Using AI to Predict Protein Structural Stability

A Major Qualifying Project submitted to the Faculty of WORCESTER POLYTECHNIC INSTITUTE

In partial fulfillment of requirements for the degree of Bachelor of Science in Bioinformatics and Computational Biology

This report represents the work of one or more WPI undergraduate students submitted to the faculty as evidence of completion of a degree requirement. WPI routinely publishes these reports on the web without editorial or peer review.

Written By:
Mir Sultan (BCB)

Advised By:
Dmitry Korkin, PhD - Advisor (WPI)

Abstract

Alternative splicing contributes significantly to proteome diversity in humans. Yet, many classified alternatively spliced isoforms lack physical evidence of their physiological existence in the human cell. This is likely because these isoform amino acid sequences create structurally unstable proteins that are immediately degraded. To allow for a better understanding of the human proteome's diversity, a Positive-Unlabeled learning classification algorithm was implemented to accurately predict the existence of a protein with protein structural stability being the determining factor. This algorithm was given features consisting of protein data relevant to structural stability and was tested/trained on verified structurally stable proteins and proteins with unknown structural stability. To quantifiably demonstrate its predicting power, the algorithm was then used to predict the structural stability of proteins from the genes CFTR and TP53 that were physically confirmed as structurally stable and unstable. Improvements are necessary, but good results during testing/training were acquired and the algorithm effectively predicted the structural stability of the proteins from CFTR and TP53.

Acknowledgements

A special thank you to Professor Dmitry Korkin, Professor Elizabeth Ryder, Oleksandr Narkykov, Monét Norales, and Andrew Ressler.

- Professor Dmitry Korkin is a Computer Science Professor and the Director of the Bioinformatics and Computational Biology Program at WPI. He went far above and beyond with helping me complete this project as my MQP advisor.
- Professor Elizabeth Ryder is a Biological Science Professor and the Associate Director of the Bioinformatics and Computational Biology Program at WPI. Professor Korkin and Professor Ryder were instrumental in guiding me on how to progress with this project during times of great difficulty.
- Oleksandr Narykov is a graduate student pursuing a PhD in Bioinformatics and Computational Biology at WPI. He was the reason I discovered the protein data sources that allowed this project to happen.
- Monét Norales and Andrew Ressler were my fellow WPI classmates. They also majored in, and completed MQPs in Bioinformatics and Computational Biology. They provided advice vital for working on this project.

Table of Contents

Introduction

Nearly all proteins in human cells undergo alternative splicing, a process that can obtain a diverse pool of isoforms from the same gene through alternative selection of the exons and introns of the gene (Narykov et al 2020). Single-cell and bulk RNA-sequencing data allow scientists to observe the genome-wide amount of post-transcriptional variation influenced by alternatively spliced isoforms which can be specific to tissue, cell type, developmental stage, disease phenotype, etc (Narykov et al 2020). Alternative splicing can also alter several different protein functions and even add new protein functions (Narykov et al 2020). There are several popular databases such as UniProt/SwissProt, Ensembl, and RefSeq that contain detailed information on many protein isoforms/variants. The information provided for these isoforms is often implicated by the gene the protein isoform/variant belongs to. And the amino acid sequence is usually derived from the corresponding DNA transcript. This means that most of the protein isoforms/variants in these databases may not exist.

The DNA sequences provided for these isoforms may result in proteins that are functional, non-functional, misfolded, not properly translated, etc. There are many ways to verify that a protein isoform/variant exists. But most of these methods require time, money, and a variety of lab equipment/materials. There are hundreds of thousands of possible protein isoforms that exist in these databases and there is no way to properly verify their existence in a cheap and time-efficient manner. Furthermore, research involving protein isoforms/variants will be susceptible to errors if the data used is based on protein isoforms/variants that are later proven to not exist. Thus, there is a need for a cheap and fast in-silico approach.

In this project, a machine learning algorithm was implemented to predict if a protein isoform/variant exists by determining its structural stability. Using data consisting of protein isoforms/variants with classified protein-protein interactions (PPI), AI predicted protein isoforms/variants, PDB's collection of verified stable proteins, BioPython, PyBioMed, and SciKit Learn, this machine learning algorithm was developed and used as a tool for classifying protein isoforms/variants. A pipeline was created to provide the tool with the proper input. The pipeline starts with a set of protein sequences that will be processed to determine which proteins already exist. Then, they are paired by gene to avoid feature value overlap. The first protein in the pair is the unknown target protein. The second protein in the pair is the stable known

reference protein which was identified in the first step and is from the same gene as the target protein. Next, features relevant to structural stability are generated from the unknown protein in the pair. The delta features are also recorded, which are the feature values found by subtracting the unknown protein feature values from the feature values of the known stable protein in the pair. This is done to improve classification accuracy by ensuring that feature values indicating structural stability/instability are more flexible being gene-specific. After that, the feature values are subsequently normalized. And for the final step, the normalized features are fed into the tool, which is a Positive-Unknown (PU) classification algorithm that will predict which proteins exist. This pipeline mainly employs semi-supervised learning due to the lack of sequence data for verified non-existent protein isoforms/variants and because of restrictions with PU classification.

Being able to correctly predict whether or not a protein isoform/variant is stable will directly implicate whether it exists in the cell. Many existing protein isoforms/variants retain little to no function of their original protein, yet they are not immediately degraded because they are structurally stable long enough to have their existence physically verified (Tress et al 2017). Classifying structural stability was the method chosen because it is simple and fundamentally critical to protein existence (Taverna and Goldstein 2002). Protein stability is a simple feature and there are a plethora of easily accessible tools that can allow it to be determined directly, or provide information that would help accurately determine it. This project will be beneficial to protein isoform/variant research such that the conclusions/results drawn from the isoforms/variants used will be more viable as they will not be susceptible to errors caused by the isoforms/variants not existing. In the context of diseases caused by protein mutations, this tool could also be used to quickly determine if a patient has the disease genotype.

Background

Alternative Splicing

Protein expression starts with DNA and is catalyzed by the enzyme RNA Polymerase (Hartwell et al 2017). Promoters, DNA sequences near the beginning of genes, signal RNA polymerase to begin turning the gene sequence of DNA into RNA (Hartwell et al 2017). This process is known as transcription (Hartwell et al 2017). RNA polymerase then adds RNA nucleotides to the growing RNA polymer in the 5′-to-3′ direction (Hartwell et al 2017). Transcription is stopped when the RNA polymerase reaches a terminator sequence added to the RNA polymer (Hartwell et al 2017).

Transcription turns DNA into RNA and is the first part of protein expression (Hartwell et al 2017). The last part of protein expression is called translation, but before that, the RNA created from transcription must be processed before being used to synthesize proteins (Hartwell et al 2017). The RNA strand from transcription is processed such that non-coding regions called introns are spliced out (Hartwell et al 2017). Then the strand is capped with a 5′ methyl-G cap and a poly-A tail is added to the RNA strand's 3′ end (Hartwell et al 2017). This RNA strand is now referred to as the mature messenger RNA, also written as mature mRNA (Hartwell et al 2017). The mature mRNA has a 5′ untranslated region, UTR, between the 5′ end and the start codon trinucleotide sequence of Adenine-Uracil-Guanine, AUG, and it also has a 3′ UTR that lies between the stop codon and the poly-A tail at the 3′ end of the mRNA (Hartwell et al 2017).

The mechanism of removing introns from the primary RNA transcript is known as RNA splicing (Hartwell et al 2017). There are three short sequences needed for RNA splicing which are in the primary transcript (Hartwell et al 2017). The first is the splice donor site which is where the 3′ end of a coding region, an exon, intersects the 5′ end of an intron (Hartwell et al 2017). In most splice donor sites, a Guanine-Uracil dinucleotide that begins the intron is flanked by Adenine and Guanine nucleotides, which are known as the purines (Hartwell et al 2017). The second short sequence needed is the splice acceptor site (Hartwell et al 2017). The splice acceptor site is at the 3′ end of the intron where it connects to the next exon (Hartwell et al 2017). The final nucleotides of the intron are always an Adenine-Guanine dinucleotide sequence usually preceded by 12–14 Cytosine and Uracil nucleotides which are known as pyrimidines

(Hartwell et al 2017). The third short sequence needed is the branch site, which is located within the intron around 30 nucleotides upstream of the splice acceptor site (Hartwell et al 2017). The branch site must include an Adenine and is usually rich in pyrimidines (Hartwell et al 2017). Using these three short sequences, two sequential cuts are made (Hartwell et al 2017). The first cut is at the splice donor site and the second happens at the splice acceptor site which removes the intron allowing for precise splicing of adjacent exons (Hartwell et al 2017).

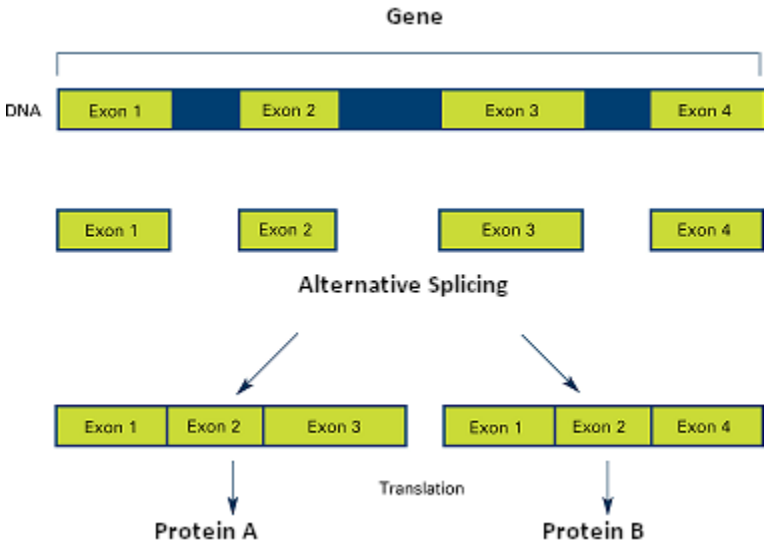


Figure 1. A visual of Alternative Splicing that shows how from the same gene, distinct exon mixes are made allowing one gene to code for many proteins (Alternative Splicing of Genes: Definition, Mechanism & Regulation 2015).

However, RNA splicing can occur in a different way known as alternative splicing (Hartwell et al 2017). RNA splicing during different stages of development is regulated such that some splicing signals/splicing sequences may be ignored (Hartwell et al 2017). As shown in Figure 1, splicing can occur between the splice donor site of one intron and the splice acceptor site of a different intron downstream that is not immediately adjacent to the splice donor site of the intron that is upstream (Hartwell et al 2017). Figure 1 also shows how that alternative splicing produces different mRNA molecules that may encode related proteins with amino acid sequences and functions that usually overlap and can be distinct from each other in parts or as a whole (Hartwell et al 2017). Alternative splicing can allow for the nucleotide sequence of a primary transcript to produce more than one kind of polypeptide increasing the proteomic diversity of the organism which can also be inferred from Figure 1 (Hartwell et al 2017). This is

what is meant to explain most of how the 27,000 genes in the human genome can encode hundreds of thousands of different proteins estimated to exist in human cells (Hartwell et al 2017).

After alternative splicing, protein translation happens normally where the ribosome starts reading the mRNA and using tRNAs to provide the amino acids to the corresponding mRNA codons such that the ribosome can start synthesizing the protein by chaining the recruited amino acids together (Hartwell et al 2017). A specific example showcasing the strength and necessity of Alternative Splicing in mammals is how the heavy chain of the antibody is expressed (Hartwell et al 2017). Alternative splicing of the gene encoding the antibody heavy chain determines whether the antibody proteins become part of the B lymphocyte membrane, or are instead secreted into the blood (Hartwell et al 2017). The gene for the antibody heavy chain has eight exons and seven introns (Hartwell et al 2017). The sixth exon has a splice-donor site (Hartwell et al 2017). To make the antibody that will be part of the B lymphocyte membrane, all exons except for the right-hand part of the sixth exon are joined to create an mRNA encoding a hydrophobic, water-hating and lipid-loving, C terminus which creates a necessity for the encoded protein to join with the B lymphocyte membrane (Hartwell et al 2017). For the antibody secreted into the blood, the entirety of the first six exons are spliced together to make an mRNA encoding a heavy chain with a hydrophilic, water-loving, C terminus which is fine on its own (Hartwell et al 2017). These two kinds of mRNAs formed by alternative splicing thus encode slightly different proteins that are directed to different parts of the body (Hartwell et al 2017).

This example showcases the significance of Alternative Splicing/splicing since it allows for increased protein functional variety that is critical to human existence. With antibodies that are just part of the B lymphocyte membrane, there would be more rampant infections in parts of the human body that the B lymphocyte cannot physically reach (Silverstein 2003). So with insoluble antibodies that are secreted into the blood, immune responses are more readily able to be initiated in more parts of the body when necessary (Silverstein 2003). Not only does this indicate significance for proteomic diversity, it also speaks to genetic efficiency since it allows for a single gene to have multiple different and necessary functions (Hartwell et al 2017). Alternative splicing is known to diversify the functions of some genes, but the extent to which protein isoforms globally contribute to functional complexity on a proteomic scale remains unknown, hence why it is important to study (Yang et al 2016).

Protein Structural Stability

In order to understand how the existence of a protein can be verified by determining protein structural stability, it is critical to understand what happens to proteins that are not structurally stable. In normal eukaryotic cells, structurally unstable proteins are degraded (Voet et al 2016). Therefore, it is important to understand how the process of protein degradation connects to protein stability.

As explained before, proteins are obtained from nucleotide sequences of the corresponding genes via transcription and translation (Voet et al 2016). The splicing of the RNA primary transcript is a process that happens before translation and after transcription normally or alternatively, creating the mature mRNA that will be used immediately for synthesizing proteins or protein isoforms/variants in translation (Voet et al 2016). To a degree, all proteins are inherently unstable which is a built-in type of regulation for protein function and quantity (Voet et al 2016). It ensures that proteins are at their desired levels and that their functions are carried out only when they need to be (Voet et al 2016). This obviously contributes to energetic efficiency regarding protein synthesis, protein degradation, and protein function (Voet et al 2016). It requires energy to make proteins and to degrade proteins, and some proteins also require energy to execute their functions (Voet et al 2016). Conserving energy in any one of these areas will allow for the other areas to progress more smoothly, thus making protein degradation have an effect on many metabolic processes (Voet et al 2016).

Protein degradation not only specifically targets structurally unstable proteins, it has three functions (Voet et al 2016). The first is to store nutrients in the form of proteins to break them down in times of metabolic need which correspond to processes most relevant in muscle tissue (Voet et al 2016). The second function is to eliminate abnormal proteins whose accumulation would be harmful to the cell (Voet et al 2016). The third function, as mentioned prior, is to permit the regulation of cellular metabolism by eliminating unnecessary enzymes and regulatory proteins (Voet et al 2016). Abnormal proteins consist of damaged, misfolded, aggregated or unnecessary non-functional proteins (Vilchez et al 2014). These types of proteins are scavenged by the proteasome, an enzyme that degrades proteins, or through autophagy-lysosome, phagocytosis initiated by the cellular organelle known as the lysosome (Vilchez et al 2014). The proteasome does not work at random whereas the lysosome can (Vilchez et al 2014). The proteasome targets proteins for degradation that are marked with ubiquitin molecules (Vilchez et

al 2014). The ubiquitin–proteasome system, UPS, is a complex mechanism where proteins are targeted for degradation by ubiquitination and subsequently recognized, unfolded, and proteolyzed by the proteasome (Vilchez et al 2014). In the lysosome, proteins are degraded in a process referred to as autophagy-mediated proteolysis (Vilchez et al 2014).

Ubiquitination itself targets areas of proteins that are not structurally stable (Vilchez et al 2014). Therefore, the longer a protein exists or is active in the cell, its inherent structural instability will cause more of it to become structurally unstable making it susceptible to being tagged by several ubiquitin molecules in different parts of the protein (Vilchez et al 2014). The more a protein is tagged by ubiquitin, the faster the proteasome is able to recognize and degrade it (Vilchez et al 2014). Because of this, if a protein is not structurally stable right after being synthesized by the ribosome, it is at risk of being rapidly tagged several times by ubiquitin, quickly getting degraded by the proteasome (Vilchez et al 2014). Furthermore, many proteins that are structurally unstable often disassemble into aggregates during or right after protein synthesis, making it susceptible to both the lysosome and the UPS (Cooper 2000). It is also important to remember that protein structure enables protein function (Cooper 2000). Therefore, if a protein is lacking structural stability, it usually cannot execute its function (Cooper 2000). It is possible for some structurally unstable proteins to be able to perform their functions, but the structurally unstable proteins will not be able to perform their function for long and would lack functional impact (Cooper 2000).

That being said, protein structural stability can allow for a protein to function in the cell long enough for it to have a significant and/or functional impact. Thus knowing the structural stability of a protein can verify whether or not it would exist in the cell. However, a distinction must be made: a protein can be structurally stable but non-functional. In fact, there are mutations that create structurally stable non-functional proteins (García-Alai et al 2008). These types of variants often contribute to genetic diseases like cancer (García-Alai et al 2008). At the very least, if a human protein is found to be structurally stable, then it most certainly exists in the human proteome, which would verify its existence. but this does not always guarantee that the protein is functional.

Protein Verification

Knowing the structural stability of a protein can indicate whether or not it can exist inside the cell, so it is necessary to understand the methods in which protein verification can be done. It is important to note that not all protein verification methods concern protein structural stability.

In regards to physically verifying the existence of a protein, there are two main methods. The first is sequencing proteins extracted from in vivo samples (Cooper 2000). This shows which proteins actually exist. Depending on the sample, it is possible to extract both isoform/variant proteins and standard proteins. If you have the DNA sequence of a gene, it is possible to put it through artificial transcription, splice it as desired, and then transfect the mRNA into a cell so that it can undergo translation (García-Alai et al 2008). Secondly, there are also methods to observe the behavior of the protein using assays and fluorescent tagging (García-Alai et al 2008). If the protein is observed to not be immediately degraded, and is even functional, then the existence of that protein can be verified. It is also worth noting that these are the main methods of physical protein verification, but physical protein verification is not limited to this. Protein structural stability and protein function are multifaceted properties that can provide protein verification and these properties can be determined through other less direct means (Yang et al 2016).

However, there is another, possibly more efficient, method to verify protein existence. It involves using databases like the Protein Database, PDB. PDB has around 53,000 human protein entries, which is less than half of all the estimated human proteins (Berman et al 2000). It is possible to use PDB to verify a protein's existence with its amino acid sequence if there is a protein entry in the database that perfectly matches the unverified sequence. This is because PDB requires structural information/validation for each of its protein entries which necessitates experimental information from physically verifying the protein's existence in the first place (Berman et al 2000). This type of information cannot be provided if the protein is not structurally stable (Berman et al 2000). Thus, using this method could be more efficient because it ensures that physical verification is not repeated for a protein that is already proven to exist.

Relevant Past Research

There have been other attempts in the past for predicting protein stability and/or protein structure using artificial intelligence (AI) programs that are worth mentioning.

ProTstab predicted protein stability by predicting the melting temperature, written as T_M, of input proteins (Yang et al 2019). T_M is the lowest temperature required for a protein to completely denature and it is the accumulation of various factors contributing to protein structural stability directly related to protein function and structure (Gorania et al 2010). Knowing this value can be informative of a protein's structural stability as a protein with a very stable structure will have a higher T_M than other proteins similar in size/molecular weight and/or function (Gorania et al 2010). This is a simple and safe approach, but this is limited since there are many more features indicative of protein structural stability such as instability index and the change in Gibbs free energy of a protein's folded and unfolded state indicated by ΔG (Chen et al 2020). Furthermore, the features used to help this AI program make its prediction are not explained and were derived from the application PROFEAT, which is no longer accessible online (Yang et al 2019). The testing and training data used in this implementation also consisted entirely of proteins from UniProt, meaning that a significant portion of the proteins used as positive stable data may not actually be stable (Yang et al 2019).

iStable 2.0 is another approach using an artificial intelligence program that combines different algorithms (Chen et al 2020). It uses PDB as a source for its stable data and goes for ΔΔG (Chen et al 2020). ΔΔG is an excellent feature to derive for protein stability because it is the ΔG change measured in kcal/mol between the wild-type/WT and mutant protein (Chen et al 2020). G represents the Gibbs free energy of a protein (Chen et al 2020). A single amino acid mutation at the very least, could change the structural stability of a protein after folding by making a smaller change in G, shown as ΔG (Chen et al 2020). The difference in folding free energy change between wild type and mutant protein, ΔΔG, is often considered as an impact factor of protein stability changes (Chen et al 2020). The ΔΔG directly indicates the thermodynamic stability of a protein which corresponds directly to its structural stability (Tokuriki et al 2008). However, iStable 2.0 relies on other algorithms accessible through the internet and will fail if one of them gets discontinued (Chen et al 2020). In this AI program, the only mutant proteins used in the data were created via point mutations so it is too specific to those proteins and not applicable to unknown protein isoforms/variants (Chen et al 2020).

SCooP is another AI program that predicts the values for ΔG and T_M using PDB, but its input requires the 3D structure and the host organism of the target protein (Pucci et al 2017). This limits this AI program to only being able to use data from verified proteins and proteins which have 3D structural data, which is not provided for many verified proteins (Pucci et al 2017). However, the features are well explained and consist of many thermodynamic variables properly aligning with the thermodynamic values this AI program is trying to predict (Pucci et al 2017). This is useful for obtaining more details on the structural stability of verified proteins regardless of whether they are variants/isoforms, but again, it is unable to provide any information on a protein that lacks such detailed data (Pucci et al 2017).

Methodology

To solve the issue of not knowing which protein isoforms actually exist inside the cell, a Positive-Unlabeled learning model was created with the purpose of accurately predicting the structural stability of a given protein. As mentioned earlier, protein structural stability indicates whether or not it exists in the cell because if a protein is not structurally stable, it will be quickly degraded. In order for this model to work, it needed to be given features corresponding to relevant protein structural information and tested/trained with the right proteins. This methodology section explains the tools used in this project and the process applying those tools in the preparation and application of this PU learning model.

PDB Usage

PDB is the Protein Database (Berman et al 2000). It is a reliable source of proteins that are structurally stable and verifiably exist due to how each protein entry in PDB requires structural validation, which cannot be provided for structurally unstable proteins (Berman et al 2000). All the human proteins in PDB were downloaded from https://www.wwpdb.org/ftp/pdb-ftp-sites such that protein matching could be done more efficiently without the PDB website (Berman et al 2000).

The First Dataset on Protein Isoforms

In an attempt to understand the extent to which protein isoforms created by Alternative Splicing contribute to contribute to functional complexity on a proteomic scale, the authors of *Widespread expansion of protein interaction capabilities by alternative splicing* systematically cloned full-length open reading frames of alternatively spliced transcripts for several human genes and used protein-protein interaction profiling to functionally compare hundreds of protein isoform pairs (Yang et al 2016). The protein isoform sequences used in this paper were used in this project for what would become the positive stable samples.

They expressed several different protein isoforms in vitro and only kept record of the ones they successfully expressed with their genes and amino acid sequences (Yang et al 2016). They profiled the protein-protein interactions, PPIs, of the isoforms which made it possible to

determine which isoforms could be verified as structurally stable (Yang et al 2016). As mentioned prior, there are other methods of physically verifying the existence of a protein. In this research, the protein isoforms were successfully expressed in vitro as opposed to in vivo (Yang et al 2016). If a protein isoform was able to successfully interact with another protein, then that protein isoform was structurally stable and thus its existence was verified since PPIs are not possible without structural stability (Yang et al 2016).

All the protein isoforms whose PPIs they profiled were found to be structurally stable since they all had at least one PPI (Yang et al 2016). As a result, the profiled protein isoforms used in this research article is the source of the positive labeled data used in this project.

ASPicDB Usage

There are hundreds of thousands of proteins estimated to exist in the human proteome alone (Yang et al 2016). Thus, there is a need to profile all the proteins there are in order to better understand the human proteome. ASPicDB is a database designed to provide access to reliable annotations of the alternative splicing pattern of human genes (LITA 2012). ASPicDB provides annotations of predicted alternative splicing that human genes can undergo and these predictions and annotations are provided by multiple machine learning tools (Martelli et al 2011). This means that this database contains the genes and predicted amino acid sequences for human protein isoforms/variants that have not been verified yet (LITA 2012).

In order to better understand the usage of predicted unverified protein data, it is necessary to discuss the aspects of this project that contribute to its significance. The objective of this project is to use a machine learning algorithm to predict the structural stability of protein isoforms. Machine learning algorithms need two types of data in order to make accurate classification predictions: positive and negative data (Elkan and Noto 2008). In this project, positive data would correspond to stable proteins and negative data would correspond to unstable proteins. However, due to the lack of information for protein variants that would be structurally unstable, the data from unverified proteins was the only possible substitute. This must be acknowledged because this unverified data likely consists of both positive and negative data which complicates the predicting power of any AI program that would be used. Therefore, it would be optimal if the unverified data is likely to contain negative proteins. With the sequences in ASPicDB being predictions not directly derived from actual DNA sequences as opposed to

other popular databases like UniProt and RefSeq, this increases the likelihood of unstable proteins being in this database (Berman et al 2000). DNA regulation and repair is often effective, ensuring that most genes expressed are done purposefully and efficiently such that the resulting protein is functionally relevant (Hartwell et al 2017). For the unverified protein isoforms/variants that only have corresponding DNA transcripts, it is expected that most of these unverified protein isoforms/variants actually exist in vivo simply because of this. Therefore, using a source that has proteins whose existence is not verified in DNA transcripts could allow for more variety in the proteins whose existences are unknown. In an attempt to avoid issues caused by a positive data bias or lack of negative data, the predicted protein isoforms/variants in this database were used to supply the protein isoforms/variants of unknown stability in this project.

PU Learning

The lack of accessible negative data influenced many vital components of this project. The most vital component being the machine learning algorithm chosen for this project. The objective of this project was to predict if a given protein isoform will be structurally stable using a machine learning algorithm. This type of objective would require a binary classification machine learning algorithm since only two types of predictions will be made, stable or not stable. In traditional binary classification algorithms, the goal is to learn a model that is able to distinguish between positive and negative examples where the training data contains both positive and negative examples and all the data is fully labeled (Bekker and Davis 2020). As stated prior, negative data was not accessible which mandated the use of an algorithm that could work without it. Positive-Unlabeled learning, referred to as PU learning, is the setting where a "learner" only has access to positive and unlabeled data assuming that the unlabeled data may contain positive and negative samples (Bekker and Davis 2020). PU learning is a variant of the traditional format where the training data consists of positive and unlabeled data (Bekker and Davis 2020). PU learning incorporates unlabeled data into the learning process and specializes in the standard semi-supervised learning where there are usually some labeled examples for all classes (Bekker and Davis 2020).

The method chosen for the PU learning implementation in this project was the prior class incorporation with post-processing (Bekker and Davis 2020). This method was implemented in four primary steps. First, a classifier is fit onto the data set containing labeled and unlabeled data

(Agmon 2020). Fitting the classifier on the data will train it to predict the probability that a given sample is labeled, $P(s=1|x)$ (Agmon 2020). Second, the classifier will predict the probability that the known positive samples in the data set are labeled so that the predicted results will show the probability that a positive sample is labeled $P(y=1|s=1)$. This will be done multiple times through implementation of k-fold cross validation where the mean of these predicted probabilities will be calculated to provide the label frequency $P(s=1|y=1)$. With the label frequency estimation, in order to predict the probability that a specific sample k is positive is to estimate the probability that k is labeled $P(s=1|k)$. Third, the classifier already trained to provide $P(s=1|k)$ will be used to estimate the probability that k is labeled. Fourth, k can be classified by dividing the probability of it being labeled by the label frequency, $P(s=1|k)/P(s=1|y=1)$, to get the probability of which class it belongs to.

There are multiple ways to implement PU learning, but the implementation in this project was chosen because of how it can be evaluated (Bekker and Davis 2020). The first thing required for the method of prior class incorporation with post-processing implementation is allowing for the Selected Completely At Random, SCAR, assumption to be made (Bekker and Davis 2020). Before any probabilities are estimated, a subset of the positive labeled data has to be unlabeled in accordance with the SCAR assumption (Bekker and Davis 2020). The labeled examples must be a subset selected completely at random, independent from their attributes/feature values, from the true positive distribution to ensure that the probability for selecting a positive example, is constant and equal to the label frequency so that the SCAR assumption can be made (Bekker and Davis 2020). Additionally, this assumption was made because it is likely the entirety of the positively labeled data used is representative of the true positives due to the gene variation. With the SCAR assumption and method chosen, this allows for the PU learning performance to be evaluated by the recall metric (Bekker and Davis 2020). But because the data contains unlabeled samples, normal evaluation metrics cannot be calculated due to the requirement of knowing the true positive and true negative samples (Bekker and Davis 2020). However, recall can be calculated and viable evaluation metric estimates can be made with bias correction (Ramola et al 2019). Traditionally, PU learning model performances are evaluated under the assumption that all the unlabeled samples are negative (Ramola et al 2019) Depending on the actual distribution of positive and negative samples that are within the set of unlabeled samples, this could provide wildly inaccurate estimations due to positive/negative bias (Ramola et al 2019). With theoretical

support as well as empirical evidence, reliable performance estimation methods that employ bias-correcting were created for PU learning models (Ramola et al 2019).

The estimation methods implemented in this project are for the evaluation metrics accuracy, balanced accuracy, F-measure, and the Matthews correlation coefficient (Ramola et al 2019). Accuracy is the probability that a random example is correctly classified, balanced accuracy is the average accuracy on the positive and negative examples weighed equally, the F-measure is the harmonic mean of recall and precision, and the Matthews correlation coefficient is the correlation between the true and predicted class (Ramola et al 2019).

Table 1. Evaluation Metric Formulae

Evaluation Metric	Formula
True positive rate	$\widehat{\gamma} = \frac{tp}{tp+fn}$
False positive rate	$\widehat{\eta} = \frac{fp}{tn+fp}$
Positive classification prediction probability	$\widehat{\theta} = \frac{tp+fp}{tp+fn+tn+fp}$
Positive Sample Proportion	$\widehat{\pi} = \frac{tp+fn}{tp+fn+tn+fp}$
Accuracy	$acc = \pi\gamma + (1-\pi)(1-\eta)$
Balanced Accuracy	$bacc = \frac{1+\gamma-\eta}{2}$
F score	$F = \frac{2\pi\gamma}{\pi+\theta}$
Matthews Correlation Coefficient	$mcc = \sqrt{\frac{\pi(1-\pi)}{\theta(1-\theta)}} * (\gamma - \eta)$

The classification predictions provided by the PU model, or any classification model, can provide four subsets: the true positives (tp), false negatives (fn), false positives (fp), and true negatives (tn). When a model correctly predicts a positive sample to be positive, it is part of the tp subset. But when the model incorrectly predicts a positive sample to be negative, it is part of

the fn subset. A sample is part of the fp subset when the model incorrectly predicts it to be positive. And a sample is part of the tn subset when the model correctly predicts it to be negative. When dealing with data that is labeled, the number of samples each subset contains can be used to provide values that inform useful evaluation metrics.

These values are represented by the greek symbols γ, η, π and θ in Table 1. The true positive rate is represented by γ, which is the number of correctly classified positive samples or tp, divided by the total number of positive samples; the tp and fn. The false positive rate is represented by η, which is the number of incorrectly classified negative samples or fp, divided by the total number of negative samples; the tn and fp. The probability of a positive classification prediction is represented by θ, which is the sum number of fp and tp samples divided by the total number of samples in the data set. The proportion of positive samples in the entire data set is represented by π, which is the sum number of tp and fn samples divided by the total number of samples in the data set.

These values are labeled with ‘$\widehat{\ }$’ since they are experimentally determined. Yet, the values they provide can be 100% accurate should they be found from a data set that is completely labeled. Thus, when these values are provided for a data set with unlabeled samples, they must be estimated and marked as experimentally determined.

Table 2. Bias-Corrected Evaluation Metric Estimation Formulae

Bias Corrected Evaluation Metric Estimate	Formula
True positive rate	$\widehat{\gamma}_{cr} = (\widehat{\beta} - \widehat{\alpha})^{-1}((1 - \widehat{\alpha})\,\widehat{\gamma} - (1 -$
False positive rate	$\widehat{\eta}_{cr} = (\widehat{\beta} - \widehat{\alpha})^{-1}(\widehat{\beta\eta} - \widehat{\alpha\gamma})$
Positive classification prediction probability	$\widehat{\theta} \rightarrow \theta,\ \widehat{\theta}$estimates θ
Positive Sample Proportion	$c = \widehat{\pi},\ \ \widehat{\pi}_{cr} = c\widehat{\beta} + (1 - c)\,\widehat{\alpha}$
Accuracy	$\widehat{acc}_{cr} = \widehat{\pi}_{cr}\widehat{\gamma}_{cr} + (1 - \widehat{\pi}_{cr})(1 - \widehat{\eta}_{cr}$
Balanced Accuracy	$\widehat{bacc}_{cr} = \frac{1+\widehat{\gamma}_{cr}-\widehat{\eta}_{cr}}{2}$
F score	$\widehat{F}_{cr} = \frac{2\widehat{\pi}_{cr}\widehat{\gamma}_{cr}}{\widehat{\pi}_{cr}+\widehat{\theta}}$
Matthews Correlation Coefficient	$\widehat{mcc}_{cr} = \sqrt{\frac{\widehat{\pi}_{cr}(1-\widehat{\pi}_{cr})}{\widehat{\theta}(1-\widehat{\theta})}} * (\widehat{\gamma}_{cr} - \widehat{\eta}_{cr})$

The evaluation metric estimation formulae provided in Table 2 is proven reliable by theoretical support and empirical evidence (Ramola et al 2019). Table 2 displays adjustments against the inherent PU learning negative bias in γ, η, π, θ, accuracy, balanced accuracy, F score, and Matthews correlation coefficient. A new symbol is used as denoted by c and represented by $\widehat{\pi}$ (Ramola et al 2019). It is the proportion of the pseudo-positives and a value necessary for getting reliable estimates for $\widehat{\pi}_{cr}$(Ramola et al 2019). Also, the experimentally determined value for θ was neatly found to be a reliable estimate for θ specific to PU learning (Ramola et al 2019). In Table _2, the symbols α and β indicate the values used to correct some of the metrics. The

symbol α is the proportion of positive samples in the unlabeled data subset, and β is the proportion of positive samples in the labeled subset, which should be equal to 1 (Ramola et al 2019). Again, the the true proportions of negative and positive samples in the unlabeled set are unknown so the values used for α have to be experimentally derived from the predicted labels the PU model assigns to the unlabeled samples (Ramola et al 2019).

Table 3. Evaluation Metric Formulae for data samples of known structural stability

Evaluation Metric	Formula
Recall	$Recall = \frac{True\ Positive}{True\ Positive + False\ Negative}$
Precision	$Precision = \frac{True\ Positive}{True\ Positive + False\ Posit}$
F1 score	$F1 = 2 * (\frac{Precision*Recall}{Precision + Recall})$

When there were data samples of known structural stability, the evaluation metrics of Recall, Precision, ROC, and F1 score were used. Recall is the percentage corresponding to the true positives divided by the sum of the true positive samples and false negative samples as seen in Table 3 (Shung 2018). Precision, as displayed in Table 3, is the number of true positives divided by the sum of true and false positives (Shung 2018). The overall accuracy as shown by the F1 score. As seen in Table 3, it is the harmonic mean of precision and recall thus it is a value representing a balance between precision and recall making it a good indication for the model accuracy (Shung 2018). ROC stands for the receiver operating characteristic which is the 2-dimensional curve representing the relationship between the recall and the false positive rate (Narkhede 2018). ROC along with the area under the ROC curve corresponds to the capability of the model to distinguish between classes (Narkhede 2018). This capability is presented as a number between 0 and 1 where the closer to 1 the number is, the more capable the model is (Narkhede 2018). The ROC score was calculated using roc_auc_score function from the metrics module in SciKit Learn (Buitinck et al 2013).

XGBoost

XGBoost is a decision-tree-based ensemble Machine Learning algorithm that uses a gradient boosting framework (Morde 2019). It has a wide range of applications including but not limited to regression and classification (Morde 2019). Gradient boosting refers to a special case of boosting where errors are minimized by gradient descent algorithm meaning the new decision trees include and build upon the previous trees (Morde 2019). XGBoost uses a gradient boosting framework and improves upon it with system optimization including parallelization, tree pruning, and hardware optimization (Morde 2019). In addition to that, it has built-in features like regularization to avoid overfitting, sparsity awareness to handle different feature sparsity patterns, weighted quantile sketch to find the optimal split points among weighted datasets, and cross-validation taking away the need to explicitly program this search and to specify the exact number of boosting iterations required in a single run (Morde 2019). These are many reasons as to why XGBoost is considered to be one of, if not the best decision tree based algorithms in use today (Morde 2019). This is used as the classifier in the PU learning implementation because the prior class incorporation method desires use of non-traditional classifiers like XGBoost as opposed to traditional binary classifiers like Support Vector Mechanisms (Bekker and Davis 2020).

k-fold Cross Validation

k-fold cross validation is a data partitioning strategy to effectively use a dataset to build a more generalized model which avoids overfitting on the training data allowing for the model to perform better on unseen data (Pramoditha 2021). Performing well with unseen data is the entire point behind Machine Learning and is accordingly an ideal way to evaluate it hence why input data is often split into a training set and a test set (Pramoditha 2021). This process of splitting data can be done more effectively with k-fold cross-validation (Pramoditha 2021). k-fold cross-validation can be used for evaluating a model's performance and hyperparameter tuning (Pramoditha 2021). In this project, k-fold cross validation is implemented in the PU learning to help evaluate the model's performance.

This may seem redundant knowing that the classifier XGBoost used in this project's PU learning implementation automatically implements cross-validation (Morde 2019). However, it

must be noted that due to the non-traditional setup of the PU learning model, the classifier is used to predict two separate probabilities before the PU learning model is able to test and make its predictions (Agmon 2020). The XGBoost classifier is not being used to make the full prediction. Therefore, k-fold cross validation was manually implemented specific to PU learning to allow for better model evaluation. When it comes to traditional k-fold cross validation, the data, which is all labeled, is randomly split up into a number of k folds/subsets (Pramoditha 2021). There will be k iterations where each iteration will test on a distinct k subset, and the remaining subsets will be what the model is trained on for that iteration, the training data (Pramoditha 2021). The testing performance will be evaluated each iteration and the resulting metrics after all the iterations will be averaged and that average will be collected (Pramoditha 2021). In PU learning, one distinct difference has to be made; the k subsets that will be tested upon will not be subsets of the entire data, but only of the positive labeled data (Elkan and Noto 2008). This is because only the labeled data in PU learning is positive and therefore can be evaluated in comparison to the unlabeled data, which cannot be evaluated because its true classification is unknown and cannot be verified (Elkan and Noto 2008).

t-SNE plotting

t-distributed Stochastic Neighbor Embedding, known as t-SNE, is a tool to visualize high-dimensional data (van der Maaten and Hinton 2008). The technique is an improvement of Stochastic Neighbor Embedding that is easier to optimize and reduces the tendency to crowd data points about the graph origin producing significantly better visualizations (van der Maaten and Hinton 2008). This is important and useful for high-dimensional data that contain many different, but related, low-dimensional manifolds (van der Maaten and Hinton 2008). An example of this would be a data set of objects where the data for each object contains different classes as well as image data corresponding to different viewpoints of it (van der Maaten and Hinton 2008). For visualizing the structure of large data sets, t-SNE can use random walks on neighborhood graphs to allow the implicit structure of all the data to influence how a data subset should be visualized (van der Maaten and Hinton 2008).

This technique is popular in the field of machine learning since it is able to create compelling two-dimensional maps from data with hundreds to thousands of dimensions/features without data point labels (Wattenberg et al 2016). t-SNE has a tunable parameter called

perplexity that can affect the resulting visualizations it produces (van der Maaten and Hinton 2008). Perplexity can be interpreted as a smooth measure of the effective number of neighbors thus making the performance of t-SNE robust to perplexity changes (van der Maaten and Hinton 2008). But this can allow for more accurate visualizations to be made by manually informing the tool on how many smooth neighbors there should be in the data if the visualizations initially do not reflect as much (van der Maaten and Hinton 2008). Because this technique is popular for visualizing high-dimensional data in two dimensions relatively well, this tool can be used to showcase clustering in large data sets with thousands of features to visually display the different types of data within the set. Thus, it can potentially inform the performance of the PU learning implementation by displaying clusters that likely correspond to the true negatives and the true positives.

Feature Selection

In this project, features are derived from the protein isoform/variant pairs and are then fed into the PU learning implementation for training and testing. These features correspond to a variety of values relevant to protein structural stability in order to provide the AI program only the essential information it needs to make accurate predictions. However, it is possible that there are some features employed in the AI program that contribute very little to its predicting power. The AI program would be more efficient if it did not have to process such irrelevant feature data. Feature selection reduces the amount of features and it occurs before any actual training and testing happens. Feature selection is employed in this PU learning implementation to identify the features that are most useful and only use those features during the testing and training. There are multiple ways feature selection can work and there are three feature selection methods that were used in this project.

The first feature selection method was univariate feature selection. This works by selecting the best features according to univariate statistical tests (Buitinck et al 2013). SelectKBest was chosen as the method for univariate feature selection that removes all but a number k, highest scoring features according to a specified univariate statistical test (Buitinck et al 2013). In this method, chi squared was the specified univariate statistical test because it measures dependence between stochastic variables allowing the identification of features that are

the most likely to be independent of class and therefore irrelevant for classification (Gajawada 2019).

The second feature selection method employs a Random Forest Classifier. Random forests can have over one thousand decision trees, with each tree built over a random extraction of the features (Dubey 2018). Each tree is a sequence of yes-no questions based on a combination of features and general data observations (Dubey 2018). Each node corresponds to a question and is where the tree divides the dataset into two buckets with each bucket having observations that are similar yet distinct from each other (Dubey 2018). The importance of each feature is directly related to the purity of each bucket (Dubey 2018). The measure of impurity is the Gini impurity/the information gain due to the purpose of this project being classification (Dubey 2018). From the Scikit-learn module ensemble, a Random Forest classifier can be accessed which has this impurity measurement implemented and accessible (Buitinck et al 2013). When coupled with the SelectFromModel meta-transformer, it is possible to easily extract the most relevant features using the Random Forest's impurity-based feature evaluations (Buitinck et al 2013).

The third and final feature selection approach is a tree-based feature selection. When coupled with the SelectFromModel meta-transformer, tree-based estimators can be used to compute impurity-based feature importance which can be used to discard irrelevant features (Buitinck et al 2013). This method uses the ExtraTreesClassifier from the Scikit-learn module ensemble (Buitinck et al 2013). It is a meta estimator that fits a number of randomized decision trees/extra-trees on various sub-samples of the dataset and uses averaging to improve the predictive accuracy and control over-fitting (Geurts et al 2006). But coupled with the SelectFromModel meta-transformer, it instead informs which features should be kept and which should be discarded (Buitinck et al 2013).

Feature Importance

In addition to selecting only the best features for the AI program such that it becomes more efficient, it is also worth evaluating the features used. Feature importance refers to evaluating the amount specific features contributed to the function of an AI program. In this project, the function is classification and the feature importance can be evaluated directly from the XGBoost classifier after it has been used to provide the two probability values necessary for

PU learning. Evaluating features is done after an AI program makes its predictions to estimate which ones contributed the most and how much.

In this project, the permutation feature importance technique was used. It is a model inspection technique especially useful for non-linear or opaque estimators like XGBoost (Buitinck et al 2013). The permutation feature importance is defined to be the decrease in a model score when a single feature value is randomly shuffled, and this technique is able to find it by breaking the relationship between the feature and the target where the new model score indicates how much the model depends on that feature (Breiman 2001). This technique can be calculated many times with different permutations of the feature (Buitinck et al 2013). Through the Scikit-learn module inspection, the permutation_importance function is accessible which calculates the feature importance of a classifier for a given dataset (Buitinck et al 2013). The validation performance for this technique is measured by the R^2 score where it is significantly larger than the chance level, which allows the permutation_importance function to identify which features contribute most to the model predictions (Buitinck et al 2013). However, it is necessary to acknowledge that features considered insignificant for a bad model may be important for a good one (Buitinck et al 2013). This is also why k-fold cross validation is employed because it provides another measure of the predictive power of the PU learning model (Buitinck et al 2013). Permutation importance itself does not reflect the direct predictive value of a feature itself, it only scores how important this feature is for a specific model (Buitinck et al 2013).

Feature Analysis

There are two sources of where the features are derived: the ProtParam module from BioPython, and the PyProtein module from PyBioMed. For a single amino acid sequence, there are 9,810 features derived. There are a total of 19,620 features per protein pair because the first 9,810 features are for the target protein, and the remaining 9,810 features are the delta features as described earlier. There are a total of twelve categories of features where the first eleven are derived from functions in the PyProtein module, and the last category of features is derived from the ProtParam module.

There were a total of seven ProtParam functions used for the last category of features. The first function is molecular_weight(), which takes in an amino acid sequence as a string and returns a float value of the protein's molecular weight in Daltons (Cock et al 2009). The second

function is instability_index(), which returns a float value corresponding to the instability index (Cock et al 2009). This is an implementation of the method of predicting stability characteristics from the paper *Correlation between stability of a protein and its dipeptide composition: a novel approach for predicting in vivo stability of a protein from its primary sequence* (Guruprasad et al 1990). This method tests a protein for stability using previously calculated weight values of instability per amino acid dipeptide where a resulting value above 40 means the protein has a short half-life and could therefore indicate how unstable the protein is (Guruprasad et al 1990). The third function is secondary_structure_fraction(), which provides protein structural information with three values correlating to the fraction of residues common in a helix like V, I, Y, F, W, L, residues common in a turn like N, P, G, S, or residues like E, M, A, L that are common in a sheet (Cock et al 2009). The fourth function is molar_extinction_coefficient() which returns two values (Cock et al 2009). The two values include the molar extinction coefficient where the Cysteine residues are assumed to be reduced and the molar extinction coefficient assuming there are Cysteine-Cysteine bonds (Cock et al 2009). This information is useful since it provides information on the structure integrity of the protein because a higher coefficient corresponds with residues critical to protein structure and function like Tryptophan, Tyrosine, and Cysteine (Pace et al 1995). The fifth function is gravy() which returns a float value that is a measure of a protein's hydrophobicity or hydrophilicity (Cock et al 2009). If a protein is found to be more hydrophilic with the GRAVY number, that could indicate that it is inherently unstable (Kyte and Doolittle 1982). The sixth function is isoelectric_point() which returns a float value of the protein's calculated isoelectric point, which is the pH at which a protein has no net electrical charge making it neutral (Shaw et al 2001). The closer a protein's isoelectric point is to the pH of the normal cellular environment, which is between 7.0 and 7.4, the less likely it is to remain stable as it is less soluble which could lead to its degradation thus making this a useful feature to know (Shaw et al 2001). The seventh function is aromaticity(), which returns a float value of the protein's calculated aromaticity (Cock et al 2009). Aromaticity is the relative frequency of the three residues Phe+Trp+Tyr (Anjana et al. 2012). This is relevant because aromatic interactions have been found to play an important role in maintaining the overall structure of the protein molecules and protein-DNA complexes and are integral for the proper functioning of many protein molecules (Anjana et al. 2012).

There are eleven functions used in the PyProtein module that fill in the remaining features. The first three functions are GetAAComp(), GetDPComp(), and GetTPComp() provide a total of 8,420 features that contain the percentages of all amino acids, dipeptides, and tripeptides inside a given protein (Dong et al 2018). This does not necessarily provide direct information about protein structure, but it is possible that these compositions may show a pattern relating to structural stability that the PU learning model may be able to identify. The fourth, fifth, and sixth functions were GetMoreauBrotoAuto(), GetMoranAuto(), and GetGearyAuto which return dictionaries with Moreau-Broto, Moran, and Geary autocorrelation descriptors for the given property providing a total of 720 features (Dong et al 2018).

Autocorrelation descriptors are a class of topological descriptors that describe the correlation between two objects like proteins or peptide sequences in terms of their specific structural or physicochemical properties (Ong et al 2007). The eight amino acid properties used for deriving the autocorrelation descriptors are hydrophobicity scale, average flexibility index, polarizability parameter, free energy of amino acid solution in water, residue accessible surface areas, amino acid residue volumes, steric parameters, and relative mutability (Ong et al 2007). Moreau-Broto autocorrelation uses just these property values as the basis for measurement, Moran autocorrelation uses property value deviations from the average property values, and Geary autocorrelation utilizes the square-difference of property values (Ong et al 2007).

The eighth function used was GetCTD(), which returns a dictionary with Composition, Transition, and Distribution descriptors providing a total of 147 features (). This calculates all Composition, Transition, and Distribution, CTD, descriptors based on seven different properties of amino acid dipeptides: polarizability, solvent accessibility, secondary structure, charge, polarity, normalized van der Waals volume written as VDWV, and hydrophobicity (Ong et al 2007). For each of these properties, the amino acids are split into three groups so that the amino acids in each group have approximately the same property (Ong et al 2007). For instance, residues can be divided into hydrophobic; CVLIMFW, neutral; GASTPHY, and polar; RKEDQN, groups (Ong et al 2007). Composition, C, is the number of residues with that particular property divided by the total number of residues in a given protein sequence (Ong et al 2007). Transition, T, shows the percent frequencies of residues with a particular property being followed by residues of a different property (Ong et al 2007). Distribution, D, measures the chain

length where the first, 25%, 50%, 75% and 100% of residues with a particular property are located (Ong et al 2007).

The eight and ninth functions used were GetPAAC() and GetAPAAC() which return dictionaries with Type I Pseudo amino acid composition descriptors and Amphiphilic, Type II, Pseudo amino acid composition descriptors respectively (Dong et al 2018). These two functions contribute a total of 70 features. Type I Pseudo amino acid composition descriptors are derived from a method to predict the cellular attributes of a protein based on its amino acid sequence (Chou 2001). It is a combination of a set of the 20 components of the conventional amino acid composition and discrete sequence correlation factors where values are derived for hydrophobicity, hydrophilicity, and side-chain mass (Chou 2001). The Type II amino acid composition descriptors are relevant in the same way because they also contain components of the conventional amino acid composition and a set of correlation factors that reflect different hydrophobicity and hydrophilicity distribution patterns along a protein chain (Chou 2004). The difference between Type I and Type II is that Type II only uses correlation factors that reflect hydrophobicity and hydrophilicity properties whereas Type I uses correlation factors that reflect hydrophobicity, hydrophilicity, and residue mass. These functions were used since it has been shown that the hydrophobicity, hydrophilicity, and residue mass arrangement shown in amino acid residues along a protein chain can play an important role in protein folding and a protein's interaction with other molecules and catalytic mechanisms (Chou 2004).

The tenth function used was GetQSO() returns a dictionary with Quasi sequence order descriptors (Dong et al 2018). This function uses a set of sequence-order-coupling numbers derived from the physicochemical distances between amino acids that reflect the sequence order effect also known as the quasi-sequence-order effect (Chou 2000). This function was initially made to predict the subcellular location of a protein given its amino acid sequence, but this can be used in this project since this function uses amino acid physicochemical distances to infer protein surface physical chemistry character, a structural feature which relates to structural stability (Chou 2000). This function contributes 100 features.

The eleventh and last function from PyProtein used for feature acquisition was GetTriad() which provides 343 features (Dong et al 2018). This function returns a dictionary with the conjoint triad features calculated from the protein sequence (Dong et al 2018). This function that returns conjoint triad features was initially made to predict PPIs based on amino acid sequence

(Shen at al 2007). This function considers the properties of one amino acid and its adjacent amino acids and considers any three continuous tripeptides as a triad unit (Shen at al 2007). PPIs primarily consist of electrostatic interactions, including hydrogen bonding, and hydrophobic interactions (Shen at al 2007). These types of interactions are influenced by dipoles and volumes of the amino acid side chains, therefore allowing the 20 amino acids to be clustered into seven classes based on these features and accordingly what types of PPIs they could influence (Shen at al 2007). The triads can be differentiated according to these classes of amino acids, because triads that contain amino acids of the same class are known to play similar roles in PPIs (Shen at al 2007). A total of 343 tripeptide percent compositions are returned by this function because there are 343 types of tripeptide combinations possible using the seven amino acid PPI classes (Shen at al 2007).

Project Pipeline

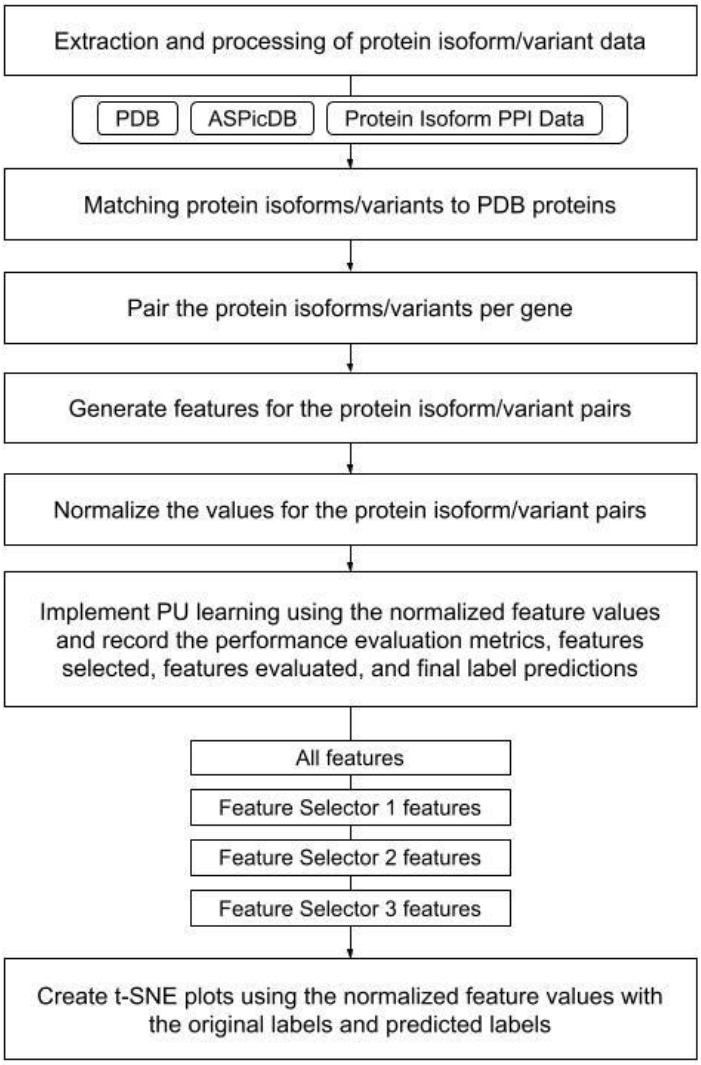


Figure 2. The Protein Data Processing Pipeline; a flowchart showing the chronological steps for processing the protein isoform data in this project.

The process of how the results for this project followed a simple pipeline process that is eight full steps as shown in Figure 2. As specified in Figure 2, the steps in order are extracting the protein isoforms/variants for training and testing, matching of protein isoforms/variants to known stable proteins in PDB, pairing of the protein isoforms/variants per gene, generating features for the protein isoform/variant pairs, normalizing those feature values, entering the normalized features into the PU learning model to get the final predictions with performance

metrics, and finally making t-SNE plots with the original labels and the labels predicted by the PU learning model.

The first step of protein isoform/variant extraction refers to the initial processing of PDB protein data, ASPicDB protein data, and the PPI data from the paper *Widespread expansion of protein interaction capabilities by alternative splicing* as specified earlier. The data from PDB and ASPicDB were first received in FASTA files. Using the BioPython module SeqIO and the Python Pandas module, the FASTA files were read into separate Comma-Separated Value files, csv files. The first column corresponded to the protein name as provided by ASPicDB or PDB, and the second column was the amino acid sequence. The gene symbol for each protein in ASPicDB was provided in the protein name. For the PDB proteins, the organism of the protein was specified in the name so only the proteins of homo sapien origin were kept. In both data sets, the FASTA files were read into csv files whilst ensuring that there were no duplicate entries and that no proteins were listed if their amino acid sequences were identical to a previously listed protein sequence. For the first data set from *Widespread expansion of protein interaction capabilities by alternative splicing*, it was provided in an Excel file with eight columns. The second column contains the gene symbols corresponding directly to the amino acid sequences in the third column, the fifth column contains the gene symbols corresponding directly to the amino acid sequences in the seventh column, and the eighth column contained the symbols '+' and '-' where '+' indicates a successful PPI between the two proteins and '-' indicates no PPI between the two proteins (Yang et al 2016). In this situation, if a PPI exists between two proteins, both proteins in the pair are structurally stable (Yang et al 2016). Using this logic, it was determined that each protein in this file was structurally stable and placed into separate csv with three columns where the first is all the distinct protein sequences, the second column contains the corresponding genes, and the third column contains a repeat of positive labels of 1.

The second step refers to the usage of PDB to verify which proteins in ASPicDB have sequences corresponding to proteins that have already been confirmed as structurally stable. This process of matching with ASPicDB proteins was done through a program in Python. The program took in an input of a csv file, which contained the names, amino acid sequences, and genes of the proteins from ASPicDB. The first step of the program was to create a list of all the PDB human protein names and a list of the corresponding PDB amino acid sequences from another csv file. Next, the amino acid sequences for each protein isoform/variant from the input,

ASPicDB, were read into a list along with the protein names provided by ASPicDB. Then, using the list of ASPicDB names, a list of the genes corresponding to the list of ASPicDB amino acid sequences was made. The program then iterates through each sequence in this list of ASPicDB protein isoform/variant sequences. During each iteration, the program checks if the current input sequence matches anything in the PDB list. If it does, then the information for this input sequence is added to a new output csv file with three changes. First, an extra column is added which contains binary labels of 0s and 1s where 1 indicates a structurally stable protein, and 0 indicates a protein of unknown structural stability. Second, the name for the matched PDB protein is combined with the name provided by ASPicDB for the corresponding sequence. And third, another column is added with the gene corresponding to the original ASPicDB protein. If no match in PDB is found, then the information corresponding to this sequence is added to the output csv file with a 0 for the protein label. Furthermore, this matching process imposed rules such that the output csv file would not contain any redundant information nor would there be any positive proteins added if the sequences were already listed in the first data set from *Widespread expansion of protein interaction capabilities by alternative splicing*.

The third step refers to the process of protein isoform/variant pairing. The protein isoforms/variants are paired by gene to avoid any potential overlap in feature values. The gene symbols were thankfully provided in both data sets. For the ASPicDB proteins that were matched, the pairing was done through another Python program. This program input is the csv file that was the output of the matching program. This program creates a list of the proteins that were labeled 1, signifying that a protein is structurally stable, and makes a list of all the distinct genes within that data set that have a structurally stable protein. Using the list of distinct genes with a stable protein, the list of all stable proteins is processed such that there is only one structurally stable protein per gene. This processing is done exactly the same for the proteins labeled 0, signifying that the structural stability of a protein is unknown. Then for each protein in the processed list of the structurally stable proteins, the protein isoform/variant pairing occurs with a protein of unknown stability that is of the same gene. The first protein in the pair is the unknown target protein. This protein is the target because the structural stability of this protein is what the PU learning model will predict. The second protein in the pair is the stable known reference protein from the same gene as the target protein, which will provide information that will help the PU learning model make better predictions. The pairs were output in a new csv file

with three columns. The first column specifies the target protein sequence, the second column specifies the reference protein sequence, and the third column specifies the structural stability label of the target protein. This means that all the pairs from the ASPicDB data set were correctly labeled 0 indicating the target proteins of all those pairs are of unknown structural stability. The first data set from *Widespread expansion of protein interaction capabilities by alternative splicing* had the data pairing done with the same program, but with an additional input specifying that known stable proteins can be the target protein. This was done with this first data set after it was realized that the list of distinct protein sequences from this data set was the same exact list of protein sequences corresponding to proteins that had at least one PPI. Therefore, the first data set contained all the positive samples, or structurally stable protein pairs. The output csv files for the protein isoform/variant pairs from the ASPicDB data set and the protein isoform/variant pairs from the first data set were combined into one csv file creating a full data set of labeled and unlabeled protein pairs.

For the fourth step, features relevant to structural stability were generated. The features and corresponding methods used to derive them were stated earlier. The features were derived using the amino acid sequence of the target protein and the reference protein in the protein pair. But for that pair, the target protein features and the delta features were only recorded. The delta features are the feature values found by subtracting the unknown target protein feature values from the feature values of the reference known stable protein in the pair. As stated before, this is done to improve classification accuracy by ensuring that feature values indicating structural stability/instability can be more easily identified seeing the contrast between what could be a negative or structurally unstable protein and a positive or structurally stable protein from the same gene. Using Python, a feature generator was created using the feature-creating functions previously mentioned and the Pandas module to better deal with csv files. This feature generator would be able to take a csv file from the third step and output a csv file which had the same amount of rows as the input csv file which corresponds to the number of distinct protein pairs, and 19,621 columns. The first 9,810 columns are for the original feature values of the target protein. The next 9,810 columns are for the delta features. And the last column is the label for the target protein in each pair.

The next step involves normalizing the feature values in the output csv file in the fourth step. This fifth step was again done in Python with a function that used the Pandas module and

the Scikit-learn module preprocessing (Buitinck et al 2013). This function took in the output csv file of feature values from the fourth step and returned a csv file with the feature values set between 0 and 1. This was done per column/feature using the lowest value as reference for 0 and the highest value as a reference for 1. This was done for two reasons. The first is that a reliable t-SNE plot could not be made without normalized values (Buitinck et al 2013). And the second reason is because it would allow for the PU learning model to have an easier time processing the feature values since the variation in feature values is kept, but not the actual values which may contain outliers of troublesome magnitude that would otherwise give problems to the PU learning model.

The sixth step will be to run the PU learning model onto the csv file with the normalized feature values. A full run had the provided ten recall values from the ten k-folds recorded. The bias-corrected evaluation metrics for the final predictions made were recorded. Then features returned from the feature importance implementation will be recorded with the given metric for its contribution to the final predictions. And finally, the PU learning model's predictions will be recorded. There were four full runs. The first run was with all the features. The second, third, and fourth runs employed the use of a distinct feature selector described above where the features used in these runs were recorded.

The seventh and final step is using the output csv file with normalized feature values from the fifth step to create a t-SNE plot and then to create another t-SNE plot using the predicted labels from the sixth step. This will be done for each full run using the PU learning model. This was done such that the clustering of data could be visualized to verify whether the feature information was enough to potentially distinguish between what could be the true positives and the true negatives. Furthermore, this use of the t-SNE plot with the original labels may provide an insightful contrast to the plot with the labels from the PU learning model's predictions. This may allow for a visual evaluation of the PU learning model's performance/predictions.

Results

In order to evaluate the effectiveness of the PU learning model, the datasets consisting of positive data and unlabeled data were combined and used as the testing and training data as specified in the Methodology. Furthermore, using three feature selector functions, three distinct feature subsets of the data were made and run through the PU learning model. Due to the unlabeled data, the evaluation metrics for the four runs were not exact, but these estimations have proven to be relatively accurate (Ramola et al 2019). In order to combat this, t-SNE plots of the data with original labels and predicted labels were made to provide a visual analysis that could inform whether the model made accurate predictions depending on how the new labels follow the clustering patterns. Additionally, the feature importance was calculated for each of the four runs to better understand which features contributed most to the model and how much (Breiman 2001). This can infer how the model made its predictions given the feature data it was supplied. And finally, to counterbalance the uncertainties provided by the evaluation metric estimates, case studies on relevant genes were conducted which also demonstrates whether the PU learning model has a real world application in predicting protein isoform/variant stability.

PU Learning Evaluation

The first run had the entirety of the positive data and unlabeled data run through the PU learning model with all features. As is standard per run in this PU learning model, k-fold cross validation was implemented with ten folds.

Table 4. Evaluation Metrics for Data with All Features

	k-fold value Average	k-fold Standard Deviation	Final Value
Recall	0.9535	0.0139	1.0
BC Accuracy estimate	1.0	0	1.0
BC Balanced Accuracy	1.0	0	1.0
BC F score	1.0	0	1.0
BC Matthews Correlation Coefficient	1.0	0	1.0

In Table 4, the recall and the bias-corrected evaluation metrics were recorded for the folds and the final predictions. With the SCAR assumption made, it is possible to consider the positive labeled data as a proper representative of the entire distribution of the true positives in the data set. The SCAR assumption is further enforced by how the testing subset per fold were randomly generated subsets of the labeled positives. Therefore, the recall values recorded per fold can be considered representative for the entirety of the true positives in the data set. Table 4 shows that the average k-fold recall value was high at 0.9535 with a low standard deviation of 0.0139. Furthermore, the bias-corrected evaluation metric estimates were also calculated per fold with the averages for accuracy, balanced accuracy, F score, and the Matthews Correlation coefficient unanimously coming out to perfect values of 1.0 with no standard deviation values above 0. The final recall and bias-corrected evaluation metric estimates were all marked as perfect at 1.0. It should be clarified that the PU learning model heavily adjusts its label probabilities based on the testing data composed of just positive labeled data. In this traditional PU learning implementation, it considers the unlabeled samples as negative which typically provides a

negative bias for the classification of unlabeled samples hence why the bias-corrected estimates were used. However, the bias-corrected estimates are exactly that.

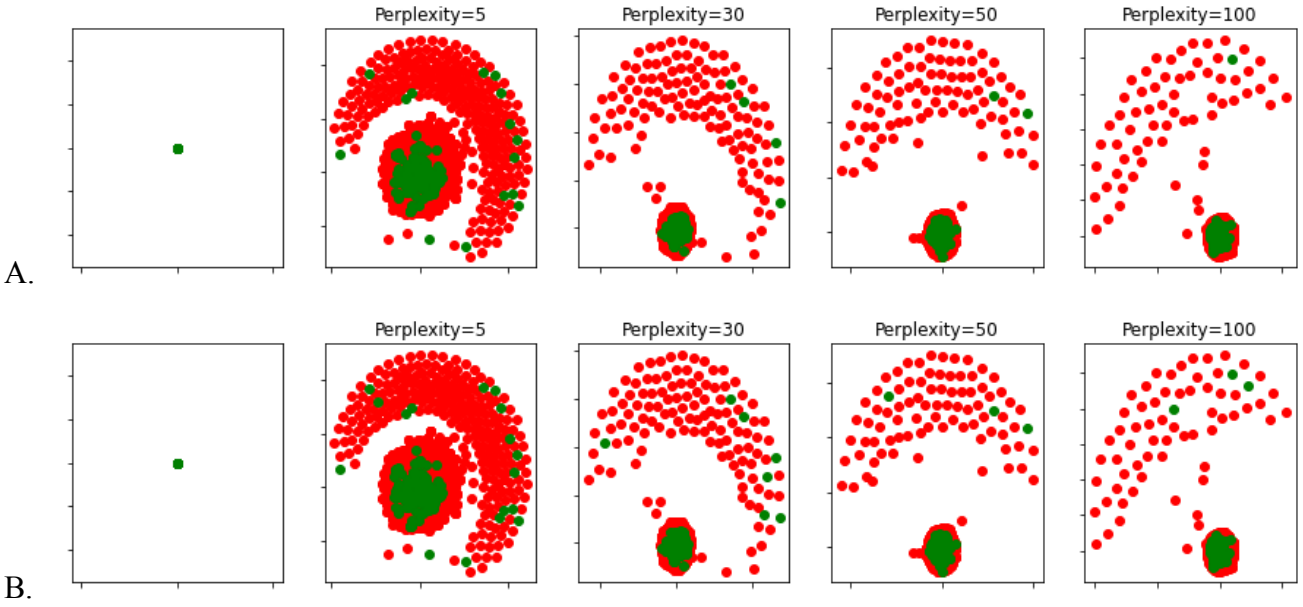
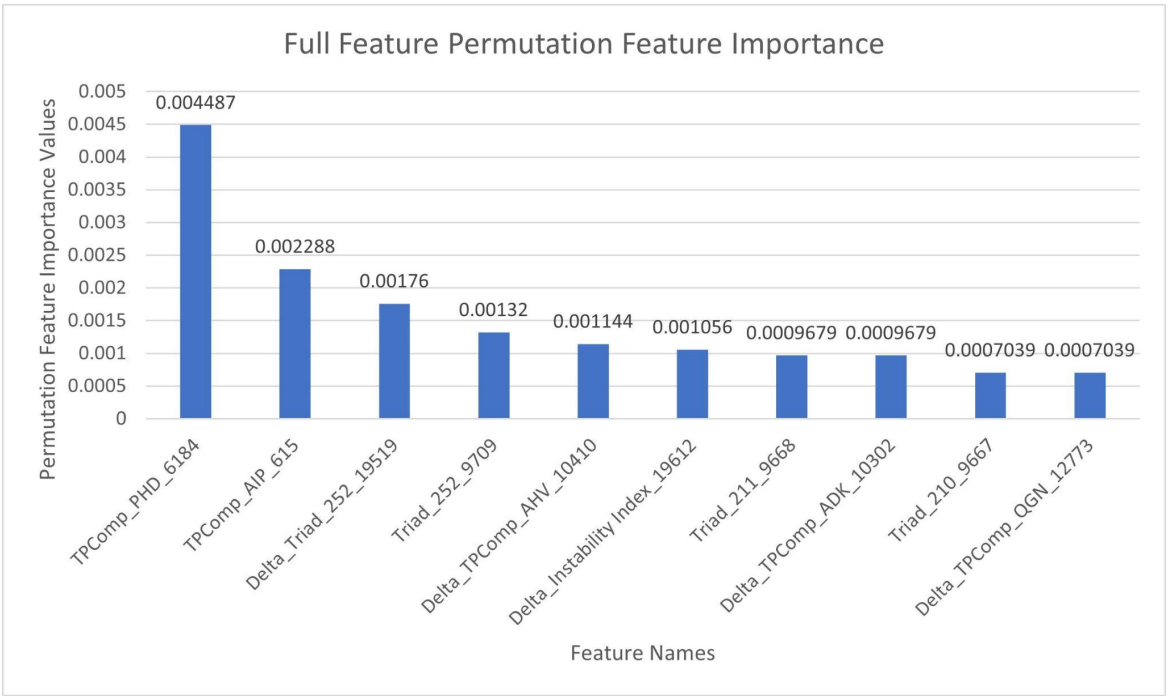


Figure 3. t-SNE plot of data with all features. Panel A is the plot with the original labels. Panel B is the plot with the labels predicted by the PU learning model. 27 total new predicted positives.

Figure 3 is an attempt to compensate for the lack of exact evaluation metrics for the PU learning model. Panel A of Figure 3 is a t-SNE plot of the data set with all features where the green dots are the positive labeled data samples and the red dots are the unlabeled data samples. In panel B of Figure 3, the t-SNE plot is made the same except that the labels for the data samples are provided by the final prediction of the PU learning model. In panel B, there is a slight increase in green dots mainly where the unlabeled samples were in panel A that is noticeable in all complexities. Yet, at the lower complexities of 5 and 30, a small increase of green dots can be seen where the positively labeled data was previously clustering in panel A.

A.



B.

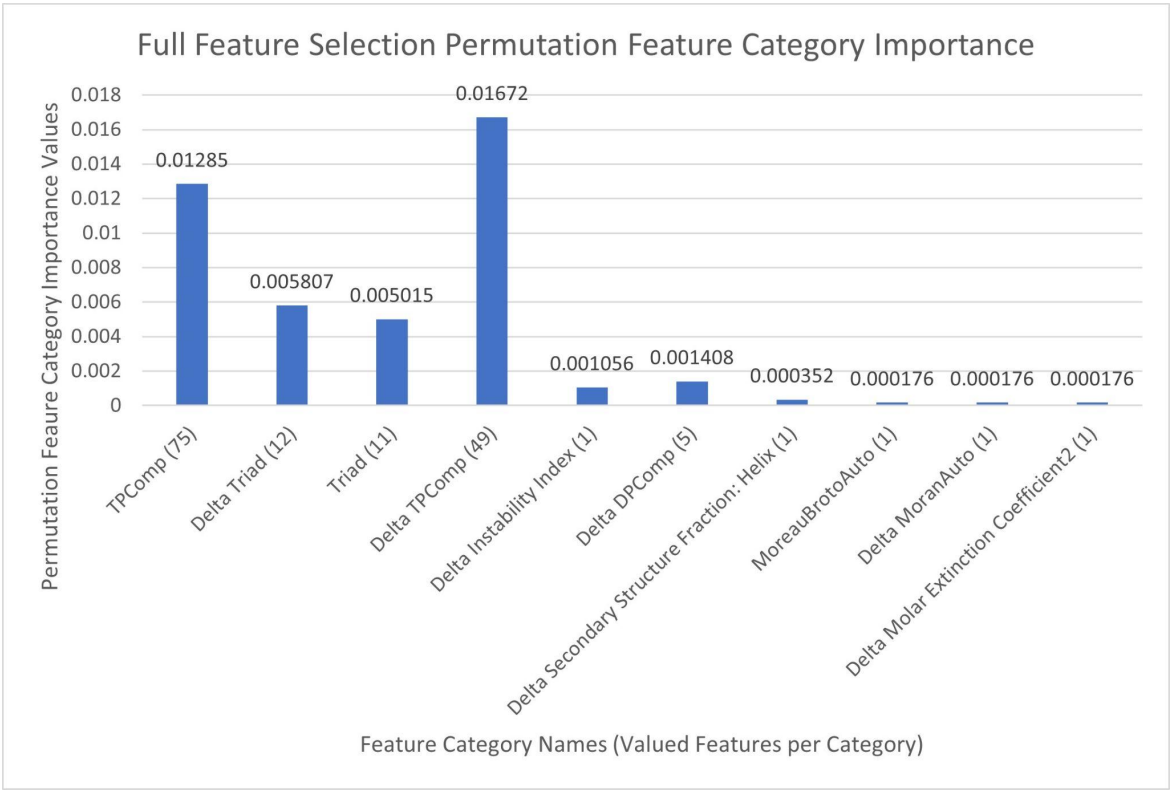


Figure 4. Permutation feature importances for data with all features

Figure 4 presents the permutation feature importance values the PU learning model assigned the data set features when all the features were used by the model. Panel A presents a

bar graph with the top ten individual features according to their permutation feature importance. Panel B shows the feature categories of all the features whose permutation importance was above zero. Panel A provides insight on the specific feature rankings whereas panel B provides a more accurate depiction of what type of protein structural data influenced the PU learning model the most. In both panels, tripeptide composition and delta tripeptide composition predominantly contributed the most to the model contrary to what was expected to be features specifically relevant to protein structure such as instability index and the molecular extinction coefficients.

The second run of the PU learning model employed the usage of the features selected by the first feature selector function which was influenced by a user specified input of 2,000 to specify the number of features to be returned according to the highest chi squared values (Buitinck et al 2013). This run along with all the other runs followed the same process of k-fold cross validation.

Table 5. Evaluation Metrics for Data with Features from First Feature Selector

	k-fold value Average	k-fold Standard Deviation	Final Value
Recall	0.9535	0.0140	1.0
BC Accuracy estimate	1.0	0	1.0
BC Balanced Accuracy	1.0	0	1.0
BC F score	1.0	0	1.0
BC Matthews Correlation Coefficient	1.0	0	1.0

Table 5 has the same exact content as Table 4. The average k-fold recall was surprisingly the same as the one in Table 4 at 0.9535. The k-fold recall standard deviation in Table 5 is

unremarkably higher. The k-fold average values for the accuracy, balanced accuracy, F score, and Matthews Correlation estimations were again all 1.0 in Table 5, with no standard deviation values above 0 just like Table 4. Furthermore, the final recall and bias-corrected evaluation metric estimates were all 1.0 in Table 5.

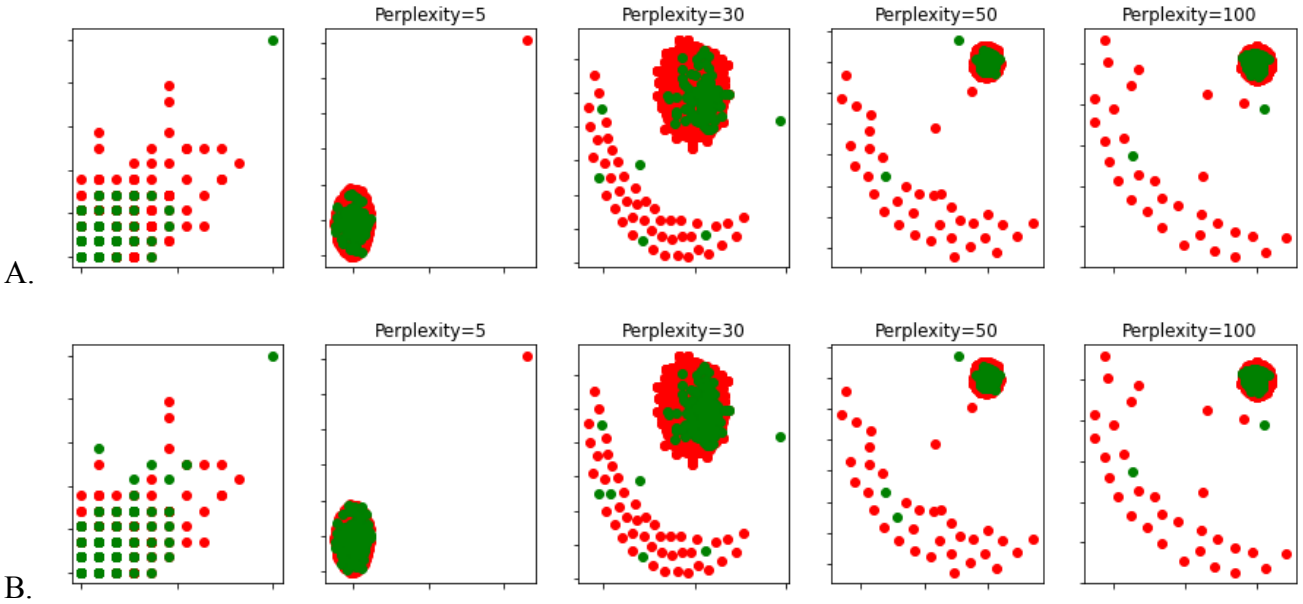
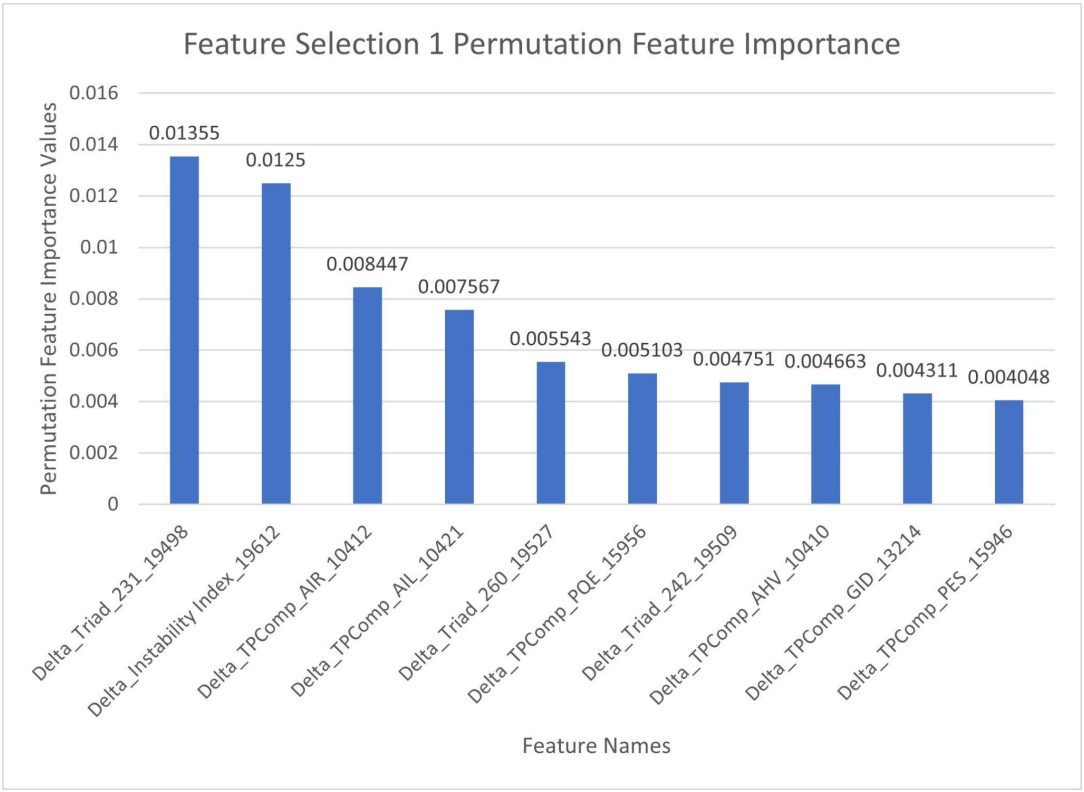


Figure 5. t-SNE plot of data with features from the first feature selector. Panel A is the plot with the original labels. Panel B is the plot with the labels predicted by the PU learning model. 88 total new predicted positives.

In Figure 5, a t-SNE plot of the data with the features chosen by the first feature selector function with the original labels is in panel A, which is contrasted by panel B that is another t-SNE plot that has the same features but with the predicted labels. It can be clearly seen at complexities 0, 5, and 30 that the cluster of positive samples in panel A become larger and more concentrated in panel B for Figure 5.

A.



B.

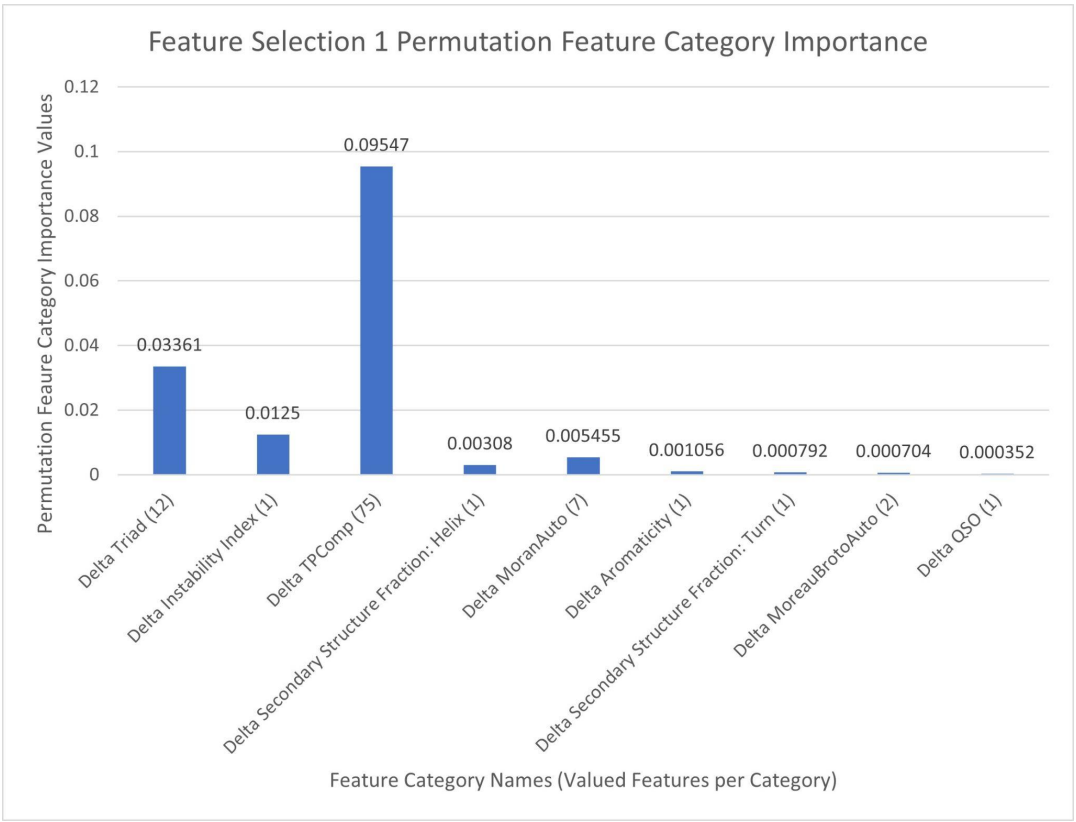


Figure 6. Permutation feature importances for data with features from first feature selector

Figure 6 shows the permutation feature importance values the PU learning model assigned the data set features with the feature selection provided by the first feature selector function. Like Figure 4, panel A in Figure 6 is a bar graph with the top ten individual features according to the permutation feature importance assigned by the model. Panel B shows the feature categories of all the features whose permutation importance was above zero for this feature selection. Much like Figure 4, panel B demonstrates a surprising reliance the model had on a feature category with many features including the delta tripeptide compositions and the delta triad values. However, panel A displays on the individual level, the second most important feature was the delta instability index only surpassed by a delta triad feature. The rest of the top scoring features in panel A consisted of delta triad and delta tripeptide composition features. Despite this pattern persisting from Figure 4, the delta instability index scoring as high as it did was the first instance of a feature being assigned an importance that was expected.

Table 6. Evaluation Metrics for Data with Features from Second Feature Selector

	k-fold value Average	k-fold Standard Deviation	Final Value
Recall	0.9521	0.0176	1.0
BC Accuracy estimate	1.0	0	1.0
BC Balanced Accuracy	1.0	0	1.0
BC F score	1.0	0	1.0
BC Matthews Correlation Coefficient	1.0	0	1.0

Table 6 follows suit with Table 4 and Table 5. The average k-fold recall was a bit less at 0.9521. The k-fold recall standard deviation in Table 6 is the highest at 0.0176. The k-fold average values for the accuracy, balanced accuracy, F score, and Matthews Correlation estimations were all 1.0 in Table 6 as well. Like Table 4 and Table 5, there were no standard deviation values above 0 for the metric estimates. In accordance with Table 4 and Table 5, the final recall and bias-corrected evaluation metric estimates were also 1.0 in Table 6.

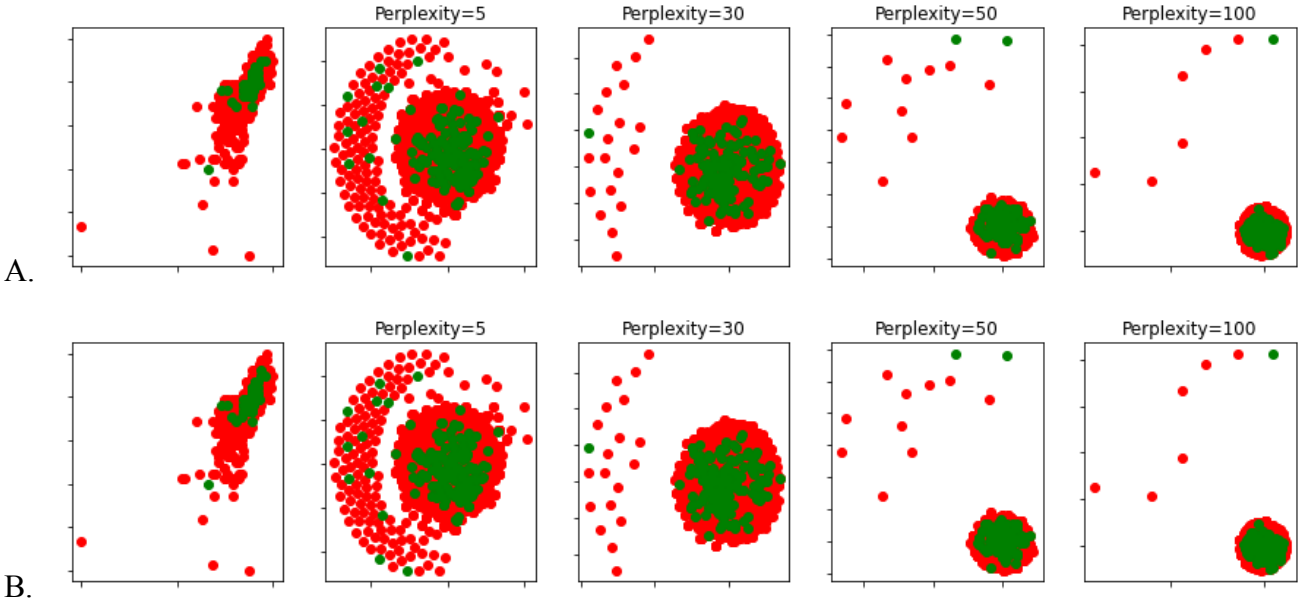
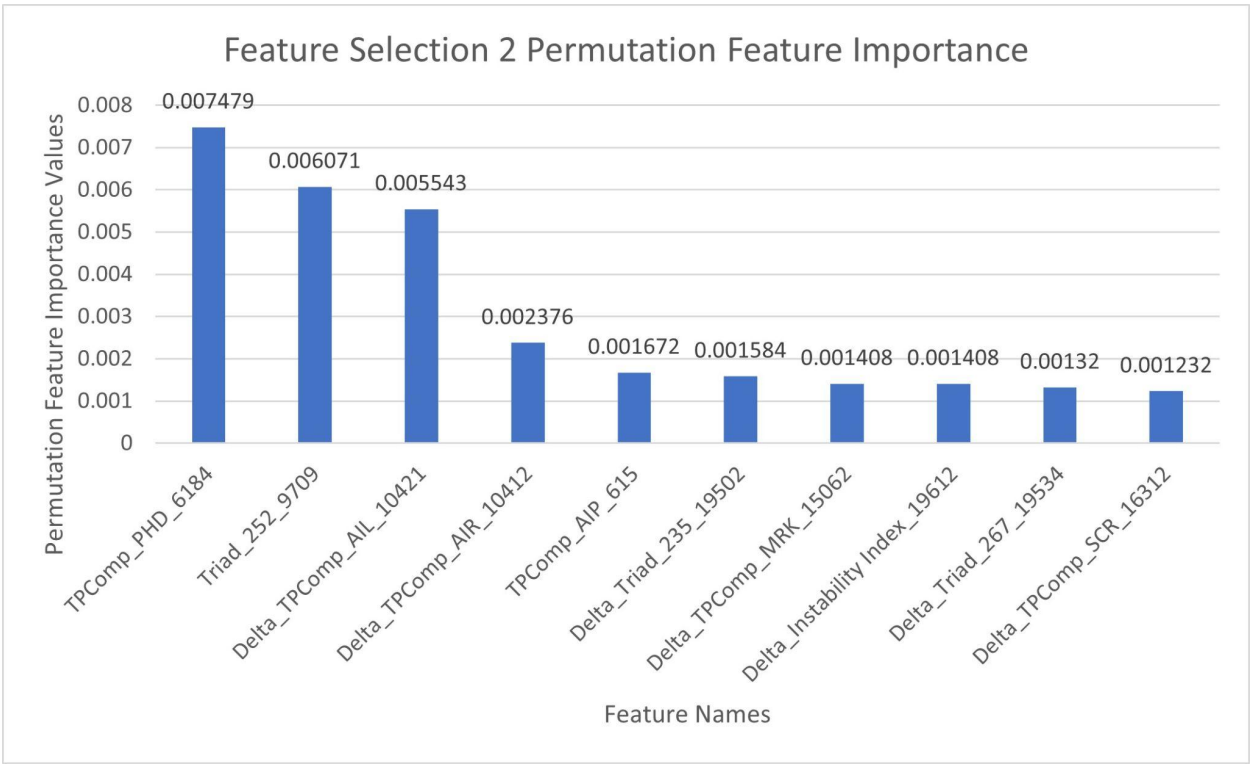


Figure 7. t-SNE plot of data with features from the second feature selector. Panel A is the plot with the original labels. Panel B is the plot with the labels predicted by the PU learning model. 30 total new predicted positives

Figure 7 like Figure 5 and Figure 3 shows a t-SNE plot of the data with the features chosen by the second feature selector function with the original labels is in panel A and panel B has the same data samples plotted but with the predicted labels. In complexities 5, 30, 50, and 100 in Figure 7, it is not clear whether the primary clustering of the original positively labeled data represented by the green dots shows any increase or change. Only at the complexity of 0 is there a minuscule increase from panel A to panel B in the amount of green dot/positive data clustering.

A.



B.

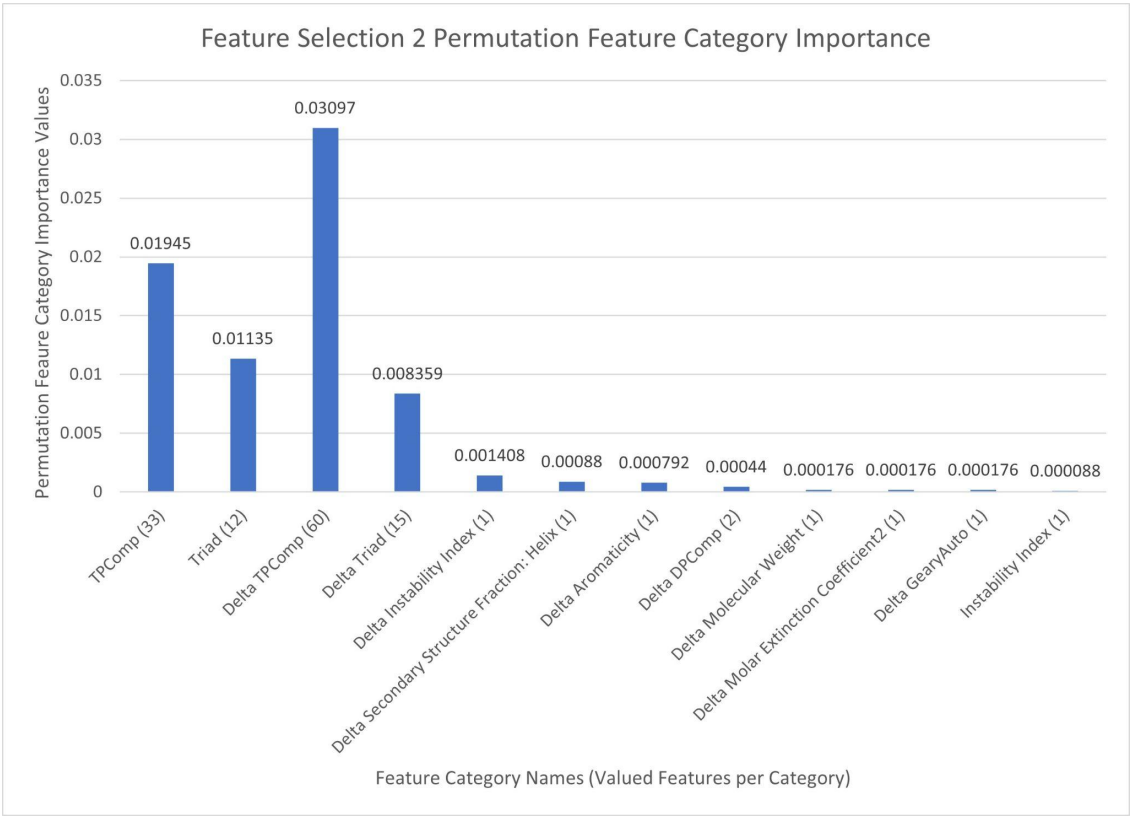


Figure 8. Permutation feature importances for data with features from second feature selector

Figure 8 shows the permutation feature importance values the PU learning model assigned the data set features with the feature selection provided by the second feature selector function like Figure 6 with the second feature selection function. Panel A in Figure 8 is a bar graph with the top ten individual features according to the permutation feature importance and Panel B shows the feature categories of all the features whose permutation importance was above zero for this feature selection. Similar to Figure 4 and Figure 6, panel B shows that even in this distinct selection of features, the model highly values the delta tripeptide compositions and the tripeptide composition values. Unlike Figure 6, the most valuable features in panel A of Figure 8 consist predominantly of tripeptide composition, delta tripeptide, and triad features.

Table 7. Evaluation Metrics for Data with Features from Third Feature Selector

	k-fold value Average	k-fold Standard Deviation	Final Value
Recall	0.9500	0.0135	1.0
BC Accuracy estimate	1.0	0	1.0
BC Balanced Accuracy	1.0	0	1.0
BC F score	1.0	0	1.0
BC Matthews Correlation Coefficient	1.0	0	1.0

Table 7 is most similar to Table 6 which makes sense considering the results in Table 6 are directly influenced by the feature selection provided by Extra Trees Classifier and Table 7 by the similar Random Forest Classifier. The average k-fold recall was lowest in Table 7 at 0.9500. The k-fold recall standard deviation in Table 7 is the lowest at 0.0135. The k-fold average values for the accuracy, balanced accuracy, F score, and Matthews Correlation estimations were all 1.0

in Table 6 as well. Like Table 4, Table 5, and Table 6, there were no standard deviation values above 0 for the metric estimates in Table 7. And as expected, the final recall and bias-corrected evaluation metric estimates were also 1.0 in Table 7.

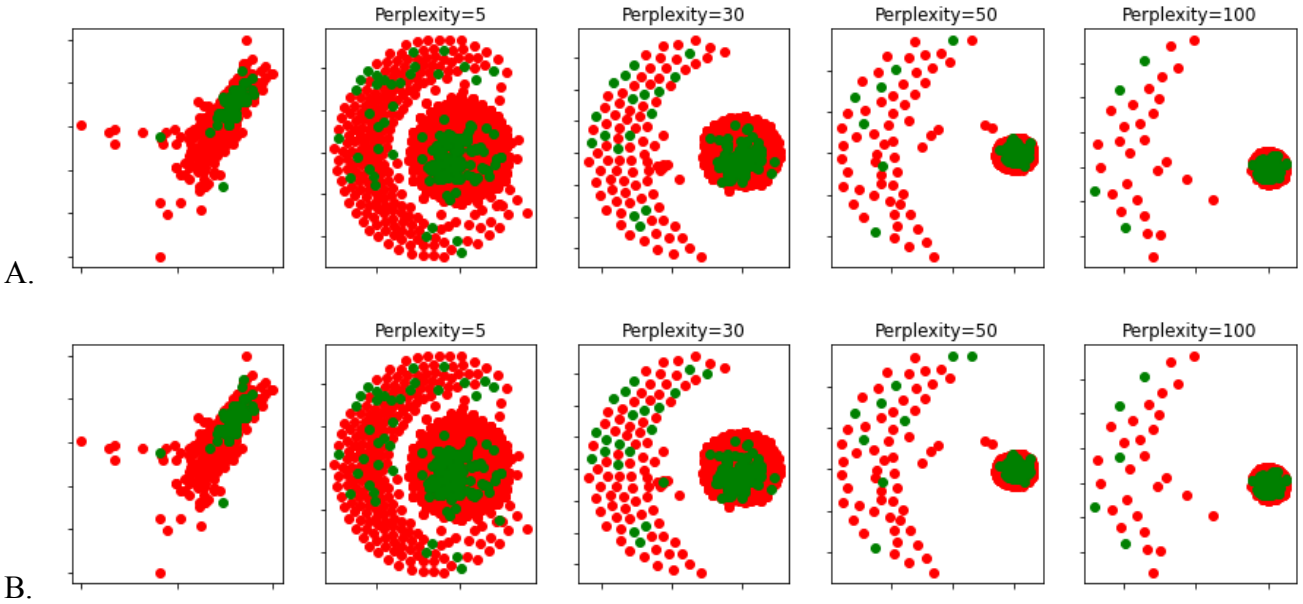
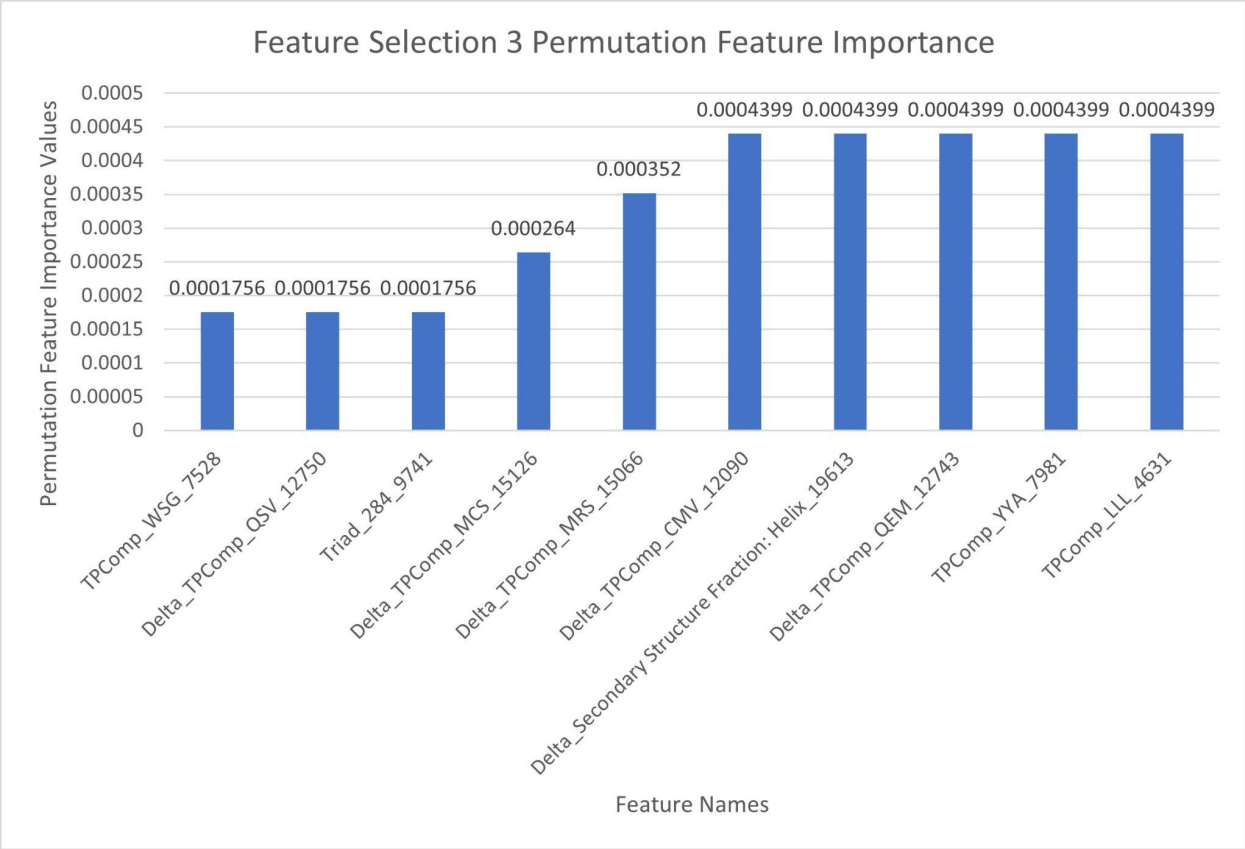


Figure 9. t-SNE plot of data with features from the third feature selector. Panel A is the plot with the original labels. Panel B is the plot with the labels predicted by the PU learning model. 30 total new predicted positives.

Figure 9 is another t-SNE plot of the data with the features chosen by the third feature selector function with the original labels is in panel A and panel B has the same data samples plotted but with the predicted labels. Figure 9 is most similar to Figure 7 meaning that for all complexities in this figure, there is no obvious change or increase in the primary clustering of the original positively labeled data represented by the green dots.

A.



B.

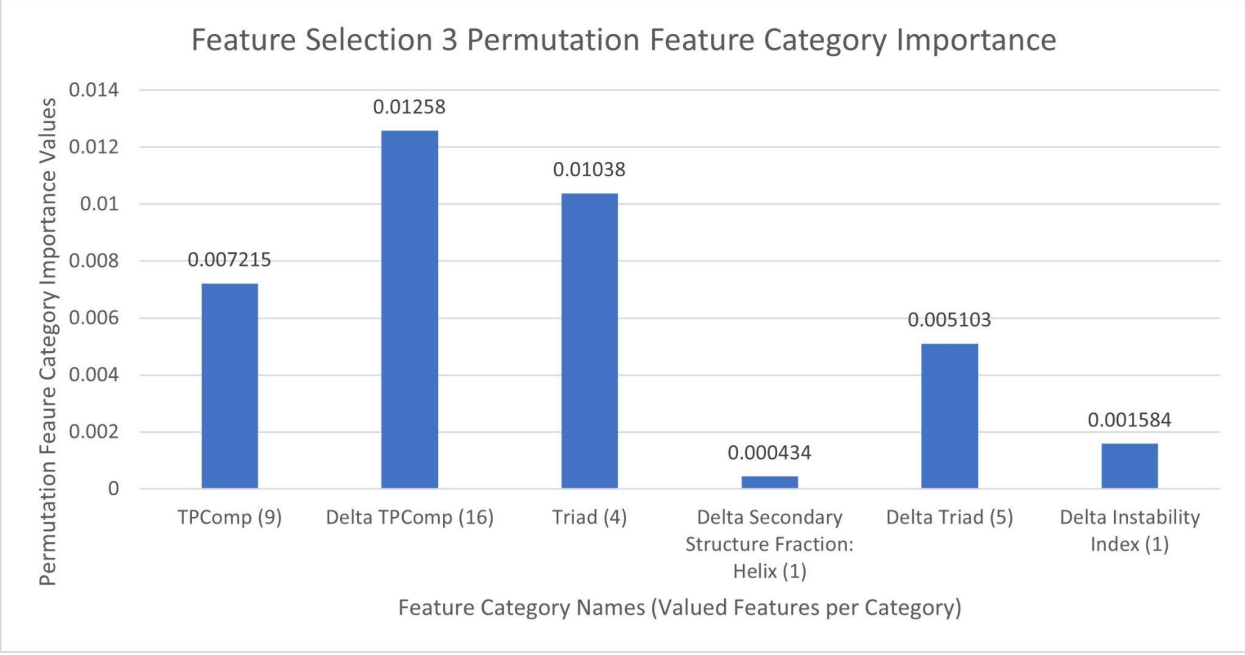


Figure 10. Permutation feature importances for data with features from third feature selector

Figure 10 shows the permutation feature importance values the PU learning model assigned the data set features with the feature selection provided by the third and final feature selector function. Panel A in Figure 10 is a bar graph with the ten highest valued individual features according to the permutation feature importance with Panel B showing the sum feature importances for the feature categories. Again, as seen in Panel A, most of the highest importance features belong to the tripeptide and delta tripeptide feature categories. The delta secondary structure fraction corresponding to protein helix structures was ranked high in panel A, yet panel B shows the entire picture where the feature categories that contribute the most to the predictions of the PU learning model were again tripeptide composition, delta tripeptide composition, and triad.

Case Studies

Case Study 1: CFTR

Cystic Fibrosis is a disease caused by mutations in the CFTR gene leading to misfolding amongst other defects of the CFTR protein (Fraser-Pitt and O'Neil 2015). Cystic Fibrosis, CF, affects many organ systems with a major symptom of the disease being degeneration of lung function caused by chronic respiratory infection and invasion of the airways by microbial pathogens (Fraser-Pitt and O'Neil 2015). CF is primarily treated by antibiotics which combat the symptoms of the disease (Fraser-Pitt and O'Neil 2015). Directly treating the main cause of CF is complicated by it being an autosomal recessive disease (Fraser-Pitt and O'Neil 2015). The CFTR gene codes for the CFTR protein which is a membrane protein belonging to the ABC transporter family that functions as a chloride/anion channel in epithelial cells (Yankaskas et al 2004). The CFTR protein is mainly involved in the production of secretions like sweat, digestive fluids, and mucus which become significantly thicker when the CFTR protein is damaged leading to the aforementioned organ damage and lung infections (Yankaskas et al 2004).

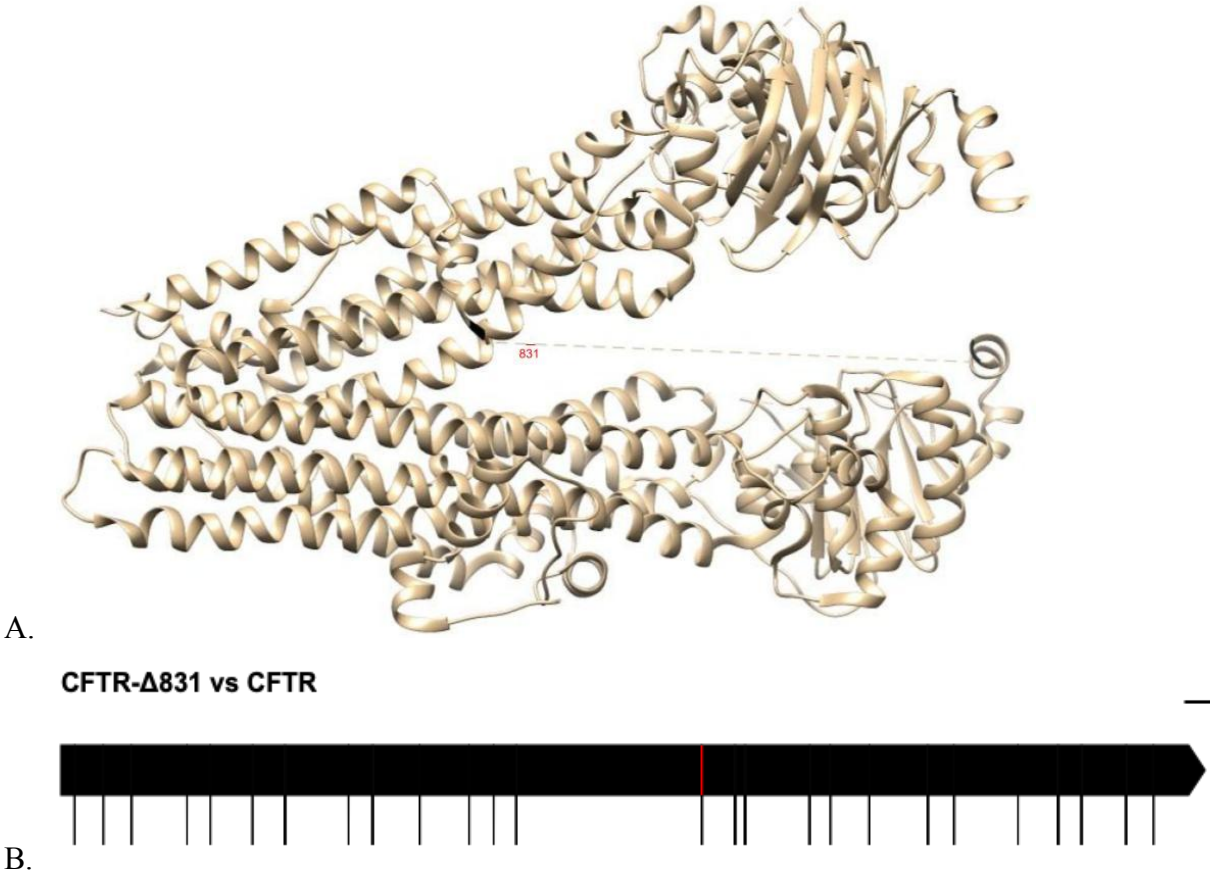
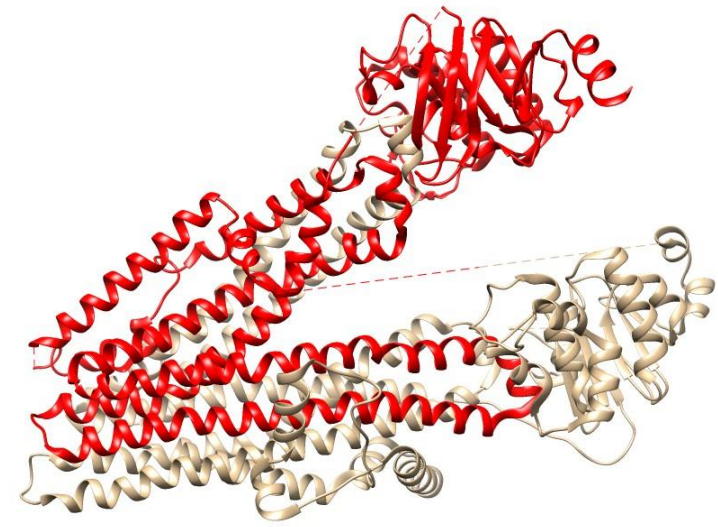


Figure 11. 3D Structure of the wild-type CFTR protein with the red section indicating the section missing in the structurally stable CFTR protein isoform CFTR-Δ831. Panel A shows the missing section in the 3D model in WT CFTR in a red highlight with the residue number written in red next to it. Panel B shows the missing section highlighted in red in the exon map of WT CFTR only at the very beginning of the fourteenth exon in WT CFTR. The black dashes at the bottom of Panel B indicate where the introns are.

There were four total CFTR isoforms used in addition to wild-type CFTR. Wild-type CFTR was the stable reference protein for all four pairs. The first two CFTR isoforms used were true positives, but only the first isoform was labeled as such. The labeled structurally stable CFTR isoform was CFTR-Δ831 (Sharma et al 2018). This isoform differs from wild-type CFTR by missing the 831st residue that wild-type CFTR has as shown in Figure 11 (Sharma et al 2018).

A.



B.

Figure 12. 3D Structure of the wild-type CFTR protein with the red section indicating the section missing in the structurally stable CFTR protein isoform CFTR-E831X. Panel A shows the missing section in the 3D model in WT CFTR in a red highlight. Panel B shows the missing section highlighted in red in the exon map of WT CFTR starting at the beginning of the fourteenth exon with the remaining exons also highlighted in WT CFTR. The black dashes at the bottom of Panel B indicate where the introns are.

The second structurally stable CFTR isoform that was not labeled was CFTR-E831X (Sharma et al 2018). This isoform differs from the wild-type due to the mRNA transcript for this isoform being alternatively spliced right after the trinucleotide coding for the 831st residue in wild-type CFTR, resulting in this variant only having the first 831 residues of wild-type CFTR as shown in Figure 12 (Sharma et al 2018). These two functional isoforms are derived from the CFTR gene having a nonsense mutation making it more susceptible to being alternatively spliced right at the beginning of the fourteenth exon of CFTR (Sharma et al 2018). These isoforms are considered structurally stable as they were expressed in HEK293, MDCK and CFBE41o- cell lines and were observed to not be immediately degraded, and they were even functional since they were able to combat CF with help from various modulator therapies like lumacaftor and tezacaftor (Sharma et al 2018).

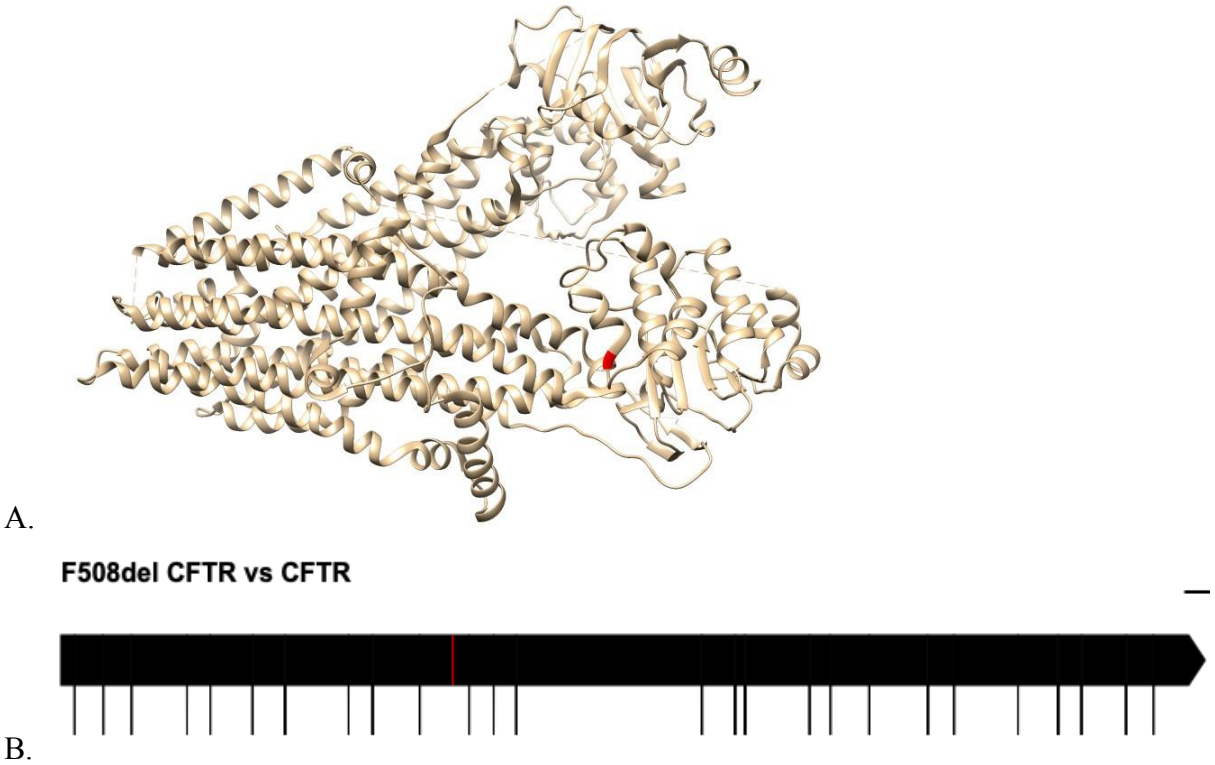
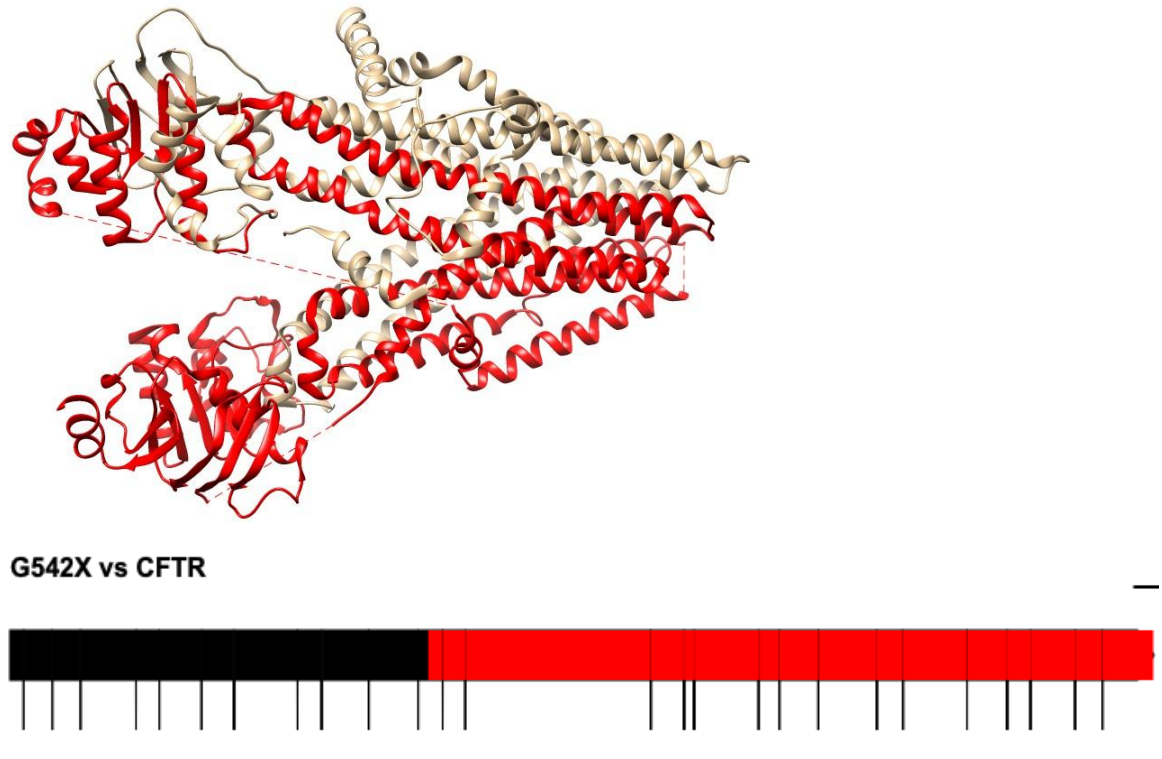


Figure 13. 3D Structure of the wild-type CFTR protein with the red section indicating the section missing in the structurally unstable CFTR mutant protein F508del CFTR. Panel A shows the missing section in the 3D model in WT CFTR in a red highlight. Panel B shows the missing section highlighted in red in the exon map of WT CFTR in a small part in the middle of the eleventh exon in WT CFTR. The black dashes at the bottom of Panel B indicate where the introns are.

There are over 1,500 mutations that have been shown to cause CF (Meng et al 2018). The most common CF-causing mutation is the deletion of a phenylalanine residue at position 508 of the wild-type CFTR protein (Meng et al 2018). This mutation is estimated to be the cause of 70% of all CF cases (Meng et al 2018). This mutant variant of CFTR is known as F508del CFTR, and the absence of the phenylalanine residue at position 508 leads to the structural instability of the protein which causes it to be quickly degraded by the endoplasmic reticulum quality control mechanism (Meng et al 2018). This phenylalanine residue at position 508 is highlighted in red in panel A of Figure 13 to display its placement in the structure of the wild-type CFTR protein. Panel B of Figure 13 showcases the single nucleotide codon missing in F508del CFTR that is present in the eleventh exon of WT CFTR. The endoplasmic reticulum quality control is a protein degradation mechanism specific to membrane proteins like the CFTR protein with a purpose similar, but not limited to other protein degradation mechanisms like the previously mentioned UPS and lysosomal protein degradation (Araki and Nagata 2011). Interestingly

enough, pharmacologically ‘rescued’ F508del CFTR quickly loses its structural stability leading to loss of its functional capabilities causing it to be subsequently targeted for lysosomal degradation (Meng et al 2018).

A.



B.

Figure 14. 3D Structure of the wild-type CFTR protein with the red section indicating what is absent in the structurally unstable CFTR mutant protein G542X. Panel A shows the missing section in the 3D model in WT CFTR in a red highlight. Panel B shows the missing section highlighted in red in the exon map of WT CFTR starting in the twelfth exon and missing the remaining exons in WT CFTR. The black dashes at the bottom of Panel B indicate where the introns are.

Another CF-causing mutant variant is called G542X (Fraser-Pitt and O'Neil 2015). It is caused by a point-substitution mutation creating a premature stop codon which is why G542X only has the first 541 amino acid residues of the wild-type CFTR protein (Fraser-Pitt and O'Neil 2015). In panel A of Figure 14, this sequence of missing residues in G542X is highlighted in red to visually display how significantly it differs from the wild-type CFTR protein regarding structure. Panel B of Figure 14 shows the difference regarding the exons of CFTR where G542X is shown to be different by missing the nucleotide sequences starting in the twelfth and remaining exons in WT CFTR. This premature stop codon accordingly results in premature termination of protein translation and the lack of the remaining residues causes G542X to be

nonfunctional and misfold (Fraser-Pitt and O'Neil 2015). This mutation type is estimated to be the cause of 10% of all CF cases (Fraser-Pitt and O'Neil 2015).

CF is caused by a minimal alteration in the CFTR protein in 70% of its cases which makes it difficult to detect before the introduction of its most notorious symptoms. Therefore, a real world application of the PU learning model created for this project can be demonstrated should it be able to accurately predict if F508del CFTR is structurally unstable. However, the absence of a single residue in F508del CFTR may make it difficult to accurately predict due to how feature importance has been previously assigned and just due to the difference from the wild-type being a single residue. Therefore, the model should also try predicting the stability of an unstable mutant variant of CFTR that is much more distinct from the wild-type CFTR protein like G542X. A stable labeled reference for CFTR should also be predicted along with the negative CFTR mutants because accurately predicting both positive and negative samples would be most indicative of the model's ability. Also, it must be acknowledged the PU learning model has a built-in bias for negative classification, especially with this model implementation following the traditional PU approach (Ramola et al 2019). Thus, another positive sample should be fed into the model to specifically test the effect of this bias. In this case study, four protein pairs were fed into the PU learning model after its testing/training with all four feature selections.

The wild-type CFTR protein can be found as chain A of the protein named 5UAK in PDB. The wild-type protein was used as the stable reference protein for all four samples.

Table 8. Evaluation Metrics for CFTR protein pairs

	Full Feature Selection	Feature Selection 1	Feature Selection 2	Feature Selection 3
F1 Score	66.67%	66.67%	66.67%	66.67%
ROC	75.00%	75.00%	75.00%	75.00%
Recall	50.00%	50.00%	50.00%	50.00%
Precision	100.00%	100.00%	100.00%	100.00%

Table 8 contains actual evaluation metrics instead of estimations because the true classification of the CFTR samples fed into the PU learning model are known. There were four samples fed into the model where the first two were structurally stable and the last two samples were not. Regardless of the feature selection, the model consistently predicted the labels for the negative samples correctly, but it consistently predicted the second positive CFTR sample to not be positive as can be shown by the recall values of 50.00%. The precision found in this case study was perfect at 100.00% (Shung 2018). The ROC in Table 8 was 75.00% indicating the model has a decent capability of distinguishing between structurally stable and unstable proteins specific to the CFTR gene. The F1 score for this case study as shown in Table 8 is 66.67%. It is interesting how different feature selections and different feature importances per feature selection provided no variation in the predicted labels given by the PU learning model.

Case Study 2: TP53

The protein p53 is considered to be one of the most important tumor suppressors if not the most important (García-Alai et al 2008). Tumor formation often results from the failure to properly regulate cell differentiation, death rate, and proliferation (García-Alai et al 2008). The protein p53 is of such importance as a regulator to these processes that 50% of human cancers contain mutations and alterations in its gene TP53 (García-Alai et al 2008). Typically, p53 functions as a transcription factor that can activate several antiproliferative programs by activating or repressing key effector genes (Zilfou and Lowe 2009). Also, p53 can release a variety of stress-inducing signals to different antiproliferative cellular responses like apoptosis, cell-cycle arrest, senescence or modulation of autophagy (Zilfou and Lowe 2009). This is only possible after the activation of p53, which can be achieved in response to DNA damage, oncogene activation, or hypoxia (Zilfou and Lowe 2009).

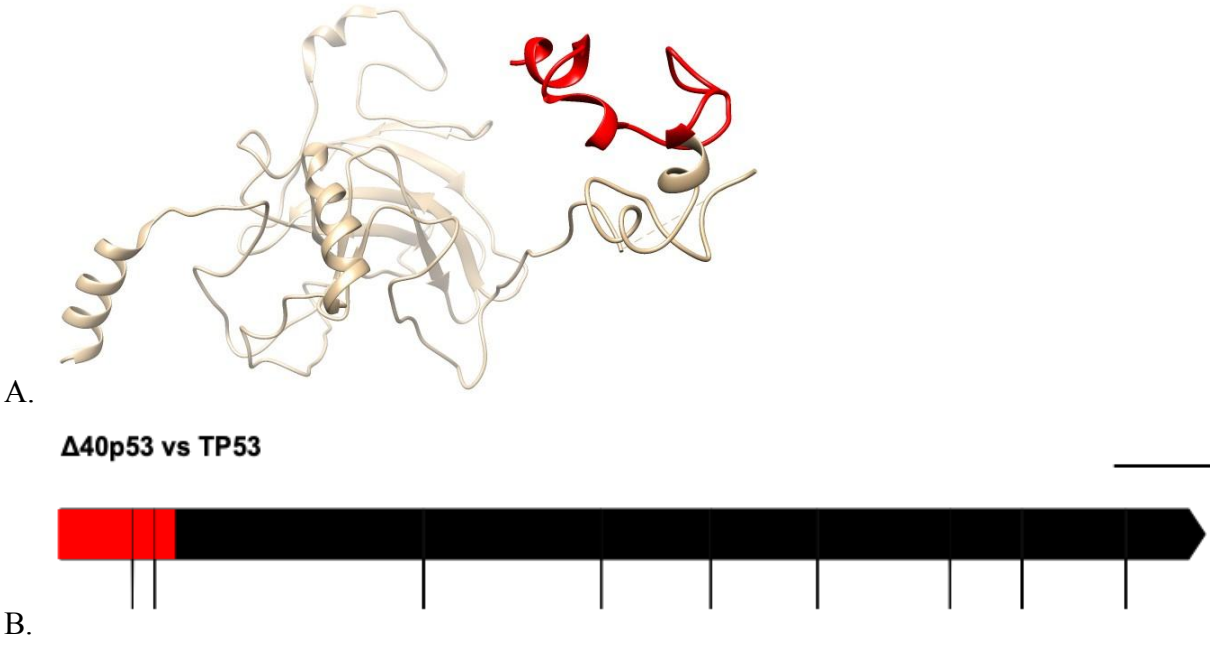


Figure 15. 3D Structure of p53 protein with the red section indicating the section missing in structurally stable p53 protein isoform Δ40p53. Panel A shows the missing section in the 3D model in WT p53 in a red highlight. Panel B shows the missing section highlighted in red in the exon map of WT p53 starting at the first exon and ending in the third exon of WT p53. The black dashes at the bottom of Panel B indicate where the introns are.

There were three total TP53 isoforms used in addition to wild-type TP53 also known as p53. Wild-type TP53 was the stable reference protein for all three pairs. The wild-type TP53

protein, p53, can be found as chain M of the protein named 6XRE in PDB. The first two TP53 isoforms used were true positives, but only the first isoform was labeled as such. The labeled structurally stable TP53 isoform was Δ40p53 (Hafsi et al 2013). This isoform differs from wild-type TP53 since it is missing the first 39 residues in wild-type TP53 as shown in Figure 15 (Hafsi et al 2013). This isoform is considered structurally stable because it was expressed from co-transfected vectors in p53-null cell lines Saos-2 and H1299 and in zebrafish and mice animal models where it was shown to not be immediately degraded, and also functional since it was able to interfere with the binding of Hdm2 to hetero-tetramers containing both Δ40p53 and normal p53 (Hafsi et al 2013).

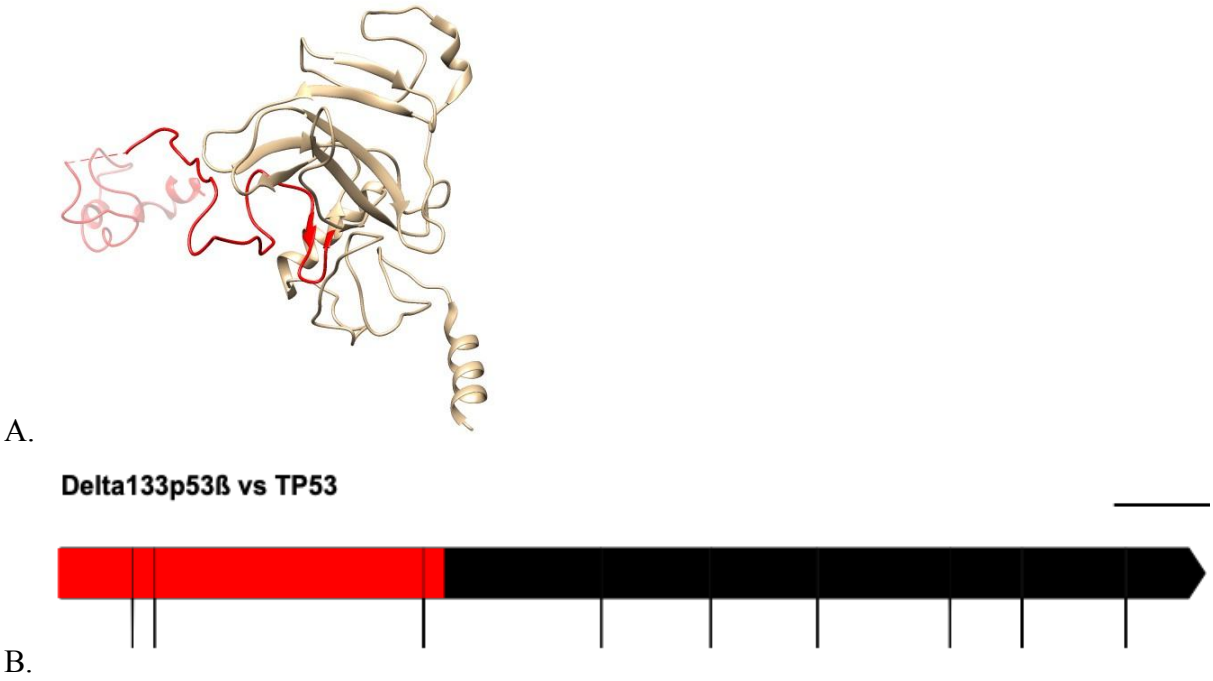
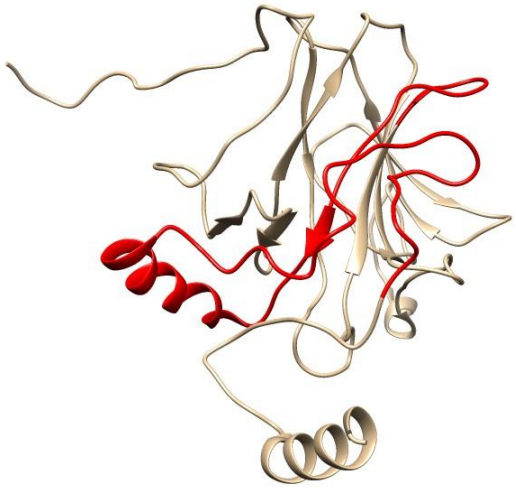


Figure 16. 3D Structure of p53 protein with the red section indicating the section missing in structurally stable p53 protein isoform delta133p53ß. Panel A shows the missing section in the 3D model in WT p53 in a red highlight. Panel B shows the missing section highlighted in red in the exon map of WT p53 starting at the first exon and ending in the fourth exon of WT p53. The black dashes at the bottom of Panel B indicate where the introns are.

The second structurally stable TP53 isoform that was not labeled was delta133p53ß (Arsic et al 2017). This isoform differs from wild-type TP53 since it lacks the first 132 residues in wild-type p53 which is shown in Figure 16 (Arsic et al 2017). This isoform is considered structurally stable since it is not immediately degraded and was proven functional since it was

observed strongly interacting with the GTPase RhoB *in vitro* and in the zebrafish animal model (Arsic et al 2017).



A.

B.

Figure 17. 3D Structure of p53 protein with the red section indicating the section missing in structurally unstable p53 isoform delta p53. Panel A shows the missing section in the 3D model in WT p53 in a red highlight. Panel B shows the missing section highlighted in red in the exon map of WT p53 starting at the end of the sixth exon, and ending towards the beginning of the eighth exon. The black dashes at the bottom of Panel B indicate where the introns are.

It has been repeatedly stated that p53 is a critical tumor suppressor and because of that, it is often nonfunctional in many human cancers (Zilfou and Lowe 2009). If TP53 was to undergo a mutation or be expressed in a way that results in p53 being structurally unstable, that would signify p53 is nonfunctional which presents a recognizable vulnerability to tumor formation (Zilfou and Lowe 2009). Thus, another real world application of this PU learning model can be demonstrated should it be able to properly predict the stability of a structurally unstable p53 variant seen in human cancers. One such common unstable variant is the p53 isoform known as delta p53 (García-Alai et al 2008). This isoform is created by an unusual alternative splicing of the seventh and eighth exons of TP53 (Rohaly et al 2005). This Alternative Splicing results in a loss of 66 residues, residues 257–322, making the isoform structurally unstable and

nonfunctional (García-Alai et al 2008). Figure 17 has the 3-dimensional structure of p53 where the section highlighted in red represents the 66 residues missing in delta p53 in panel A. Panel B of Figure 17 showcases where this mutation falls according to the exons of p53, it is seen starting in the end of the sixth exon and ending towards the beginning of the eighth exon. In vitro, delta p53 is unable to properly bind to DNA which is a very common feature amongst nonfunctional stable p53 variants that exist in human cancers (García-Alai et al 2008).

Three samples were made for this case study. The first two samples were positive proteins expressed from TP53, and the last sample was a known negative with the target protein being delta p53. This assortment of samples was done with a positive sample majority to test the negative bias of the model. Exactly, like the first study, these labels of these samples were predicted by the PU learning model after training/testing with the four feature selections.

Table 9. Evaluation Metrics for TP53 protein pairs

	Full Feature Selection	Feature Selection 1	Feature Selection 2	Feature Selection 3
F1 Score	80.00%	66.67%	100.00%	80.00%
ROC	50.00%	75.00%	100.00%	50.00%
Recall	100.00%	50.00%	100.00%	100.00%
Precision	66.67%	100.00%	100.00%	66.67%

Table 9, just like Table 8, presents the F1 score, ROC, recall, and precision values for the predicted labels provided by the PU learning model for this case study. Unlike Table 8, Table 9 displays a variety of evaluation metric values corresponding to the selection of features. The second selection of features provided by the Extra Trees Classifier was the only selection where the PU learning provided perfect results for this case study as displayed by unanimous values of 100.00% for the F1 score, ROC, recall, and precision. The third feature selection provided by the Random Forest classifier and the full feature selection had the same exact prediction with all three samples being predicted as positive/structurally stable. The full feature selection and third feature selection had a precision of 66.67% due to a single false positive, a recall of 100.00% due

to both true positives being predicted as such, 50.00% for the ROC since the model failed to distinguish one of the two classes, and 80% for the F1 score as a slight balance between recall and precision. The first selection of features provided by the chi squared based feature selector had correctly predicted the unstable sample as such, but incorrectly predicted the second stable sample as unstable. This resulted in a precision of 100.00% since no false positives were made, a recall of 50.00% because only one of the two true positives was predicted as such, 75.00% for the ROC since both classes were correctly distinguished in all but one of the three samples, and 66.67% for the F1 score.

Discussion

The key findings of this project was that despite a lack of negatively labeled data and the PU learning model constantly overvaluing features less indicative of structural stability, the model performed quite well. This is supported by the case study results as seen in Table 9 and Table 8. Table 9 shows that the second feature selection had the perfect predictions. Despite the range of predictions given the feature selections in Table 9, the predictions per feature selection showcased strengths where other feature selections were weak. As seen in Table 9, the first feature selection had an F1 score of 66.67% whereas the third feature selection and full feature selection had scores of 75.00%. Yet, the first feature selection has an ROC value of 75.00% meaning it could to an extent distinguish both the negative and positive data compared to the third feature selection and full feature selection with ROC values of 50.00% due to it only being able to identify positive data. This type of balance was not as evident in Table 8 which was also missing perfect predictions, but it was likely influenced by negative bias which follows suit with what was already known.

It was known and acknowledged that there would be an inherent negative bias for the PU learning model predictions due to how this implementation of the traditional PU learning approach treats unlabeled data as if it were negative. Yet this is balanced by the setup of PU learning itself. A prime example of this is how the testing data consisted only of the positive/labeled data allowing the model to specifically focus just on what makes a positive sample positive and adjust accordingly. And if the labeled positives are representative of the true positive distribution fulfilling the SCAR assumption, the model should follow the logic and be able to properly identify a reliable recall and the true positives (Bekker and Davis 2020). As seen in Table 4, Table 5, Table 6, and Table 7, the final recall values were consistently perfect and the average k-fold recall values were never below 0.95. Using normal evaluation metrics would inaccurately portray the obtained results and were addressed by the bias-corrected evaluation metric estimations. Those values are not exact, but empirical evidence proves they are within a concise range of the true metric values (Ramola et al 2019). These metric estimations do not solve the negative bias, they simply attempt to compensate for the inaccuracy normal metrics would provide with unlabeled data (Ramola et al 2019). So because these estimates are exactly

that; estimates, and not exact, the perfect metric estimations seen in Table 4, Table 5, Table 6, and Table 7 can be considered valid.

There were methods and metrics implemented to balance the inherent negative bias of PU learning, yet there were still some areas where this bias was unaffected. This is something that can be said looking at Figure 3, Figure 7, and Figure 9 since the contrasting the t-SNE plots in their respective A and B panels seem to show almost no difference. However, Figure 3 did show a difference in plotting corresponding to the first feature selection and it was shown in Table 9 to provide the PU learning model the best ROC values second only to the values of the second feature selection. Furthermore, the second feature selection corresponds to the t-SNE plots in Figure 7 that seemingly indicate no visual difference in the data clustering.

A consistent theme of the results from this project is balance. This type of balance can be seen in Figure 4, Figure 6, Figure 8, and Figure 10. It was discouraging to see that features less relevant to structural stability like tripeptide composition and triad were being valued higher than features that have great relevance to structural stability like instability index and molecular extinction coefficient. But tripeptide composition and triad features and their corresponding delta features are not consistently guaranteed to provide values that can be computationally useful due to lack of the corresponding peptide. Whereas features like instability index and molecular extinction coefficient are consistently guaranteed to provide values that can be computationally useful. In panel A of Figure 4, Figure 6, Figure 8, and Figure 10, delta instability index was always among the top ten valued features proving that the model was able to still identify quality features amidst an excess of less relevant ones.

Conclusion and Future Work

Overall, this project showcased that despite some explicit disadvantages, the PU learning model created can be of applicable use in the real world and potentially provide accurate predictions on the stability of protein isoforms/variants. Despite the abnormally high values assigned to less relevant features, the model was usually able to compensate by realizing the importance of more relevant features. That being said, consistent performances with higher exact performance evaluation metrics is still desired.

For future work, it is recommended to test/train this PU model using training/testing sets composed of data samples with known labels where the negative samples and a random subset of the positive samples are unlabeled before the actual testing/training (Bekker and Davis 2020). This will allow the predictions of the model on actual unlabeled samples to be considered more reliable (Bekker and Davis 2020). Next, the selection of features should be managed more directly such that the model is not biased towards identifying patterns from feature categories with bulk features. And finally, to properly verify the existence of a protein isoform/variant, it is unavoidable but it has to be done physically. Therefore, it would be advised that physical protein verification continues to happen in bulk such that training/testing machine learning algorithms can be provided with more data and thus be decisively more accurate when they try to predict protein verification.

References

Agmon A. (2020). Semi-Supervised Classification of Unlabeled Data (PU Learning). Medium. towardsdatascience.com. https://towardsdatascience.com/semi-supervised-classification-of-unlabeled-data-pu-learning-81f96e96f7cb

Alternative Splicing of Genes: Definition, Mechanism & Regulation. (2015). Retrieved from https://study.com/academy/lesson/alternative-splicing-of-genes-definition-mechanism-regulation.html.

Anjana R, Vaishnavi M, Sherlin D, Kumar S, Naveen K, Kanth P, Sekar K. (2012). Aromatic-aromatic interactions in structures of proteins and protein-DNA complexes: a study based on orientation and distance. Bioinformation, 8(24), 1220–1224. https://doi.org/10.6026/97320630081220

Araki K, & Nagata K. (2011). Protein folding and quality control in the ER. Cold Spring Harbor perspectives in biology, 3(11), a007526. https://doi.org/10.1101/cshperspect.a007526

Arsic N, Ho-Pun-Cheung A, Evelyne C, Assenat E, Jarlier M, Anguille C, Colard M, Pezet M, Roux P, Gadea G. (2017). The p53 isoform delta133p53ß regulates cancer cell apoptosis in a RhoB-dependent manner. PLOS ONE, 12(2), 1–15. https://doi.org/10.1371/journal.pone.0172125

Bekker J, Davis J. (2020). Learning from positive and unlabeled data: a survey. Machine Learning, 109(4), 719–760. https://doi.org/10.1007/s10994-020-05877-5

Berman H, Westbrook J, Feng Z, Gilliland G, Bhat T, Weissig H, Shindyalov I, Bourne P. (2000) The Protein Data Bank. Nucleic Acids Research, 28, 235-242. https://www.rcsb.org/

Breiman L. (2001). Random Forest. Machine Learning, 45(1), 5–32. https://doi.org/10.1023/a:1010933404324

Buitinck L, Louppe G, Blondel M, Pedregosa F, Mueller A, Grisel O, Niculae V, Prettenhofer P, Gramfort A, Grobler J, Layton R, VanderPlas J, Joly A, Holt B, Varoquaux G (2013). API design for machine learning software: experiences from the scikit-learn project. In ECML PKDD Workshop: Languages for Data Mining and Machine Learning, 108–122. Available from: https://scikit-learn.org/stable/modules/classes.html

Chen C, Lin M, Liao C, Chang H, Chu, Y. (2020). iStable 2.0: Predicting protein thermal stability changes by integrating various characteristic modules. Computational and Structural Biotechnology Journal, 18, 622–630. https://doi.org/10.1016/j.csbj.2020.02.021

Chou K. (2000). Prediction of Protein Subcellular Locations by Incorporating Quasi-Sequence-Order Effect. Biochemical and Biophysical Research Communications, 278(2), 477–483. https://doi.org/10.1006/bbrc.2000.3815

Chou K. (2001). Prediction of protein cellular attributes using pseudo-amino acid composition. Proteins: Structure, Function, and Genetics, 43(3), 246–255. https://doi.org/10.1002/prot.1035

Chou K. (2004). Using amphiphilic pseudo amino acid composition to predict enzyme subfamily classes. Bioinformatics, 21(1), 10–19. https://doi.org/10.1093/bioinformatics/bth466

Cock PA, Antao T, Chang JT, Chapman BA, Cox CJ, Dalke A, Friedberg I, Hamelryck T, Kauff F, Wilczynski B, de Hoon MJL. (2009) Biopython: freely available Python tools for computational molecular biology and bioinformatics. Bioinformatics, 25, 1422-1423

Cooper GM. (2000) The Cell: A Molecular Approach (2nd edition). Sinauer Associates. Available from: https://www.ncbi.nlm.nih.gov/books/NBK9957/

Dong J, Yao Z, Zhang L, Luo F, Lin Q, Lu A, Chen A, Cao D. (2018). PyBioMed: a python library for various molecular representations of chemicals, proteins and DNAs and their interactions. Journal of cheminformatics, 10(1), 16. https://doi.org/10.1186/s13321-018-0270-2

Dubey A. (2018). Feature Selection Using Random forest. Medium. towardsdatascience.com. https://towardsdatascience.com/feature-selection-using-random-forest-26d7b747597f

Elkan C, Noto K. (2008) Learning classifiers from only positive and unlabeled data. Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining. ACM. 213–220. Available from: https://cseweb.ucsd.edu/~elkan/posonly.pdf

Fraser-Pitt D, O'Neil D. (2015). Cystic fibrosis - a multiorgan protein misfolding disease. Future science OA, 1(2), FSO57. https://doi.org/10.4155/fso.15.57

García-Alai M, Tidow H, Natan E, Townsley F, Veprintsev D, Fersht A. (2008). The novel p53

isoform "delta p53" is a misfolded protein and does not bind the p21 promoter site. Protein science : a publication of the Protein Society, 17(10), 1671–1678. https://doi.org/10.1110/ps.036996.108

Gajawada S. (2020). Chi-Square Test for Feature Selection in Machine learning. Medium. towardsdatascience.com. https://towardsdatascience.com/chi-square-test-for-feature-selection-in-machine-learning-206b1f0b8223

Geurts P, Ernst D, Wehenkel, L. (2006). Extremely randomized trees. Machine Learning, 63(1), 3–42. https://doi.org/10.1007/s10994-006-6226-1

Gorania M, Seker H, Haris P. (2010). Predicting a protein's melting temperature from its amino acid sequence. Annual International Conference of the IEEE Engineering in Medicine and Biology Society. IEEE Engineering in Medicine and Biology Society. Annual International Conference, 2010, 1820–1823. https://doi.org/10.1109/IEMBS.2010.5626421

Guruprasad K, Reddy B, Pandit M. (1990). Correlation between stability of a protein and its dipeptide composition: a novel approach for predicting in vivo stability of a protein from its primary sequence. “Protein Engineering, Design and Selection,” 4(2), 155–161. https://doi.org/10.1093/protein/4.2.155

Hafsi H, Santos-Silva D, Courtois-Cox S, Hainaut P. (2013). Effects of Δ40p53, an isoform of p53 lacking the N-terminus, on transactivation capacity of the tumor suppressor protein p53. BMC Cancer, 13(1). https://doi.org/10.1186/1471-2407-13-134

Hartwell L, Goldberg M, Fischer J, Hood L. (2017). Genetics: From Genes to Genomes (6th ed.). McGraw-Hill Education.

Kyte J, & Doolittle R. (1982). A simple method for displaying the hydropathic character of a protein. Journal of Molecular Biology, 157(1), 105–132. https://doi.org/10.1016/0022-2836(82)90515-0

Laboratorio Interdisciplinare di Tecnologie Avanzate (LITA). (2012). ASPicDB. Alternative Splicing Prediction Database (ASPicDB). http://srv00.recas.ba.infn.it/ASPicDB/index.php

Martelli P, D'Antonio M, Bonizzoni P, Castrignanò T, D'Erchia A, D'Onorio De Meo P, Fariselli

P, Finelli M, Licciulli F, Mangiulli M, Mignone F, Pavesi G, Picardi E, Rizzi R, Rossi I, Valletti A, Zauli A, Zambelli F, Casadio R, Pesole G. (2011). ASPicDB: a database of annotated transcript and protein variants generated by alternative splicing. Nucleic acids research, 39(Database issue), D80–D85. https://doi.org/10.1093/nar/gkq1073

Meng X, Clews J, Kargas V, Wang X, Ford R. (2017). The cystic fibrosis transmembrane conductance regulator (CFTR) and its stability. Cellular and molecular life sciences : CMLS, 74(1), 23–38. https://doi.org/10.1007/s00018-016-2386-8

Morde V. (2019). XGBoost Algorithm: Long May She Reign!. Medium. towardsdatascience.com. https://towardsdatascience.com/https-medium-com-vishalmorde-xgboost-algorithm-long-she-may-rein-edd9f99be63d

Narkhede S. (2018). Understanding AUC - ROC Curve. towardsdatascience.com. Medium. https://towardsdatascience.com/understanding-auc-roc-curve-68b2303cc9c5

Narykov O, Johnson N, Korkin D. (2020). Predicting Protein Interaction Network Perturbation by Alternative Splicing with Semi-Supervised Learning. SSRN Electronic Journal. Published. https://doi.org/10.2139/ssrn.3650349

Pace C, Vajdos F, Fee L, Grimsley G, Gray T. (1995). How to measure and predict the molar absorption coefficient of a protein. Protein science : a publication of the Protein Society, 4(11), 2411–2423. https://doi.org/10.1002/pro.5560041120

Pramoditha R. (2021). k-fold cross-validation explained in plain English. Medium. towardsdatascience.com. https://towardsdatascience.com/k-fold-cross-validation-explained-in-plain-english-659e33c0bc0

Pucci F, Kwasigroch J, Rooman M. (2017). SCooP: an accurate and fast predictor of protein stability curves as a function of temperature. Bioinformatics (Oxford, England), 33(21), 3415–3422. https://doi.org/10.1093/bioinformatics/btx417

Ramola R, Jain S, Radivojac P. (2019). Estimating classification accuracy in positive-unlabeled learning: characterization and correction strategies. Pacific Symposium on Biocomputing. Pacific Symposium on Biocomputing, 24, 124–135.

Rohaly G, Chemnitz J, Dehde S, Nunez A, Heukeshoven J, Deppert W, Dornreiter I. (2005). A

Novel Human p53 Isoform Is an Essential Element of the ATR-Intra-S Phase Checkpoint. Cell, 122(1), 21–32. https://doi.org/10.1016/j.cell.2005.04.032

Sharma N, Evans T, Pellicore M, Davis E, Aksit M, McCague A, Joynt A, Lu Z, Han S, Anzmann A, Lam A, Thaxton A, West N, Merlo C, Gottschalk L, Raraigh K, Sosnay P, Cotton C, Cutting G. (2018). Capitalizing on the heterogeneous effects of CFTR nonsense and frameshift variants to inform therapeutic strategy for cystic fibrosis. PLOS Genetics, 14(11), e1007723. https://doi.org/10.1371/journal.pgen.1007723

Shaw K, Grimsley G, Yakovlev G, Makarov A, Pace C. (2001). The effect of net charge on the solubility, activity, and stability of ribonuclease Sa. Protein science : a publication of the Protein Society, 10(6), 1206–1215. https://doi.org/10.1110/ps.440101

Shen J, Zhang J, Luo X, Zhu W, Yu K, Chen K, Li Y, Jiang H. (2007). Predicting protein-protein interactions based only on sequences information. Proceedings of the National Academy of Sciences of the United States of America, 104(11), 4337–4341. https://doi.org/10.1073/pnas.0607879104

Shung K. (2018). Accuracy, Precision, Recall or F1? towardsdatascience.com. Medium. https://towardsdatascience.com/accuracy-precision-recall-or-f1-331fb37c5cb9

Silverstein A. (2003). Cellular versus humoral immunology: a century-long dispute. Nature Immunology, 4(5), 425–428. https://doi.org/10.1038/ni0503-425

Taverna D, Goldstein R. (2001). Why are proteins marginally stable? Proteins: Structure, Function, and Genetics, 46(1), 105–109. https://doi.org/10.1002/prot.10016

Tokuriki N, Stricher F, Serrano L, Tawfik D. (2008). How Protein Stability and New Functions Trade Off. PLoS Computational Biology, 4(2), e1000002. https://doi.org/10.1371/journal.pcbi.1000002

Tress M. Abascal F, Valencia A. (2017). Most Alternative Isoforms Are Not Functionally Important. Trends in biochemical sciences, 42(6), 408–410. https://doi.org/10.1016/j.tibs.2017.04.002

Wattenberg M, Viégas F, Johnson I. (2016). How to Use t-SNE Effectively. Distill, 1(10). https://doi.org/10.23915/distill.00002

van der Maaten L, Hinton G. (2008). Visualizing Data using t-SNE. Journal of Machine Learning Research, 9(86), 2579–2605. https://www.jmlr.org/papers/volume9/vandermaaten08a/vandermaaten08a.pdf

Vilchez D, Saez I, Dillin A. (2014). The role of protein clearance mechanisms in organismal ageing and age-related diseases. Nature Communications, 5(1). https://doi.org/10.1038/ncomms6659

Voet D, Voet J, Pratt C. (2016). Fundamentals of Biochemistry: Life at the Molecular Level (5th ed.). Wiley.

Yankaskas J, Marshall B, Sufian B, Simon R, Rodman D. (2004). Cystic Fibrosis Adult Care. CHEST, 125(1), 1S-39S. https://doi.org/10.1378/chest.125.1_suppl.1S

Yang X, Coulombe-Huntington J, Kang S, Sheynkman G, Hao T, Richardson A, Sun S, Yang F, Shen Y, Murray R, Spirohn K, Begg B, Duran-Frigola M, MacWilliams A, Pevzner S, Zhong Q, Trigg S, Tam S, Ghamsari L, Sahni N, … Vidal M. (2016). Widespread Expansion of Protein Interaction Capabilities by Alternative Splicing. Cell, 164(4), 805–817. https://doi.org/10.1016/j.cell.2016.01.029

Yang Y, Ding X, Zhu G, Niroula A, Lv Q, Vihinen M. (2019). ProTstab – predictor for cellular protein stability. BMC Genomics, 20(1). https://doi.org/10.1186/s12864-019-6138-7

Zilfou J, & Lowe S. (2009). Tumor suppressive functions of p53. Cold Spring Harbor perspectives in biology, 1(5), a001883. https://doi.org/10.1101/cshperspect.a001883