Technical constraints such as the 500-sequence length cap and the necessity for data sampling may contribute to their relatively poorer performance. It is our contention that the observed suboptimal performance of these algorithms is not inherent to

their capabilities, as functions with higher prevalence consistently achieved superior scores compared to other functions in the dataset.

In summary, the amino acid property grouping technique showcases promise in enhancing predictive capabilities, particularly when integrated with certain algorithms. The nuanced observations across various algorithms underscore the need for a comprehensive understanding of algorithmic behavior and its interaction with feature sets in the context of protein function prediction.

Future Works

The demonstrated results substantiate the veracity of the grouping of amino acid properties. However, before asserting the significance of this property, further analyses must be conducted. Alternative sampling techniques could be applied to assess the performance of models under reduced sequence variance. Sampling strategies that ensure a balance in the distribution of functions could also be explored.

Another avenue for validation involves leveraging high-performance computation, enabling the utilization of the entire dataset rather than relying on a subset. This approach mitigates concerns about data imbalance, given the ample volume of sequences available for effective model learning.

Furthermore, recent advancements in neural network architectures have given rise to the Transformer algorithm, surpassing the capabilities of traditional models such as RNN, CNN, and DNN. The seminal work '*Attention is All You Need*', introduces the application of 3-D matrices that captures not only the sequential structure of language but also contextual information. The Transformer algorithm exhibits multidirectional processing, allowing it to focus on the context of each word individually; this process can be observed in Figure 6.1. In contrast, RNNs are limited to a singular direction as the word sequence expands within a sentence or paragraph. [33] [18]

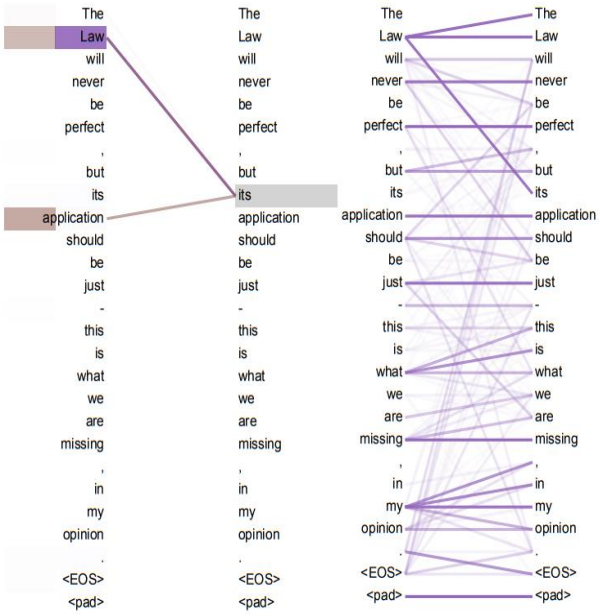


Figure 6.1: Shows how the transformer algorithm perceives language by computing the importance of each word as it relates to another. [18]

This transformative technique holds promise for protein function prediction. Several recent publications have explored the application of Transformer architectures to protein sequences, introducing unique encoders. However, none of these encoders align with the concept of Amino Acid protein grouping; should the viability of this technique be confirmed through extensive testing with various sampling methodologies or through the utilization of high-performance computing, a specialized encoder grounded in the idea of grouping amino acids based on their properties could be integrated into a Transformer encoder by furthering the use of such technique in the field.

Moreover, the technique could be further refined by testing variations in the property-based binning process of the amino acids. As mentioned in the methodology chapter, the amino acids can't always be cleanly binned into their respective properties. To improve our predictions, we could test changing the way certain acids were binned. However, this level of refinement could be employed much later in the research.

Appendix

Appendix A

Table A-1

Name	Molecular Formula	Residue Weight	pKa
Alanine (Ala/A)	$C_3H_7NO_2$	71.08	2.34
Arginine (Arg/R)	$C_6H_{14}N_4O_2$	156.19	2.17
Asparagine (Asn/N)	$C_4H_8N_2O_3$	114.11	2.02
Aspartic acid (Asp/D)	$C_4H_7NO_4$	115.09	1.88
Cysteine (Cys/C)	$C_3H_7NO_2S$	103.15	1.96
Glutamic acid (Glu/E)	$C_5H_9NO_4$	129.12	2.19
Glutamine (Gln/Q)	$C_5H_{10}N_2O_3$	128.13	2.17
Glycine (Gly/G)	$C_2H_5NO_2$	57.05	2.34
Histidine (His/H)	$C_6H_9N_3O_2$	137.14	1.82
Hydroxyproline(Hyp/O)	$C_5H_9NO_3$	113.11	1.82
Isoleucine (Ile/I)	$C_6H_{13}NO_2$	113.16	2.36
Leucine (Leu/L)	$C_6H_{13}NO_2$	113.16	2.36
Lysine (Lys/K)	$C_6H_{14}N_2O_2$	128.18	2.18
Methionine (Met/M)	$C_5H_{11}NO_2S$	131.20	2.28
Phenylalanine (Phe/F)	$C_9H_{11}NO_2$	147.18	1.83
Proline (Pro/P)	$C_5H_9NO_2$	97.12	1.99
Pyroglutamatic (Glp/U)	$C_5H_7NO_3$	121.09	-
Serine (Ser/S)	$C_3H_7NO_3$	87.08	2.21
Threonine (Thr/T)	$C_4H_9NO_3$	101.11	2.09
Tryptophan (Trp/W)	$C_{11}H_{12}N_2O_2$	186.22	2.83

Tyrosine (Tyr/Y)	$C_9H_{11}NO_3$	163.18	2.20
Valine (Val/V)	$C_5H_{11}NO_2$	99.13	2.32

Table A-1: Shows the 22 different amino acids and their molecular formula, as well as the approximate weights of the residues and finally the pKa values. [3]

Table A-2

Residue Type	**Hydrophobicity**
Ile	4.5
Val	4.2
Leu	3.8
Phe	2.8
Cys	2.5
Met	1.9
Ala	1.8
Gly	-0.4
Thr	-0.7
Ser	-0.8
Trp	-0.9
Tyr	-1.3
Pro	-1.6
His	-3.2
Glu	-3.5
Gln	-3.5
Asp	-3.5
Asn	-3.5
Lys	-3.9
Arg	-4.5

Table A-2: Hydrophobicity of amino acids within a sequence. [31]

Appendix B

Figure B-1

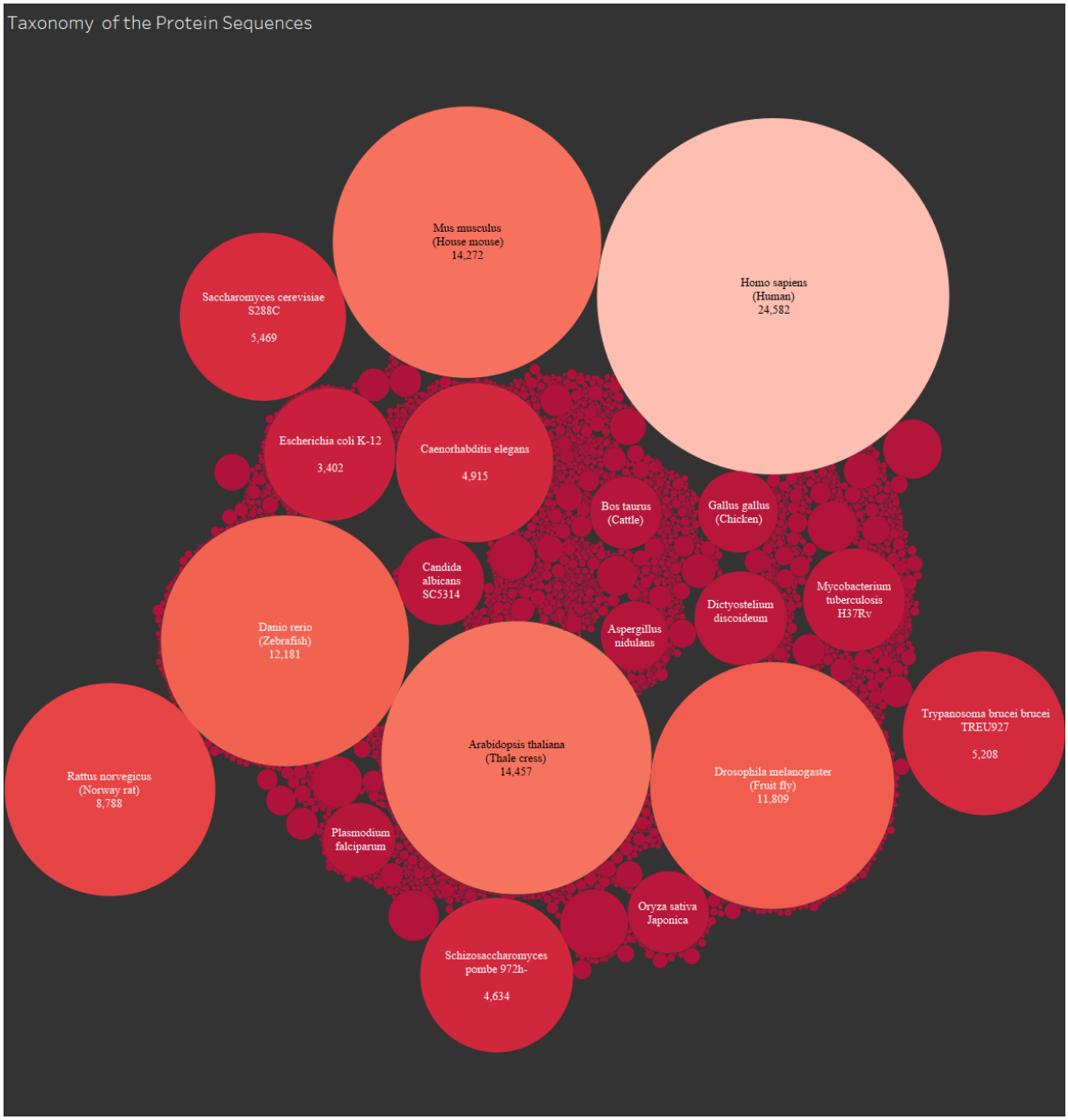


Figure B-1: Presents the magnitude of the various taxonomies within the dataset. Overall, 3131 different taxonomies were within the dataset. Most sequences seem to belong to humans.

Figure B-2

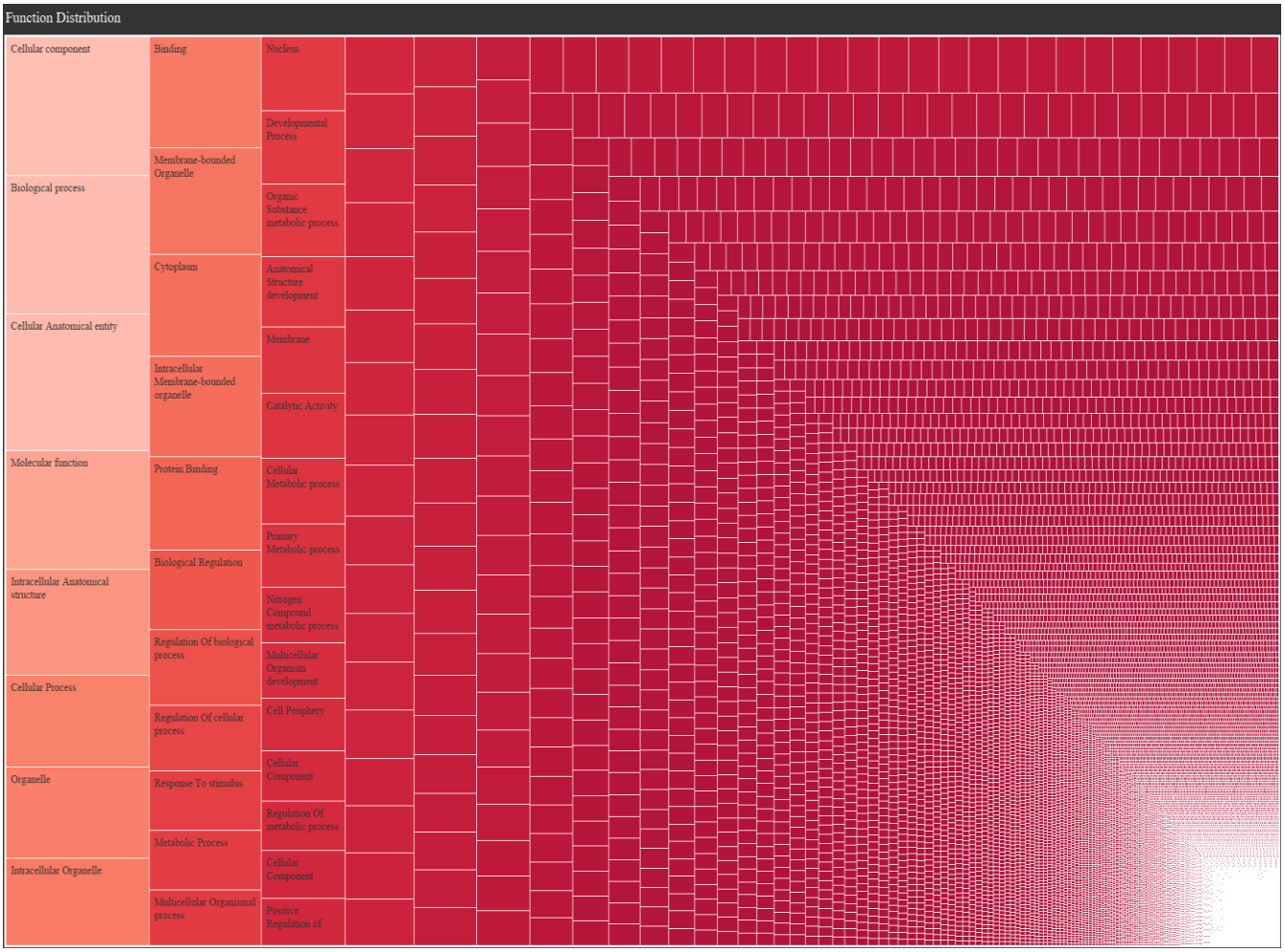


Figure B-2: Shows the magnitude of various functions within the dataset. There are 43,248 unique functions. We can observe that our dataset is highly imbalanced; such that a large majority of proteins favor a particular function. The sampled data should encapsulate most of the top ten functions within CAFA5 dataset.

Figure B-3

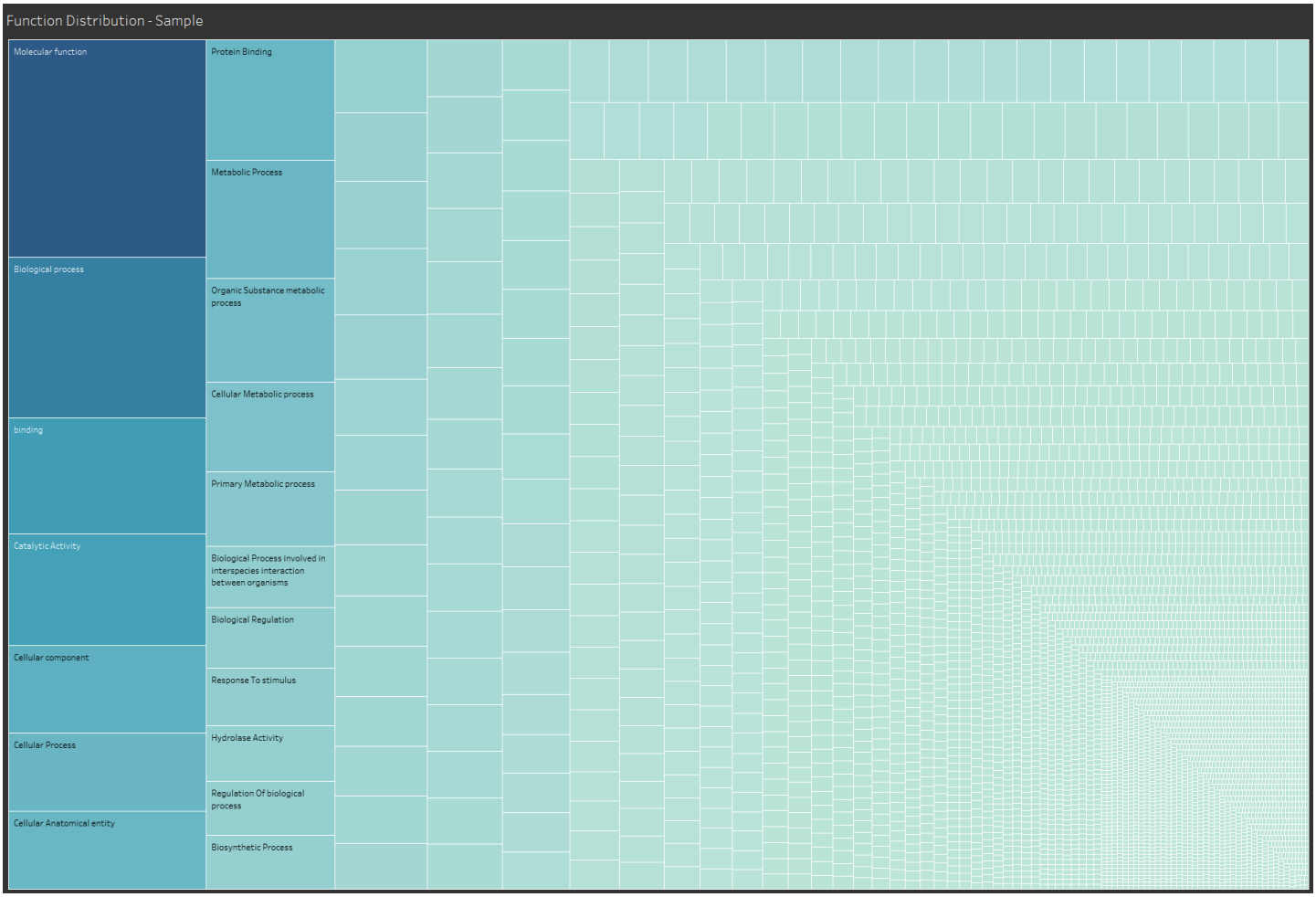


Figure B-3: shows the magnitude of various functions within the sample data. Observations can be made regarding the highly imbalanced nature of the sampled data; such that a single function represents a majority of the sampled data. The same top ten functions that were of note in the CAFA5 dataset are also present in high volumes within the sampled data.

Figure B-4

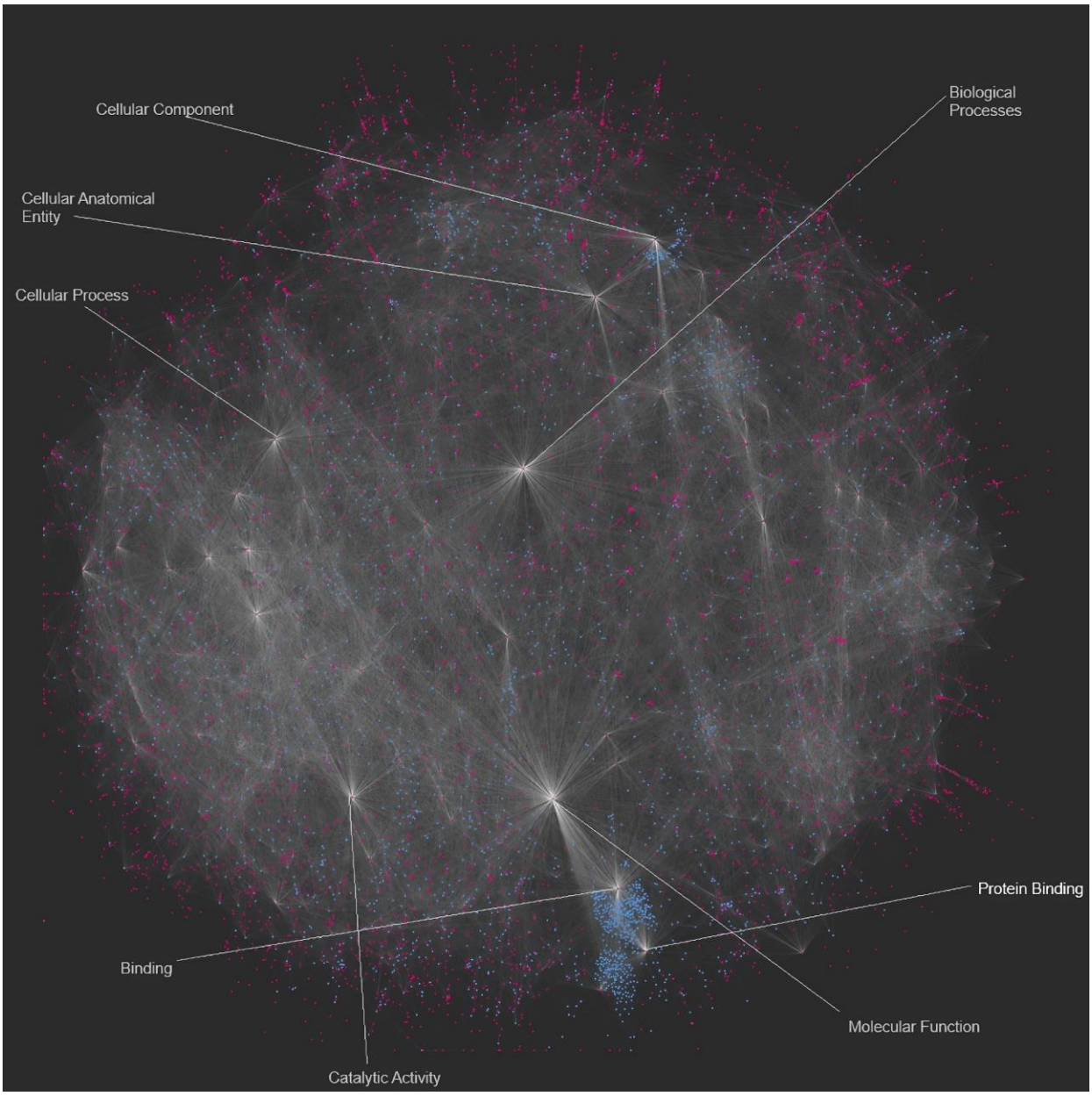


Figure B-4: A Bipartite network between proteins (in blue) and their respective functions (red) to show the complexity of our sample data. Nodes with high connectivity have also been labeled. We can observe that labels correspond to the top 10 functions.

Appendix C

Figure C-1

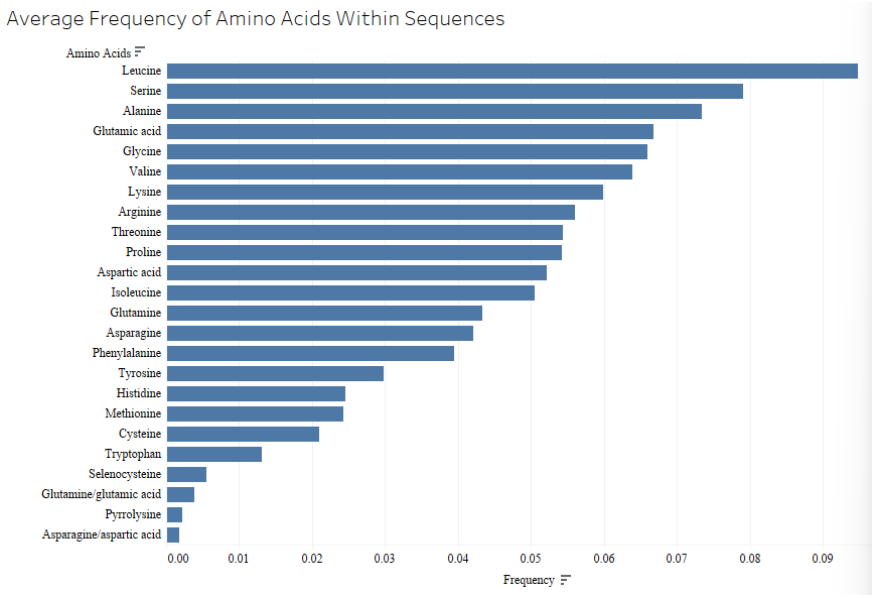


Figure C-1: The mean frequency distribution of individual amino acids within all 142,246 sequences

Figure C-2

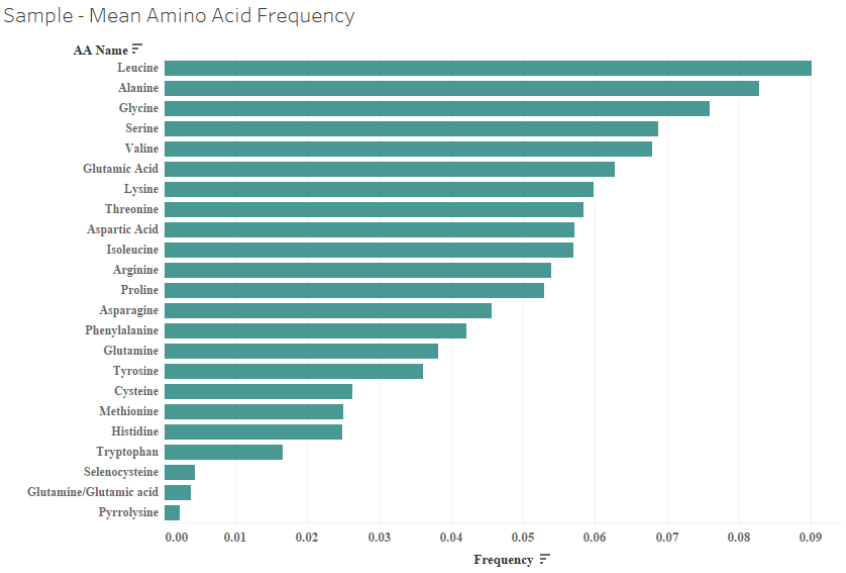


Figure C-2: The mean frequency distribution of individual amino acids within the 3131 sequences. The mean frequency is approximately similar to the original dataset.

Figure C-3

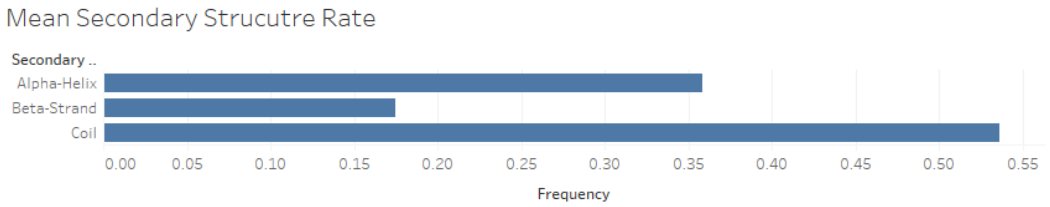


Figure C-3: The mean frequency distribution of the secondary structures in the sampled data.

Appendix D

Table D-1

SVM			
Base			
Go-Ids	go_function	F1-Score	Accuracy
GO:0003674	molecular_function	0.8524	0.7457
GO:0008150	biological_process	0.6631	0.5319
GO:0003824	catalytic activity	0.6451	0.7436
GO:0008152	metabolic process	0.2341	0.7564
GO:0071704	organic substance metabolic process	0.1862	0.7862
GO:0005488	binding	0.1663	0.5947
GO:0009987	cellular process	0.1418	0.7426
GO:0005515	protein binding	0.1375	0.7330
GO:0110165	cellular anatomical entity	0.0392	0.7394
GO:0005575	cellular_component	0.0211	0.7032
	Mean	0.3087	0.7077
Full Data			
Go-Ids	go_function	F1-Score	Accuracy
GO:0003674	molecular_function	0.8386	0.7457
GO:0008150	biological_process	0.6464	0.5415
GO:0003824	catalytic activity	0.5582	0.7457
GO:0008152	metabolic process	0.5043	0.8160
GO:0071704	organic substance metabolic process	0.4921	0.8287
GO:0005488	binding	0.3910	0.6553
GO:0009987	cellular process	0.3908	0.7745
GO:0005575	cellular_component	0.3758	0.6926
GO:0110165	cellular anatomical entity	0.3529	0.7426
GO:0005515	protein binding	0.2848	0.7543
	Mean	0.4835	0.7297
Full Data + Protein Property Grouping			
Go-Ids	go_function	F1-Score	Accuracy
GO:0003674	molecular_function	0.8421	0.7511
GO:0008150	biological_process	0.6464	0.5415
GO:0003824	catalytic activity	0.5609	0.7468
GO:0008152	metabolic process	0.5043	0.8160
GO:0071704	organic substance metabolic process	0.4921	0.8287
GO:0005488	binding	0.4022	0.6585
GO:0009987	cellular process	0.3886	0.7723
GO:0005575	cellular_component	0.3652	0.6894
GO:0110165	cellular anatomical entity	0.3511	0.7404
GO:0005515	protein binding	0.2892	0.7543

	Mean	0.4842	0.7299
Removed Secondary Sequence, Hydrophobicity,and PKA			
Go-Ids	go_function	F1-Score	Accuracy
GO:0003674	molecular_function	0.8442	0.7543
GO:0008150	biological_process	0.6478	0.5362
GO:0003824	catalytic activity	0.5582	0.7457
GO:0008152	metabolic process	0.5043	0.8160
GO:0071704	organic substance metabolic process	0.4921	0.8287
GO:0005488	binding	0.3993	0.6574
GO:0009987	cellular process	0.3908	0.7745
GO:0005575	cellular_component	0.3656	0.6862
GO:0110165	cellular anatomical entity	0.3351	0.7383
GO:0005515	protein binding	0.2848	0.7543
	Mean	0.4822	0.7291

Table D-1: Shows the F1 and accuracy scores for each function as well as the cumulative mean scores produced by the predictions made from the SVM

Table D-2

Decision Tree			
Base			
Go-Ids	go_function	F1-Score	Accuracy
GO:0003674	molecular_function	0.7685	0.6564
GO:0003824	catalytic activity	0.5502	0.6713
GO:0008150	biological_process	0.5392	0.5309
GO:0005488	binding	0.4691	0.5883
GO:0008152	metabolic process	0.4587	0.7213
GO:0071704	organic substance metabolic process	0.3982	0.7074
GO:0005515	protein binding	0.3906	0.6681
GO:0009987	cellular process	0.3660	0.6426
GO:0005575	cellular_component	0.3574	0.6213
GO:0110165	cellular anatomical entity	0.3360	0.6511
	Mean	0.4634	0.6459
Full Data			
Go-Ids	go_function	F1-Score	Accuracy
GO:0003674	molecular_function	0.7881	0.6819
GO:0003824	catalytic activity	0.6033	0.6936
GO:0008150	biological_process	0.5519	0.5319
GO:0005488	binding	0.4941	0.5926
GO:0008152	metabolic process	0.4622	0.7351
GO:0005575	cellular_component	0.3827	0.6500
GO:0071704	organic substance metabolic process	0.3688	0.7415

GO:0110165	cellular anatomical entity	0.3597	0.6819
GO:0009987	cellular process	0.3471	0.6638
GO:0005515	protein binding	0.3410	0.6628
	Mean	0.4699	0.6635
Full Data + Protein Property Grouping			
Go-Ids	go_function	F1-Score	Accuracy
GO:0003674	molecular_function	0.7929	0.6894
GO:0003824	catalytic activity	0.6097	0.7085
GO:0008150	biological_process	0.5796	0.5617
GO:0005488	binding	0.5073	0.6074
GO:0008152	metabolic process	0.4395	0.7340
GO:0071704	organic substance metabolic process	0.4389	0.7606
GO:0110165	cellular anatomical entity	0.4126	0.7032
GO:0005575	cellular_component	0.4060	0.6638
GO:0005515	protein binding	0.3934	0.6851
GO:0009987	cellular process	0.3711	0.6755
	Mean	0.4951	0.6789
Removed Secondary Sequence, Hydrophobicity,and PKA			
Go-Ids	go_function	F1-Score	Accuracy
GO:0003674	molecular_function	0.784397	0.676596
GO:0003824	catalytic activity	0.618132	0.704255
GO:0008150	biological_process	0.553942	0.542553
GO:0005488	binding	0.541114	0.631915
GO:0071704	organic substance metabolic process	0.447301	0.771277
GO:0110165	cellular anatomical entity	0.423729	0.710638
GO:0008152	metabolic process	0.418103	0.712766
GO:0005515	protein binding	0.411067	0.682979
GO:0005575	cellular_component	0.407619	0.669149
GO:0009987	cellular process	0.360656	0.668085
	Mean	0.4966	0.6770

Table D-2: Shows the F1 and accuracy scores for each function as well as the cumulative mean scores produced by the predictions made from the Decision Tree

Table D-3

Random Forest			
Base			
Go-Ids	go_function	F1-Score	Accuracy
GO:0003674	molecular_function	0.8439	0.7457
GO:0003824	catalytic activity	0.6765	0.7660
GO:0008150	biological_process	0.6200	0.5787
GO:0005488	binding	0.4306	0.6511

GO:0008152	metabolic process	0.3771	0.7681
GO:0110165	cellular anatomical entity	0.3673	0.7691
GO:0071704	organic substance metabolic process	0.3551	0.8106
GO:0005575	cellular_component	0.3485	0.7255
GO:0009987	cellular process	0.2936	0.7543
GO:0005515	protein binding	0.2803	0.7596
	Mean	0.4593	0.7329

Full Data

Go-Ids	go_function	F1-Score	Accuracy
GO:0003674	molecular_function	0.8501	0.7553
GO:0003824	catalytic activity	0.6894	0.7872
GO:0008150	biological_process	0.6377	0.5660
GO:0110165	cellular anatomical entity	0.4257	0.7904
GO:0008152	metabolic process	0.3975	0.7968
GO:0005575	cellular_component	0.3797	0.7394
GO:0005488	binding	0.3532	0.6415
GO:0071704	organic substance metabolic process	0.3281	0.8170
GO:0009987	cellular process	0.2305	0.7585
GO:0005515	protein binding	0.2267	0.7532
	Mean	0.4519	0.7405

Full Data + Protein Property Grouping

Go-Ids	go_function	F1-Score	Accuracy
GO:0003674	molecular_function	0.8519	0.7585
GO:0003824	catalytic activity	0.6583	0.7691
GO:0008150	biological_process	0.6429	0.5851
GO:0008152	metabolic process	0.4444	0.8032
GO:0110165	cellular anatomical entity	0.4164	0.7883
GO:0005575	cellular_component	0.3878	0.7447
GO:0005488	binding	0.3790	0.6479
GO:0071704	organic substance metabolic process	0.3650	0.8223
GO:0005515	protein binding	0.2313	0.7596
GO:0009987	cellular process	0.2000	0.7447
	Mean	0.4577	0.7423

Removed Secondary Sequence, Hydrophobicity,and PKA

Go-Ids	go_function	F1-Score	Accuracy
GO:0003674	molecular_function	0.8501	0.7553
GO:0003824	catalytic activity	0.6583	0.7691
GO:0008150	biological_process	0.6386	0.5713
GO:0110165	cellular anatomical entity	0.4024	0.7851
GO:0005575	cellular_component	0.3832	0.7500
GO:0008152	metabolic process	0.3824	0.7904

GO:0005488	binding	0.3665	0.6543
GO:0071704	organic substance metabolic process	0.3059	0.8117
GO:0005515	protein binding	0.2177	0.7553
GO:0009987	cellular process	0.2069	0.7553
	Mean	0.4412	0.7398

Table D-3: Shows the F1 and accuracy scores for each function as well as the cumulative mean scores produced by the predictions made from the Random Forest

Table D-4

DNN			
Base			
Go-Ids	go_function	F1-Score	Accuracy
GO:0003674	molecular_function	0.8580	0.7574
GO:0008150	biological_process	0.6571	0.5436
GO:0003824	catalytic activity	0.5732	0.7085
GO:0110165	cellular anatomical entity	0.3364	0.7734
GO:0005575	cellular_component	0.3306	0.7372
GO:0008152	metabolic process	0.2824	0.7404
GO:0071704	organic substance metabolic process	0.1992	0.7862
GO:0005488	binding	0.1975	0.5936
GO:0005515	protein binding	0.1351	0.7277
GO:0009987	cellular process	0.0242	0.7426
	Mean	0.3594	0.7111
Full Data			
Go-Ids	go_function	F1-Score	Accuracy
GO:0003674	molecular_function	0.8462	0.7532
GO:0003824	catalytic activity	0.5872	0.7532
GO:0008150	biological_process	0.5230	0.6138
GO:0008152	metabolic process	0.5128	0.7979
GO:0071704	organic substance metabolic process	0.5027	0.8043
GO:0005575	cellular_component	0.4227	0.7181
GO:0110165	cellular anatomical entity	0.4105	0.7372
GO:0009987	cellular process	0.4022	0.7660
GO:0005488	binding	0.3935	0.6426
GO:0005515	protein binding	0.3380	0.7457
	Mean	0.4939	0.7332
Full Data + Protein Property Grouping			
Go-Ids	go_function	F1-Score	Accuracy
GO:0003674	molecular_function	0.8457	0.7543
GO:0008150	biological_process	0.6383	0.5202

GO:0003824	catalytic activity	0.5821	0.7511
GO:0008152	metabolic process	0.5271	0.8053
GO:0071704	organic substance metabolic process	0.5154	0.8160
GO:0005488	binding	0.4077	0.6415
GO:0009987	cellular process	0.3852	0.7521
GO:0005575	cellular_component	0.3765	0.7181
GO:0110165	cellular anatomical entity	0.3622	0.7489
GO:0005515	protein binding	0.3239	0.7468
	Mean	0.4964	0.7254
	Removed Secondary Sequence, Hydrophobicity,and PKA		
Go-Ids	go_function	F1-Score	Accuracy
GO:0003674	molecular_function	0.8391	0.7479
GO:0005488	binding	0.6384	0.6011
GO:0008150	biological_process	0.6319	0.5266
GO:0003824	catalytic activity	0.6297	0.5947
GO:0005515	protein binding	0.5006	0.5691
GO:0005575	cellular_component	0.4561	0.4596
GO:0008152	metabolic process	0.4410	0.5011
GO:0009987	cellular process	0.4307	0.5021
GO:0071704	organic substance metabolic process	0.4010	0.4883
GO:0110165	cellular anatomical entity	0.3991	0.4362
	Mean	0.5368	0.5427

Table D-4: Shows the F1 and accuracy scores for each function as well as the cumulative mean scores produced by the predictions made from the DNN

Table D-5

Hard Voting			
Base			
Go-Ids	go_function	F1-Score	Accuracy
GO:0003674	molecular_function	0.8570	0.7553
GO:0008150	biological_process	0.6625	0.5436
GO:0003824	catalytic activity	0.6445	0.7500
GO:0005575	cellular_component	0.2682	0.7330
GO:0110165	cellular anatomical entity	0.2680	0.7734
GO:0008152	metabolic process	0.2450	0.7574
GO:0005488	binding	0.1857	0.5989
GO:0071704	organic substance metabolic process	0.1478	0.7915
GO:0005515	protein binding	0.1139	0.7351
GO:0009987	cellular process	0.0625	0.7447
	Mean	0.3455	0.7183

Full Data			
Go-Ids	go_function	F1-Score	Accuracy
GO:0003674	molecular_function	0.8517	0.7606
GO:0008150	biological_process	0.6324	0.5734
GO:0003824	catalytic activity	0.5714	0.7511
GO:0008152	metabolic process	0.5000	0.8128
GO:0071704	organic substance metabolic process	0.4984	0.8287
GO:0009987	cellular process	0.3815	0.7723
GO:0110165	cellular anatomical entity	0.3777	0.7511
GO:0005575	cellular_component	0.3747	0.7160
GO:0005488	binding	0.3733	0.6500
GO:0005515	protein binding	0.2963	0.7574
	Mean	0.4857	0.7373
Full Data + Protein Property Grouping			
Go-Ids	go_function	F1-Score	Accuracy
GO:0003674	molecular_function	0.8524	0.7617
GO:0008150	biological_process	0.6451	0.5340
GO:0003824	catalytic activity	0.5719	0.7500
GO:0008152	metabolic process	0.5085	0.8149
GO:0071704	organic substance metabolic process	0.4938	0.8277
GO:0005488	binding	0.3872	0.6532
GO:0009987	cellular process	0.3768	0.7713
GO:0005575	cellular_component	0.3741	0.7223
GO:0110165	cellular anatomical entity	0.3631	0.7574
GO:0005515	protein binding	0.2919	0.7574
	Mean	0.4865	0.7350
Removed Secondary Sequence, Hydrophobicity,and PKA			
Go-Ids	go_function	F1-Score	Accuracy
GO:0003674	molecular_function	0.8543	0.7660
GO:0003824	catalytic activity	0.7113	0.7936
GO:0008150	biological_process	0.6466	0.5383
GO:0008152	metabolic process	0.5459	0.8213
GO:0071704	organic substance metabolic process	0.4856	0.8287
GO:0005488	binding	0.4542	0.6702
GO:0005575	cellular_component	0.4025	0.6968
GO:0009987	cellular process	0.3771	0.7681
GO:0110165	cellular anatomical entity	0.3756	0.7383
GO:0005515	protein binding	0.3030	0.7553
	Mean	0.5156	0.7377

Table D-5: Shows the F1 and accuracy scores for each function as well as the cumulative mean scores produced by the predictions made from the Hard Voting

Table D-6

Soft Voting			
Base			
Go-Ids	go_function	F1-Score	Accuracy
GO:0003674	molecular_function	0.8649	0.7670
GO:0008150	biological_process	0.6568	0.5564
GO:0003824	catalytic activity	0.5829	0.7457
GO:0005575	cellular_component	0.2273	0.7468
GO:0005488	binding	0.2022	0.6138
GO:0110165	cellular anatomical entity	0.1916	0.7755
GO:0008152	metabolic process	0.1275	0.7670
GO:0005515	protein binding	0.0746	0.7362
GO:0071704	organic substance metabolic process	0.0385	0.7872
GO:0009987	cellular process	0.0320	0.7426
	Mean	0.2998	0.7238
Full Data			
Go-Ids	go_function	F1-Score	Accuracy
GO:0003674	molecular_function	0.8553	0.7638
GO:0003824	catalytic activity	0.5863	0.7553
GO:0008150	biological_process	0.5512	0.6223
GO:0008152	metabolic process	0.5097	0.8117
GO:0071704	organic substance metabolic process	0.5076	0.8287
GO:0009987	cellular process	0.3933	0.7702
GO:0005575	cellular_component	0.3874	0.7309
GO:0110165	cellular anatomical entity	0.3804	0.7574
GO:0005488	binding	0.3774	0.6489
GO:0005515	protein binding	0.3049	0.7574
	Mean	0.4853	0.7447
Full Data + Protein Property Grouping			
Go-Ids	go_function	F1-Score	Accuracy
GO:0003674	molecular_function	0.8558	0.7660
GO:0008150	biological_process	0.6490	0.5340
GO:0003824	catalytic activity	0.5848	0.7553
GO:0008152	metabolic process	0.5152	0.8138
GO:0071704	organic substance metabolic process	0.5106	0.8287
GO:0009987	cellular process	0.3864	0.7702
GO:0005488	binding	0.3783	0.6468
GO:0005575	cellular_component	0.3550	0.7255
GO:0110165	cellular anatomical entity	0.3506	0.7596
GO:0005515	protein binding	0.3015	0.7585

	Mean	0.4887	0.7359
Removed Secondary Sequence, Hydrophobicity,and PKA			
Go-Ids	go_function	F1-Score	Accuracy
GO:0003674	molecular_function	0.8592	0.7723
GO:0003824	catalytic activity	0.6967	0.7202
GO:0005488	binding	0.6443	0.6277
GO:0008150	biological_process	0.6433	0.5340
GO:0008152	metabolic process	0.5665	0.7851
GO:0071704	organic substance metabolic process	0.5100	0.8181
GO:0005515	protein binding	0.4539	0.7543
GO:0005575	cellular_component	0.4372	0.6330
GO:0009987	cellular process	0.4213	0.7223
GO:0110165	cellular anatomical entity	0.3975	0.6936
	Mean	0.5630	0.7061

Table D-6: Shows the F1 and accuracy scores for each function as well as the cumulative mean scores produced by the predictions made from the Soft Voting

Table D-7

RNN			
No weights			
Go-Ids	go_function	F1-Score	Accuracy
GO:0003674	molecular_function	0.8391	0.7352
GO:0008150	biological_process	0.6707	0.5662
GO:0005575	cellular_component	0.3254	0.7289
GO:0110165	cellular anatomical entity	0.2911	0.7592
GO:0005488	binding	0.0084	0.6252
GO:0071704	organic substance metabolic process	0.0000	0.7671
GO:0005515	protein binding	0.0000	0.7560
GO:0009987	cellular process	0.0000	0.7193
GO:0008152	metabolic process	0.0000	0.7145
GO:0003824	catalytic activity	0.0000	0.6077
	Mean	0.2135	0.6979
Weights			
Go-Ids	go_function	F1-Score	Accuracy
GO:0003674	molecular_function	0.7644	0.6746
GO:0008150	biological_process	0.6642	0.5662
GO:0003824	catalytic activity	0.6422	0.6268
GO:0005488	binding	0.5505	0.4322
GO:0008152	metabolic process	0.5154	0.5470
GO:0009987	cellular process	0.4736	0.5072

GO:0071704	organic substance metabolic process	0.4557	0.5199
GO:0005515	protein binding	0.4034	0.3301
GO:0005575	cellular_component	0.3922	0.7033
GO:0110165	cellular anatomical entity	0.3740	0.7384
	Mean	0.5236	0.5646

Table D-7: Shows the F1 and accuracy scores for each function as well as the cumulative mean scores produced by the predictions made from the RNN.

Table D-8

RNN - Standard Encoder			
No weights			
Go-Ids	go_function	F1-Score	Accuracy
GO:0003674	molecular_function	0.8519	0.7560
GO:0008150	biological_process	0.6554	0.5455
GO:0003824	catalytic activity	0.3090	0.6220
GO:0110165	cellular anatomical entity	0.2857	0.7608
GO:0005575	cellular_component	0.2066	0.7305
GO:0071704	organic substance metabolic process	0.0265	0.7656
GO:0005515	protein binding	0.0000	0.7719
GO:0008152	metabolic process	0.0000	0.7352
GO:0009987	cellular process	0.0000	0.7129
GO:0005488	binding	0.0000	0.6459
	Mean	0.2335	0.7046
Weights			
Go-Ids	go_function	F1-Score	Accuracy
GO:0003674	molecular_function	0.7834	0.6842
GO:0003824	catalytic activity	0.5806	0.6268
GO:0008152	metabolic process	0.4876	0.6045
GO:0071704	organic substance metabolic process	0.4706	0.5550
GO:0005488	binding	0.4486	0.4785
GO:0009987	cellular process	0.4228	0.5470
GO:0008150	biological_process	0.3932	0.4880
GO:0110165	cellular anatomical entity	0.3599	0.7049
GO:0005575	cellular_component	0.3399	0.6778
GO:0005515	protein binding	0.3309	0.5550
	Mean	0.4617	0.5922

Table D-8: Shows the F1 and accuracy scores for each function as well as the cumulative mean scores produced by the predictions made from the RNN- Standard encoder.

Table D-9

CNN			
No weights			
Go-Ids	go_function	F1-Score	Accuracy
GO:0003674	molecular_function	0.8189	0.6933
GO:0008150	biological_process	0.7031	0.5421
GO:0071704	organic substance metabolic process	0.0000	0.7905
GO:0005515	protein binding	0.0000	0.7754
GO:0009987	cellular process	0.0000	0.7581
GO:0008152	metabolic process	0.0000	0.7559
GO:0110165	cellular anatomical entity	0.0000	0.7343
GO:0005575	cellular_component	0.0000	0.7041
GO:0003824	catalytic activity	0.0000	0.6458
GO:0005488	binding	0.0000	0.6264
	Mean	0.1522	0.7026
weights			
Go-Ids	go_function	F1-Score	Accuracy
GO:0003674	molecular_function	0.6837	0.5724
GO:0008150	biological_process	0.5150	0.4752
GO:0003824	catalytic activity	0.4701	0.4838
GO:0005488	binding	0.4577	0.4881
GO:0005575	cellular_component	0.3827	0.4773
GO:0110165	cellular anatomical entity	0.3351	0.4600
GO:0005515	protein binding	0.3233	0.4665
GO:0008152	metabolic process	0.3204	0.4687
GO:0009987	cellular process	0.2989	0.4428
GO:0071704	organic substance metabolic process	0.2817	0.4492
	Mean	0.4069	0.4784

Table D-9: Shows the F1 and accuracy scores for each function as well as the cumulative mean scores produced by the predictions made from CNN.

Appendix E

Polynomial Kernel SVM

```
poly_kernel_svm_clf_1 = Pipeline([("scaler", StandardScaler()),
                          ("svm_clf", SVC(kernel="poly",degree=3, coef0=1,C=5))])
```

Decision Tree

```
tree_clf_1 = DecisionTreeClassifier(random_state=42)
```

Random Forest

```
RandomForestClassifier(n_estimators=100, random_state=42)
```

Deep Neural Network

```
nn_1 = MLPClassifier(solver='adam',activation='logistic', alpha=0.001,
              hidden_layer_sizes=(100,50,25), max_iter=50, random_state=42)
```

Hard Voting

```
voting_classifier_hard_1 = VotingClassifier(estimators=[
    ('svc', SVC(kernel="poly", degree=3, coef0=1, C=5)),
    ('rf', RandomForestClassifier(n_estimators=100, random_state=42)),
    ('nn', MLPClassifier(solver='adam', activation='logistic', alpha=0.001,
hidden_layer_sizes=(100, 50, 25), max_iter=50, random_state=42))
], voting='hard')
```

Soft Voting

```
voting_classifier_soft_1 = VotingClassifier(estimators=[
    ('svc', SVC(kernel="poly", degree=3, coef0=1, C=5,probability=True)),
    ('rf', RandomForestClassifier(n_estimators=100, random_state=42)),
```

```
    ('nn', MLPClassifier(solver='adam', activation='logistic', alpha=0.001,
hidden_layer_sizes=(100, 50, 25), max_iter=50, random_state=42))

], voting='soft')
```

Recurrent Neural Network

```
model = Sequential()

model.add(Bidirectional(LSTM(24,
input_shape=(new_X.shape[1:]),activation='tanh',return_sequences=True)))

model.add(Dropout(0.5))

model.add(LSTM(32,activation='tanh'))

model.add(Dropout(0.5))

model.add(Dense(16,activation='tanh'))

model.add(Dense(1,activation='sigmoid'))

model.compile(loss='binary_crossentropy',
optimizer='adam',metrics=[tf.keras.metrics.BinaryAccuracy()])
```

Convolution Neural Network

```
model = models.Sequential()

model.add(layers.Conv2D(28, (4, 4), activation='tanh', input_shape=(28, 28,
4)))

model.add(layers.MaxPooling2D((2, 2)))

model.add(layers.Conv2D(64, (4, 4), activation='tanh'))

model.add(layers.MaxPooling2D((2, 2)))

model.add(layers.Conv2D(28, (4, 4), activation='tanh'))

model.add(layers.Flatten())

model.add(layers.Dense(16, activation='tanh'))

model.add(layers.Dense(1, activation='sigmoid'))

model.compile(loss='binary_crossentropy',
optimizer='adadelta',metrics=[tf.keras.metrics.BinaryAccuracy()])
```

Appendix F

Figure F-1

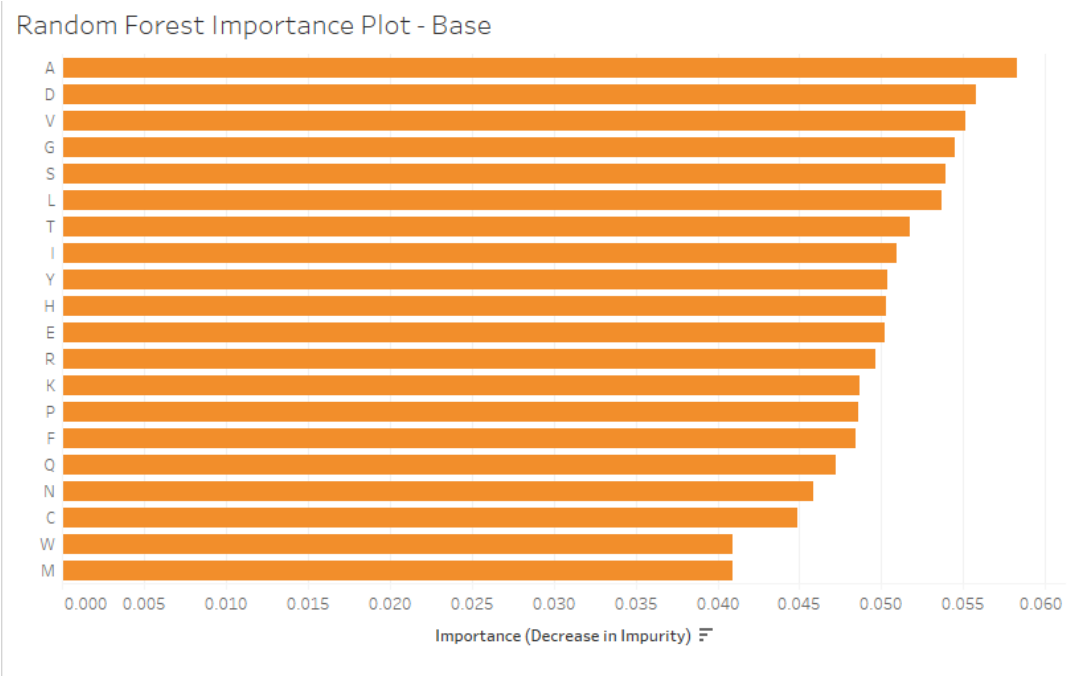


Figure F-1: The Importance plot generated from Random Forest for the Base data set with only the amino acid counts.

Figure F-2

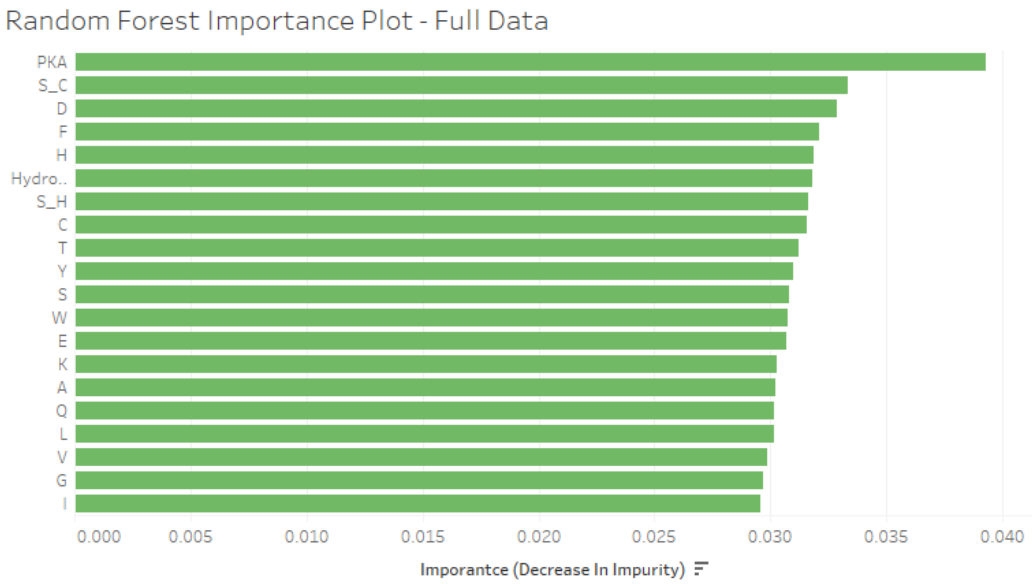


Figure F-2: The importance plot generated from Random Forest for the 'Full Data'. Full data includes the pKa, hydrophobicity, secondary structure tfidf as (S_H, S_C, and S_E for respective secondary structure), tertiary structures, and Amino Acid TFIDF.

Figure F-3

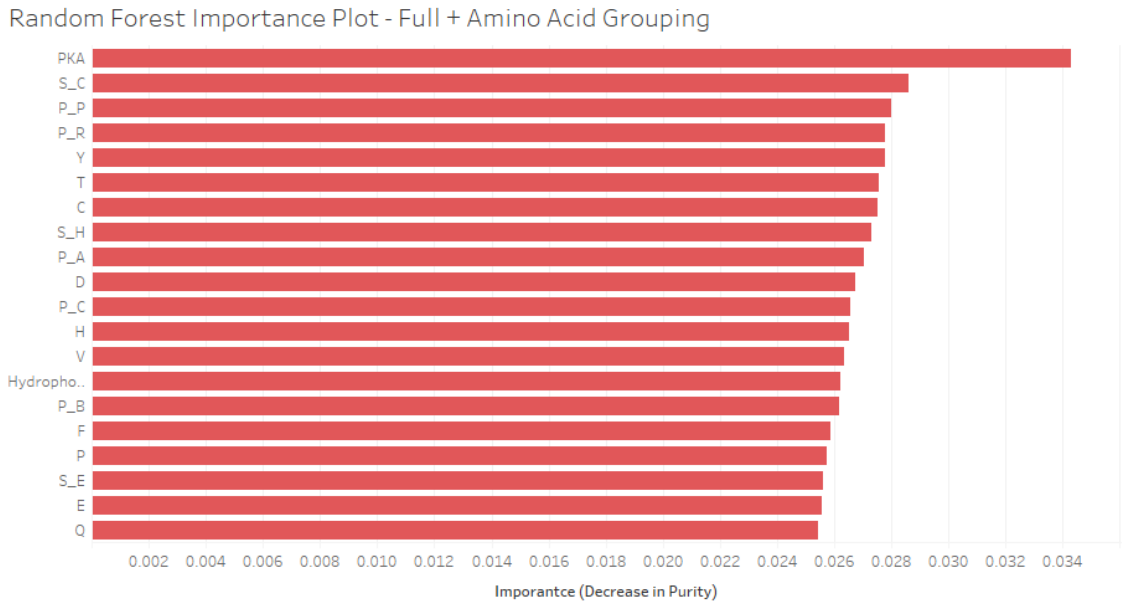


Figure F-3: The importance plot generated from Random Forest for the 'Full data + Property grouping'. The Property grouping technique variables are represented as P_A, P_C, P_B, P_P, P_R, and P_E. Full data include the pKa, hydrophobicity, secondary structure tfidf as (S_H, S_C, and S_E for respective secondary structure), tertiary structures, and Amino Acid TFIDF

Figure F-4

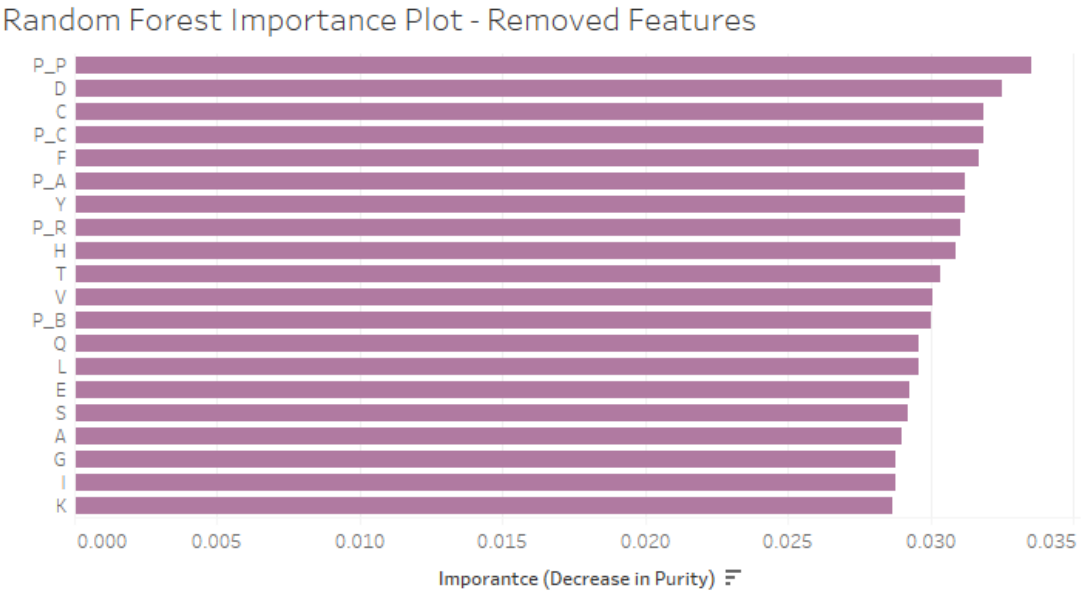


Figure F-4: The importance plot generated from Random Forest for the 'Removed Features'. The Property grouping technique variables are represented as P_A, P_C, P_B, P_P, P_R, and P_E. Full data includes only the tertiary structures and amino acid TFIDF.

References

1. Advanced layer types. Introduction to deep learning: Advanced layer types. (2023, December 12). https://carpentries-incubator.github.io/deep-learning-intro/4-advanced-layer-types.html
2. Ahmed, I. (2023, May 31). What is hard and soft voting in machine learning?. Medium. https://ilyasbinsalih.medium.com/what-is-hard-and-soft-voting-in-machine-learning-2652676b6a32
3. Amino acids reference chart. (n.d.). https://www.sigmaaldrich.com/US/en/technical-documents/technical-article/protein-biology/protein-structural-analysis/amino-acid-reference-chart
4. Argasinska, J. (2020, October). Biomacromolecular structures - an introduction to EMBL-ebi resources. EMBL-EBI homepage. https://www.ebi.ac.uk/training/online/courses/biomacromolecular-structures/
5. CAFA 5 protein function prediction. Kaggle. (n.d.). https://www.kaggle.com/competitions/cafa-5-protein-function-prediction/data
6. Cai, C. Z., Wang, W. L., Sun, L. Z., & Chen, Y. Z. (2003). Protein function classification via support vector machine approach. Mathematical Biosciences, 185(0025–5564), 111–122. https://doi.org/https://doi.org/10.1016/S0025-5564(03)00096-8
7. Cao, R., Freitas, C., Chan, L., Sun, M., Jiang, H., & Chen, Z. (2017). ProLanGO: Protein Function Prediction Using Neural Machine Translation Based on a Recurrent Neural

Network. *Molecules (Basel, Switzerland), 22*(10), 1732. https://doi.org/10.3390/molecules22101732

8. Demir, M. (2023, April 7). Machine learning algorithm series polynomial kernel SVM: Understanding the basics and applications. Medium. https://blog.devgenius.io/machine-learning-algorithm-series-polynomial-kernel-svm-understanding-the-basics-and-applications-89b4b42df137
9. Editorial, M. (2023, December 11). Scikit-learn vs Tensorflow : Which one should you choose? https://metana.io/blog/scikit-learn-vs-tensorflow/
10. "Gene Ontology Browser." *Organic Acid Metabolic Process Gene Ontology Term (GO:0006082)*, www.informatics.jax.org/vocab/gene_ontology/GO:0006082. Accessed 9 Dec. 2023.
11. "Gene Ontology Resource." *Gene Ontology Resource*, geneontology.org/. Accessed 9 Dec. 2023.
12. Hakala, K., Kaewphan, S., Björne, J., Mehryary, F., Moen, H., Tolvanen, M., Salakoski, T., & Ginter, F. (2022). Neural Network and Random Forest Models in Protein Function Prediction. IEEE/ACM Transactions on Computational Biology and Bioinformatics , 19(3), 1772–1781. https://doi.org/https://ieeexplore.ieee.org/document/9291483/authors#citations
13. Gowda, S. N., & Yuan, C. (2019). ColorNet: Investigating the importance of color spaces for image classification. In *Computer Vision–ACCV 2018: 14th Asian Conference on Computer Vision, Perth, Australia, December 2–6, 2018, Revised Selected Papers, Part IV 14* (pp. 581-596). Springer International Publishing.

14. Helmenstine, A. (2023, March 25). What is PKA in Chemistry? acid dissociation constant. Science Notes and Projects. https://sciencenotes.org/what-is-pka-in-chemistry-acid-dissociation-constant/
15. Hillisch A. (2008). Protein structure-based drug design: applications, limitations and future developments. *Chemistry Central Journal*, *2*(Suppl 1), S15. https://doi.org/10.1186/1752-153X-2-S1-S15
16. James, G., Witten, D., Hastie, T., & Tibshirani, R. (2017). An Introduction to Statistical Learning: with Applications in R. Springer.
17. Kandel, I., Castelli, M., & Popovič, A. (2020). Comparative Study of First Order Optimizers for Image Classification Using Convolutional Neural Networks on Histopathology Images. Journal of imaging, 6(9), 92. https://doi.org/10.3390/jimaging6090092
18. K, B. (2022, April 12). Transformers for text classification. Paperspace Blog. https://blog.paperspace.com/transformers-text-classification/
19. Massachusetts Institute of Technology. (n.d.). Singular Value Decomposition (SVD) tutorial. Singular value decomposition (SVD) tutorial. https://web.mit.edu/be.400/www/SVD/Singular_Value_Decomposition.htm
20. Moffat, L., & Jones, D. T. (2021). Increasing the accuracy of single sequence prediction methods using a deep semi-supervised learning framework. Bioinformatics, 37(21). https://doi.org/https://doi.org/10.1093/bioinformatics/btab491
21. Monsters, D. (2017, September 26). 7 types of artificial neural networks for Natural Language Processing. Medium. https://medium.com/@datamonsters/artificial-neural-networks-for-natural-language-processing-part-1-64ca9ebfa3b2

22. Morris, Rhiannon, et al. “Uncovering protein function: From classification to complexes.” Essays in Biochemistry, vol. 66, no. 3, 2022, pp. 255–285, https://doi.org/10.1042/ebc20200108.
23. Ofer, D., Brandes, N., & Linial, M. (2021). The language of proteins: NLP, machine learning & protein sequences. Computational and Structural Biotechnology Journal, 19(2001–0370), 1750–1758. https://doi.org/https://doi.org/10.1016/j.csbj.2021.03.022
24. Overview of protein assays methods. Thermo Fisher Scientific - US. (n.d.). https://www.thermofisher.com/us/en/home/life-science/protein-biology/protein-biology-learning-center/protein-biology-resource-library/pierce-protein-methods/overview-protein-assays.html
25. Ruder, S. (2020, March 20). An overview of gradient descent optimization algorithms. ruder.io. https://www.ruder.io/optimizing-gradient-descent/
26. Sara, S. T., Hasan, M. M., Ahmad, A., & Shatabda, S. (2021). Convolutional neural networks with image representation of amino acid sequences for protein function prediction. Computational Biology and Chemistry, 92(1476–9271), 107494. https://doi.org/https://doi.org/10.1016/j.compbiolchem.2021.107494
27. Sidharth. (2022, December 12). SVM kernels: Polynomial kernel - from scratch using python. PyCodeMates. https://www.pycodemates.com/2022/10/svm-kernels-polynomial-kernel.html
28. Sklearn.preprocessing.MinMaxScaler. scikit. (n.d.). https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.MinMaxScaler.html

29. Super Annotate. (2023, July 6). Activation functions in neural networks [Updated 2023]. Machine Learning . November 30, 2023, https://www.superannotate.com/blog/activation-functions-in-neural-networks
30. Text generation with an RNN: Tensorflow. TensorFlow. (n.d.). https://www.tensorflow.org/text/tutorials/text_generation
31. University of California San Francisco. (2018, February). Amino Acid Hydrophobicity. Amino acid hydrophobicity. https://www.cgl.ucsf.edu/chimera/docs/UsersGuide/midas/hydrophob.html
32. U.S. National Library of Medicine. (n.d.). Conserved domains database (CDD) and resources. National Center for Biotechnology Information. https://www.ncbi.nlm.nih.gov/Structure/cdd/cdd.shtml
33. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... & Polosukhin, I. (2017). Attention is all you need. Advances in neural information processing systems, 30
34. What are convolutional neural networks?. IBM. (n.d.). https://www.ibm.com/topics/convolutional-neural-networks
35. What does tf-idf mean?. Tfidf. (n.d.). https://tfidf.com/
36. Yan, T.-C., Yue, Z.-X., Xu, H.-Q., Liu, Y.-H., Hong, Y.-F., Chen, G.-X., Tao, L., & Xie, T. (2023). A systematic review of state-of-the-art strategies for machine learning-based protein function prediction. Computers in Biology and Medicine, 154(0010–4825), 106446. https://doi.org/https://doi.org/10.1016/j.compbiomed.2022.106446