APPLYING DATA SCIENCE TECHNIQUES TO UNDERSTAND DNA MUTATIONS IN HEMOPHILIA- A

Institiúid Teicneolaíochta Cheatharlach

INSTITUTE of TECHNOLOGY CARLOW

At the Heart of South Leinster

STUDENT NAME:	**Pranav Deogaonkar**
COURSE NAME:	**M.Sc. Data Science**
DEPARTMENT:	**Department of Computing and Networking**
SUPERVISOR:	**Prof. Paul Barry**
SUBMISSION DATE:	**17-08-2020**

ACKNOWLEDGEMENTS

I would like to sincerely thank Prof. Paul Barry, for his help and guidance throughout this research work as a dissertation supervisor and research mentor. I would like to extend this gratitude to the staff and lecturers from the Department of Computing and Networking in IT Carlow for all the skills and knowledge I have gained during my time here. I would also like to thank Prof. Andrew Llyod for sharing his invaluable time and knowledge of biology and genetics. I wish to acknowledge the support and encouragement from my family, and friends, without which this research would not be possible.

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

1. INTRODUCTION

Hemophilia A is a hereditary bleeding disorder that affects 1 in 5000 males worldwide (Fast Facts, 2020). Currently, more than 400,000 people are suffering from hemophilia (Fast Facts, 2020). It is seen that nearly 75% of the hemophilia population across the globe is devoid of appropriate treatment or have no medical facility available for treatment (Fast Facts, 2020). Also, hemophilia is not curable hence patients have to take lifelong medical treatment which is expensive (Fast Facts, 2020).

Hemophilia A is a genetic bleeding disorder that is not curable; requires extensive research work to be carried out in bioinformatics regarding its treatment and prevention. Integrating Artificial intelligence (AI) with bioinformatics can lead to discovering those areas in hemophilia which are unexplored. We can study different genetic disorders by analyzing the data, finding patterns, and building models that can help in designing treatment for such incurable disorders.

Today, we have a lot of bioinformatic data available to analyze and extract information from so that it can be utilized by machine learning experts to develop models that can be implemented in healthcare. Applications of machine learning particularly in genomics are influencing the way genetic research is conducted and how the physicians can provide personal health care. Genomics research is a promising field of study for overcoming global healthcare issues that can be addressed now and also in the years to come. Recently, in a couple of years costs for genetic research and testing have plunged due to advancements made in genome sequencing technology which in turn have made it economically achievable (Yeager A.,2019). But it is crucial to impart medical treatments to patients across the globe at lower costs and with high standards. This awakens our thought process to lead to a discussion on whether we can find patterns and trends in mutations seen in hemophilia A from the data of the past 35 years and thereby predicting mutation severity through generating a machine learning model to aid in early diagnosis of hemophilia A disorder.

2. PROBLEM STATEMENT

Worldwide, Hemophilia A population can be divided into severe, moderate, and mild patients as per factor VIII levels in blood plasma (OMIM Entry - # 306700 - HEMOPHILIA A; HEMA, 2020). The proportion of cases found is 50% severe, 10% moderate, and 40% mild patients (OMIM Entry - # 306700 - HEMOPHILIA A; HEMA, 2020). In prior research on hemophilia A, gene editing can be considered as a boon to coping up with this bleeding disorder. Gene editing tools require the DNA to be cut at a precise location to repair the genetic defect (Bhattacharjee and Nandi, 2018). Machine learning and AI are leveraged to predict the type of repair in gene editing techniques by the Wellcome Sanger Institute (Yeager A.,2019). Using AI for discovering the most common mutation locations, type of mechanism responsible for mutation, and predicting the overall severity of mutations in gene F8 among hemophilia A patients can be the tasks to examine.

Initially, we had framed our research question to be the prediction of the type of repair in DNA required for hemophilia A patients. We had started our research by exploring bioinformatics and particularly in gene-editing techniques such as CRISPR. We were able to retrieve DNA data of hemophilia A patients from the National Centre for Biotechnology Information (NCBI) website. But we were unable to find historical data related to the mapping of DNA level mutations to cellular repair mechanism in hemophilia A. Hence, we modified our research path and selected data related to mutations from the F8 gene in DNA from the Centers for Disease Control and Prevention (CDC) database.

Our study focusses on finding interesting patterns and trends from novel mutations found in hemophilia A patients. It also aims at creating a classification model to predict the overall severity of mutations. We will use a framework to navigate through the research work. So, we can frame our research question/s to check the authenticity of our proposed research study.

Research questions for our study can be stated below as:

We have formulated a broad research question which is, can data science technologies help in the understanding of mutations in DNA specific for hemophilia A disorder? So, to seek an answer to our research question we subdivided our research question into sub-questions as listed below:

1. Can we find patterns in the data to generate insights that can aid medical professionals/researchers to investigate more into discovering a novel approach towards hemophilia A treatment?

2. Is it possible to develop a binary classification model with high accuracy to predict the overall severity of mutations in DNA from hemophilia A patients and utilize the model into the healthcare sector for genetic diagnosis of hereditary disorders?

3. LITERATURE REVIEW

3.1 Molecular Biological concepts: DNA and RNA

The human body comprises of numerous cells with each cell having subcellular components. One of those components is nucleic acids. A nucleic acid is a type of biomolecule that gives the organism(human) its identity (Bordoloi, Roy, Nirmala, 2018). These nucleic acids are significant in carrying out various cellular processes (Lindow, Baum, Leborgne and Hege, 2019). Nucleic acids are made up of monomers known as nucleotides. There are two types of nucleic acids in the human body, deoxyribonucleic acid (DNA) and ribonucleic acid (RNA). DNA prominently carries genetic information, whereas RNA is required to perform the task of transforming this genetic information into molecular enzymes (Lindow, Baum, Leborgne and Hege, 2019). In this synthesis process, three types of RNA take part, namely messenger RNA (mRNA), transfer RNA (tRNA), and ribosomal RNA (rRNA). DNA is stored in a cell by coiling around proteins called histone. To save space, the coils undergo supercoiling. A single long supercoiled DNA molecule with all the histones is known as a Chromosome. Also, all the genetic material in the cell is collectively known as Chromatin.

In 1953, Watson and Crick discovered the structure of DNA. A double helix structure of DNA is made up of two strands of nucleotides that are interlinked by base pairs (Lindow, Baum, Leborgne and Hege, 2019). Whereas, RNA consists of single strands that are entangled with one another with base pairs from the same strand (Lindow, Baum, Leborgne and Hege, 2019). A nucleotide comprises a foundation section and a nucleobase. The backbone part consists of the sugar deoxyribose and a phosphate group which interlinks the nucleotides with the strand (Lindow, Baum, Leborgne and Hege, 2019). In DNA, the backbone segment is same for all the nucleotides but has four types of nucleobases that are divided into two groups namely single-ring pyrimidines consisting of nucleobases cytosine (C) and thymine (T), and another group known as double-ring purines with nucleobases adenine (A) and guanine (G) (Lindow, Baum, Leborgne and Hege, 2019). Hydrogen bonds are created between every nucleobase of one strand and each nucleobase of another strand. Mostly, the nucleobase pairs are Watson-Crick base pairs, wherein Adenine appears to bind in two hydrogen bonds to thymine (A-T), and Guanine binds three hydrogen bonds to cytosine (G-C) (Lindow, Baum, Leborgne and Hege, 2019). Thus, G-C base pairs are stronger than A-T pairs as they have an extra hydrogen bond. So, the count of G-C pairs and their allotment along the helix has a direct effect on DNA stability (Lindow, Baum, Leborgne and Hege, 2019).

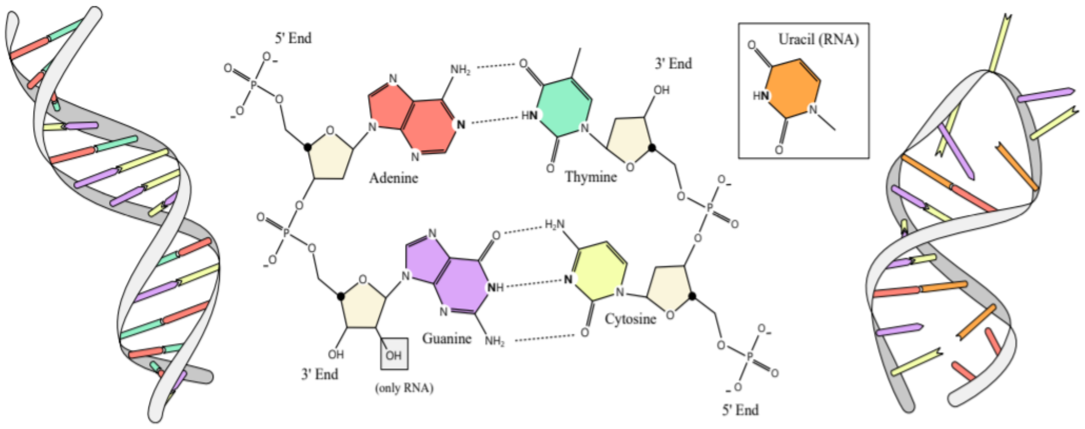


Figure 1: This shows the structure of DNA and RNA (Lindow, Baum, Leborgne and Hege, 2019)

Ribonucleic acids (RNA) is a single-stranded structure that folds back partly by generating base pairs between nucleotides of the same strand (Lindow, Baum, Leborgne and Hege, 2019). The foundation part of RNA has nearly the same structure as DNA except that it has an extra hydroxyl (OH) group attached to the beige ring. Also, the nucleobases in RNA are the same as DNA other than the thymine (T) from DNA is replaced by uracil (U) (Lindow, Baum, Leborgne and Hege, 2019). Thus, A-U and G-C are the common base pairs found in RNA.

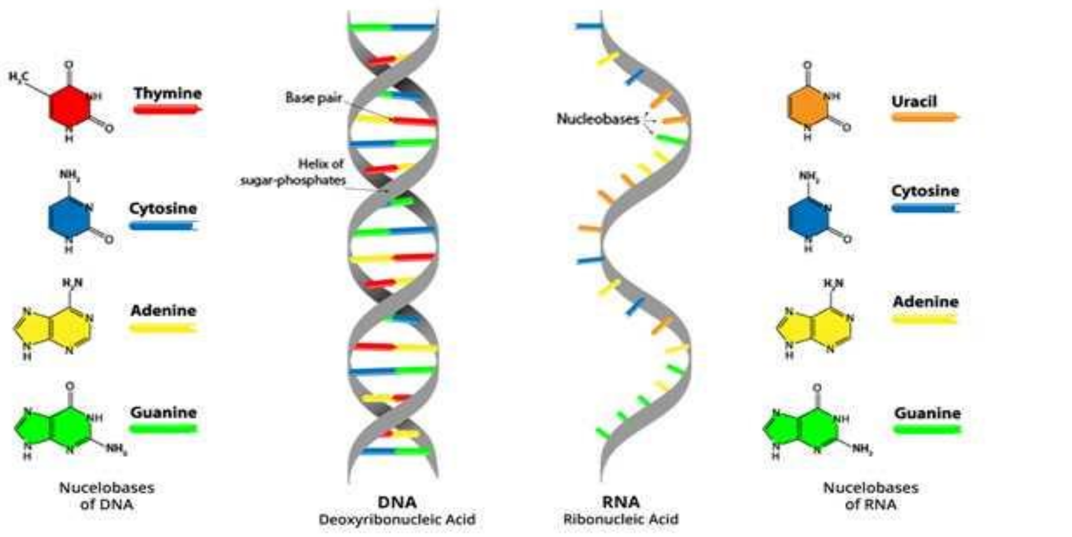


Figure 2:This figure depicts the nucleotide bases in DNA and RNA (Lindow, Baum, Leborgne and Hege, 2019)

The sequencing of DNA acts as a tool to identify the sequence of nucleotides that form DNA strands (Bordoloi, Roy, Nirmala, 2018). Genes that are involved in protein building follow a two-step process where firstly, in the transcription phase the DNA sequence is transcribed to RNA. Also, the RNA undergoes an extra processing step to convert it to messenger RNA (mRNA). Secondly, the translation phase comprises of translating nucleotide sequence in mRNA to sequence of amino acids in a protein chain (Central dogma (DNA to RNA to protein) | Biology library | Khan Academy, 2020).

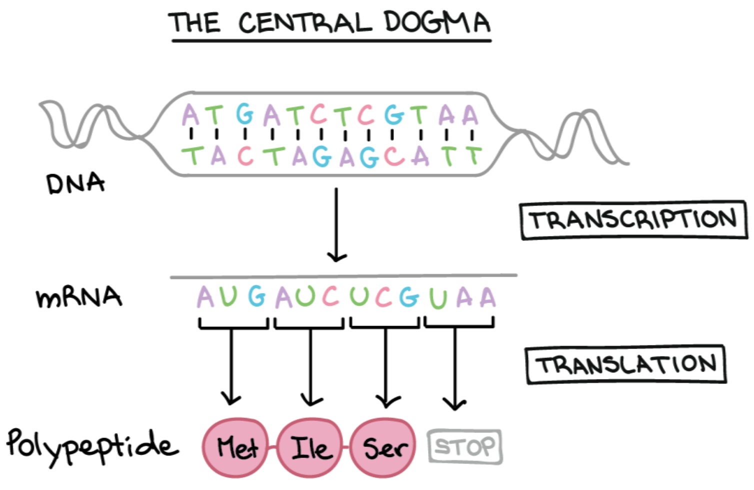


Figure 3:This figure depicts the process of translating from DNA to Amino acids (Central dogma (DNA to RNA to protein) | Biology library | Khan Academy, 2020)

This translation is carried out by reading the nucleotides in a group of three (Bordoloi, Roy, Nirmala, 2018). This triplet of nucleotide is known as a Codon (Bordoloi, Roy, Nirmala, 2018). Codons are read in mRNA sequence beginning with a start codon till a stop codon is encountered. During the translation phase, in the task of dividing the mRNA sequence into codons we have to initialize reading the sequence with the start codon but from which position in the sequence is the question to be addressed (Central dogma (DNA to RNA to protein) | Biology library | Khan Academy, 2020). This drives us to the concept of a reading frame. So, the correct position of the start codon decides the reading frame. Mutations that result in insertion/deletion of the nucleotide base often change the reading frame of the mRNA sequence causing to generate incorrect protein translated further into the polypeptide chain (Central dogma (DNA to RNA to protein) | Biology library | Khan Academy, 2020).

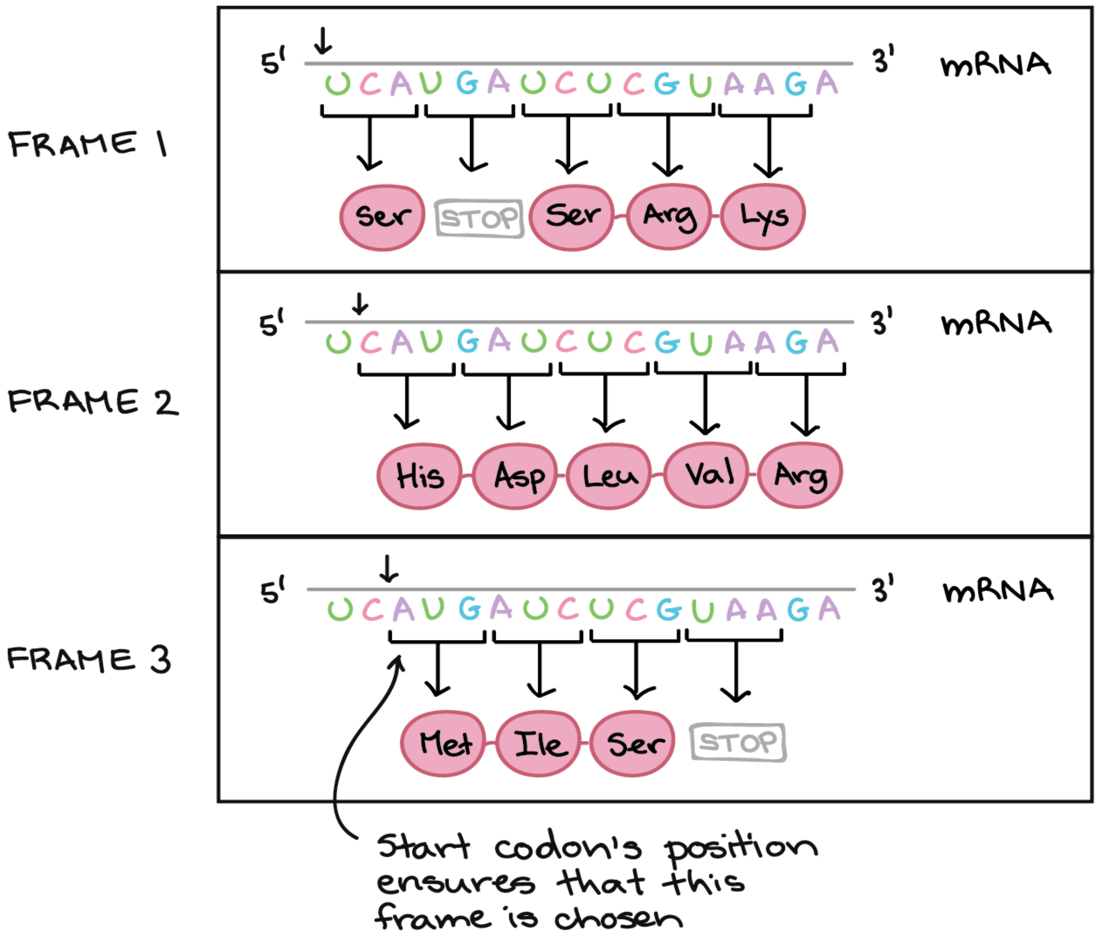
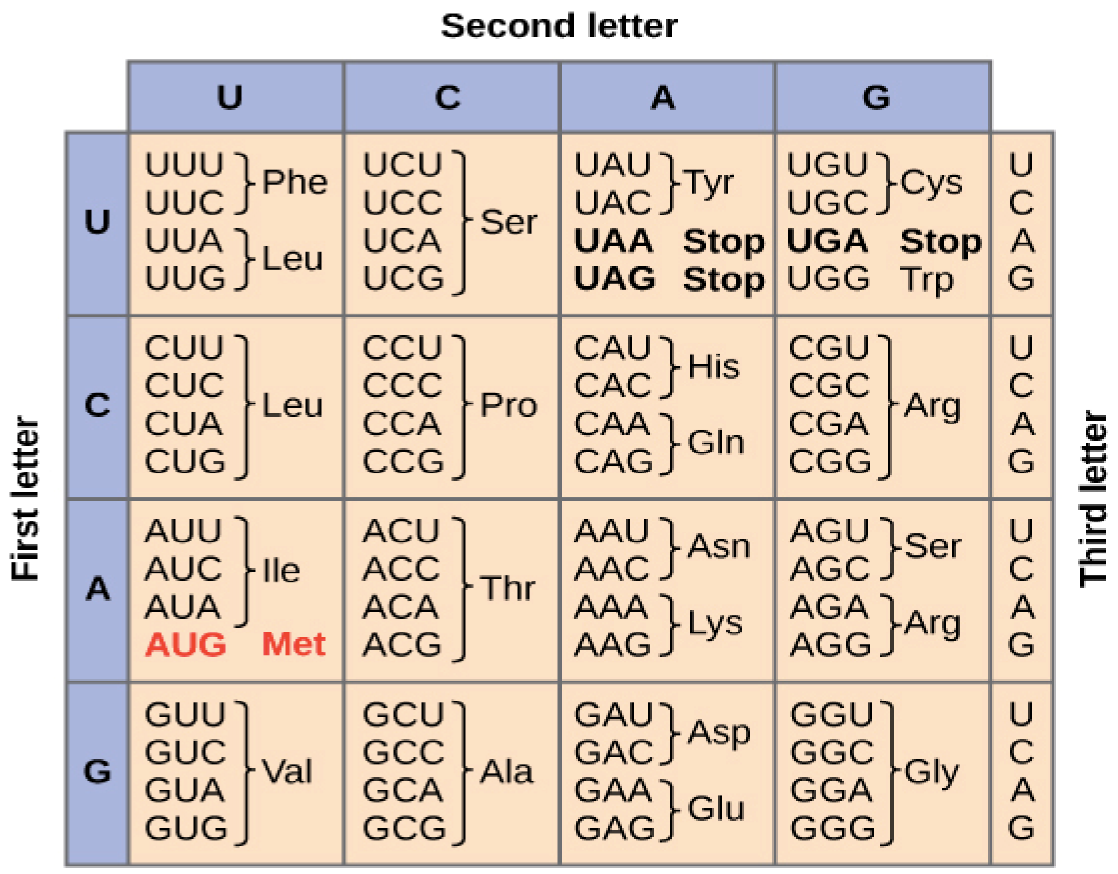


Figure 4:This image shows the concept of "reading frame" (Central dogma (DNA to RNA to protein) | Biology library | Khan Academy, 2020)

Each codon specifies an amino acid. These amino acids act as building blocks for protein (Bordoloi, Roy, Nirmala, 2018). Protein structure is derived from twenty (20) amino acids and three stop codons formed from 64 (four nucleotide bases in permutation with a single codon, contribute to 4^3) triplets of codons (Pu, Gu, 2017).

The relationship between these amino acids and codons is summarized in a genetic code table. Multiple codons are used to represent a single amino acid as per the genetic code table (Central dogma (DNA to RNA to protein) | Biology library | Khan Academy, 2020). For instance, an amino acid “Serine” can be depicted in six different ways whereas “Alanine” can be written in four ways as per mRNA language and similarly for other amino acids. This genetic code is universal for all types of species ranging from bacteria to humans. With the help of this genetic code, all species carry out the process of protein synthesis in the body (Central dogma (DNA to RNA to protein) | Biology library | Khan Academy, 2020).



First letter	Second letter: U	C	A	G	Third letter
U	UUU Phe	UCU	UAU Tyr	UGU Cys	U
	UUC	UCC Ser	UAC	UGC	C
	UUA Leu	UCA	**UAA Stop**	**UGA Stop**	A
	UUG	UCG	**UAG Stop**	UGG Trp	G
C	CUU	CCU	CAU His	CGU	U
	CUC Leu	CCC Pro	CAC	CGC Arg	C
	CUA	CCA	CAA Gln	CGA	A
	CUG	CCG	CAG	CGG	G
A	AUU	ACU	AAU Asn	AGU Ser	U
	AUC Ile	ACC Thr	AAC	AGC	C
	AUA	ACA	AAA Lys	AGA Arg	A
	AUG Met	ACG	AAG	AGG	G
G	GUU	GCU	GAU Asp	GGU	U
	GUC Val	GCC Ala	GAC	GGC Gly	C
	GUA	GCA	GAA Glu	GGA	A
	GUG	GCG	GAG	GGG	G

Figure 5:This diagram depicts the genetic code table (Central dogma (DNA to RNA to protein) | Biology library | Khan Academy, 2020)

3.2 Types of Mutations in DNA

Mutations in DNA are nothing but an alteration in the DNA sequence. Mutations in DNA can be neutral, favorable, or detrimental to the human body (Pu, Gu, 2017). The effect of DNA mutation depends on the location at which the mutation has occurred and whether the underlying function of the protein has changed or not (Pu, Gu, 2017).

These mutations can be broadly classified into two primary types based on the size of the region it affects, such as large-scale mutation where the whole chunk of the chromosome is lost, reloaded, or rearranged and point mutation in which there is an alteration of single nucleotide base pair (Prof. Dave Explains, 2016).

A point mutation can be further classified based on the impact it makes on the proteins in the polypeptide chain, such as nucleotide pair substitution where single base pair is substituted either outside a gene then it does not cause any problems or if located within a gene then it results into mutations like silent mutation which involves a change in unit base pair (bp) leading to same amino acid and thus same protein, second is missense mutation occurring due to change in nucleotide bp resulting to a different amino acid, which in turn changes the protein (Prof. Dave Explains, 2016). Missense mutation often does not have severe implications but sometimes results in diseases such as sickle cell anemia and genetic disorder such as hemophilia (Prof. Dave Explains, 2016). Another type is nonsense mutation wherein substitution of base pair in mRNA strand results to code for stop codon which ultimately forms a partial complete protein (Pu, Gu, 2017).

Another type of point mutation is Insertion/Deletion. In this type of mutation, either base pair is inserted or removed from the mRNA strand (Pu, Gu, 2017). Such mutations have an enormous impact on the resulting protein. If nucleotide bp is inserted/deleted then the entire reading frame of DNA strand gets shifted which results in non-functional proteins. This mutation is known as a frameshift mutation (Pu, Gu, 2017).

Causes of Mutation:

Mutation in DNA can be caused due to internal or external factors. Mutations caused by cellular mechanism simply by performing a mistake by itself are spontaneous mutations that appear to be once per ten billion base pairs whereas mutations caused due to external factors such as UV radiation result in pyrimidine dimers (Prof. Dave Explains, 2016).

3.3 Hemophilia A

Computational biology and gene therapy are making revolutionized advancements for the treatment of many inborn disorders. Hemophilia A is among those genetic blood disorders wherein a person bleeds abnormally due to the absence of blood coagulating protein called Factor VIII. In genetic terms, Hemophilia A is caused by alteration/mutation of the HEMA gene on the X chromosome, particularly in males which results in reduced production or functioning of factor VIII (FVIII) protein (OMIM Entry - # 306700 - HEMOPHILIA A; HEMA, 2020). Based on the plasma levels of blood concentration factor VIII, the severity of Hemophilia A can be categorized as severe, with levels of factor VIII less than 1% of normal, moderate, with 2 to 5% of normal levels and mild, having factor VIII levels between 6 to 30% of normal (OMIM Entry - # 306700 - HEMOPHILIA A; HEMA, 2020). Patients suffering from severe hemophilia experience spontaneous bleeding occurring on an average of about twenty to thirty times within a year affecting their soft tissues and joints due to lack of circulating plasma FVIII activity (OMIM Entry - # 306700 - HEMOPHILIA A; HEMA, 2020). Most commonly affected joints could be shoulders, elbows, hips, knees, and ankles (OMIM Entry - # 306700 - HEMOPHILIA A; HEMA, 2020). If not treated, frequent bleeding leads to the development of chronic arthropathy.

(Plug et al.,2006) examined a population-based survey conducted in the Netherlands. They observed that the female population suffering from hemophilia A and B had more bleeding incidents after undergoing medical treatments such as tooth extraction or surgery of palatine tonsils, than the non-carrier women.

(Denson and Feinstein et al.,1969) exemplified with the aid of F8 antibodies, that hemophilia A patients have heterogeneous Factor VIII molecules. From which they inferred that Factor VIII levels in some patients can be brought up to normal level with antibodies while in remaining patients it is not possible to neutralize F8 levels using antibodies. Thus, this categorization led Denson et al. postulate that hemophilia A has two subtypes, one that lacks immune-boosting protein and other having immunologically normal but rather protein deficient in hemostat.

3.4 Inheritance and Hemophilia A

Hemophilia A is termed as a hereditary disorder related to the X-chromosome usually prevalent in males. In cases where it has been carried from generations, the affected offspring (male child) inherits the mutant gene from his mother carrying the defect (OMIM Entry - # 306700 - HEMOPHILIA A; HEMA, 2020). But it has also been observed that 30% of cases are derived from spontaneous mutation (Mannucci and Tuddenham,2001). Also, Hermann claimed that age plays a vital role in mutation rate among hemophilia A population whereas Barrai et al. discarded the fact that age of mother or age of maternal grandfather has an effect on the mutation rate in hemophilia.

The hypothesis that a notable proportion of new mutations causing hemophilia A can be linked to the occurrence of genetically two distinct populations of cells (mosaicism) among an individual was inspected by (Leuer et al.,2001). They carried out a study on sixty-one families, whose members had severe hemophilia A. Eight out of sixty-one families showed somatic mosaicism. These family members had point mutations (OMIM Entry - # 306700 - HEMOPHILIA A; HEMA, 2020). Further on subgrouping these eight families, four of which showed an increase in mosaicism (OMIM Entry - # 306700 - HEMOPHILIA A; HEMA, 2020). On the contrary, no mosaics ie. all cells have the same genetic structure was found in thirteen families having small insertion/deletion type of mutations. These data led to an inference that hemophilia A patients can sometimes show mosaicism (OMIM Entry - # 306700 - HEMOPHILIA A; HEMA, 2020). As a result, while performing genetic counseling the counselors should also consider the chances of mosaicism due to de novo mutations among hemophilia A population, most probably considering these types of mutations.

3.5 Hemophilia A treatment, gene therapy, and diagnosis

Current treatment for hemophilia A patient involves regular self-infusion of concentrates of exogenously derived FVIII. As these concentrates have short half-lives the patients require to monitor infusions every 2-3 days (SciMag2019). Also, due to the high costs of factor concentrates there is variation in access to this treatment worldwide. (SciMag2019)

Under genomics, genome editing which is an innovative gene therapy technique that precisely targets and repairs a genetic defect can be implemented for treating hemophilia A patients without frequent provision of artificial FVIII protein (Pipe S. and Selvaraj S.,2019). Gene editing allows scientists to alter an organism’s DNA. There are several methods of genome editing that have been developed. One of those is the CRISPR Cas9 approach. In a study (Pipe S. and Selvaraj S.,2019) that elaborates on a remedy for Hemophilia A through gene editing, it clearly states an approach “clustered regularly interspaced short palindromic repeats (CRISPR) cas9 tool”. This gene-editing technique is still not implemented at clinical levels.

Preimplantation Genetic Diagnosis

To reduce the risk of having an offspring with a sex-linked genetic disease, in the early 1990’s Preimplantation genetic diagnosis (PGD) was initiated as an alternative to prenatal diagnosis (Sermon, Van Steirteghem and Liebaers, 2004). In PGD, embryos developed in vitro are inspected for genetic defects; only those that are devoid of defects are replaced into the womb. PGD was first applied to patients who had X-linked genetic disease and had 25% chances of bearing an affected offspring (Sermon, Van Steirteghem and Liebaers, 2004). PCR is a technique used to analyze the DNA from cells to diagnose monogenic disorders like hemophilia A. But PCR requires extensive manual labour work to study monogenic disorders (Sermon, Van Steirteghem and Liebaers, 2004).

The ethical component of PGD

PGD can be said to be a prior stage of prenatal diagnosis and considered to have a similar diagnosis as prenatal. However, for a few diseases, PGD seems to be a strong choice from an ethical perspective rather than a prenatal diagnosis (Sermon, Van Steirteghem and Liebaers, 2004). Non-disclosure PGD can be implemented on diseases such as Huntington's disease and also other genetic disorders where patients only wish to have defect-free offspring and not interested in knowing the carrier status (male/female) (Sermon, Van Steirteghem and Liebaers, 2004).

3.6 Introduction to Machine Learning

Machine learning systems are classified as per the type of supervision received during the training phase. It is categorized into four major groups: supervised learning, unsupervised learning, semi-supervised learning, and reinforcement learning. (Géron, n.d.)

In Supervised Learning, the training data which is input to the algorithm has labels(classes). Supervised learning system has significantly two tasks, Classification wherein the system classifies new data as per the model devised by learning from a known set of labels and Regression is a task where we can predict the outcome from the predictors from the class labels (Géron, n.d.). Unsupervised learning deals with unlabelled training data where we can group similar data items. Semi-supervised learning has a lot of unlabelled data and some labelled data in the training dataset (Géron, n.d.). Semi-supervised learning algorithms are an amalgamation of supervised and unsupervised algorithms. Whereas, in the Reinforcement learning system an agent observes the environment, selects, and performs actions (Géron, n.d.). These actions are rewarded in the form of rewards or penalties (negative rewards) in return. The system learns by itself and formulates the best strategy(policy) to get the most rewards over the period. The policy specifies the action the agent performs in the given known environment. (Géron, n.d.)

3.7 Classification: A supervised machine learning technique

In Classification, unknown data instances are classified by the model by learning from a known set of labels of a categorical feature. If data instances d1, d2……dn are required to be allocated to a predefined set of classes C1, C2…. Cl then classification comes under one of the following types below (Sokolova and Lapalme, 2009).

Binary classification deals with classifying the input instance into a single class from a set of two unique classes C1 and C2. In this type of classification, class labels can be either objective and not dependent on manual evaluation or subjective and require manual assessment. Classes can be well-defined, vague, or combination of both (Sokolova and Lapalme, 2009).

Multi-class classification is where the input is classified into one, and only one class from ‘l’ (C1, C2…. Cl) different classes. Similar to a binary classifier, multi-class classification can also have categories that are objective or subjective, precise, or unclear (Sokolova and Lapalme, 2009).

Multi-labelled classifier classifies the data instances into multiple classes from ‘l’ unique categories. An example of this type of classification can be text mining of medical information wherein medical texts are allocated to different disease codes (Sokolova and Lapalme, 2009).

Hierarchical classification involves classifying input to a single class which is further subdivided into subclasses or aggregated to form superclass. The hierarchy of classes cannot be altered during classification. In the Bioinformatics field, predicting protein structure can be formulated as a hierarchical classification problem (Sokolova and Lapalme, 2009).

Let us discuss some supervised learning classification algorithms such as Naïve Bayes Classifier, k-Nearest Neighbour, Support Vector Machines (SVM), Decision trees and random forests and Neural networks.

- Naïve Bayes Classifier:

It deals with learning systems where each instance x represents a combination of attribute values and the target function f(x). Where the target function can have values from finite set V. Naïve Bayes approach classifies by assigning most probable target value from given attribute values (a1, a2, a3……an). (Géron, n.d.)

The advantages of Naïve Bayes are that the algorithm requires less computational time for training and has good performance as it removes irrelevant features (Géron, n.d.).

Disadvantages of Naïve Bayes can be stated as, it requires a lot of training data to obtain better results and has less accuracy as compared to other classification algorithms on a particular dataset with fewer records (Géron, n.d.)

- K-Nearest Neighbour:

This machine-learning algorithm classifies the data points based on similarity and proximity. It is an instance-based learning system. It is known as a lazy learner as it stores all the training data and builds the classifier only when a new unlabelled instance needs to be classified. These classifiers learn by comparing new instance with given training data samples which are similar to it. (Comparative Study of K-NN, Naive Bayes and Decision Tree Classification Techniques, 2016)

Merits of K-Nearest Neighbour are, the algorithm is easy to understand and implement, it is robust to noisy data, and the training of data is very fast (Comparative Study of K-NN, Naive Bayes and Decision Tree Classification Techniques, 2016).

Demerits of KNN algorithm are, as it is a supervised lazy learner so execution time is slow, hence increases the computational cost and has a memory limitation (Comparative Study of K-NN, Naive Bayes and Decision Tree Classification Techniques, 2016)

- Support Vector Machines (SVM)

Support Vector Machine approach is applied to a binary classification problem. SVM algorithm usually works on the principle of detecting a plane or hyperplane that highly separates two classes (Shouval et al., 2013). SVMs locate the optimal hyperplane with a maximum distance to the closest point of the two classes (Shouval et al., 2013). The SVM algorithm can handle both linear and nonlinear problems in classification and regression tasks (Shouval et al., 2013).

SVM has three advantages namely, it performs well when there is a reasonable margin of separation between classes, secondly, it is more effective in high dimensional spaces and is a memory-efficient algorithm (Shouval et al., 2013).

The disadvantages of SVM are it does not work up to the mark on large datasets. Also, when the data is noisy SVM is not suitable for such a dataset. In the case of classification, the Support Vector classifier allocates data points above and below the optimal hyperplane, there is no probabilistic explanation for the classification (Shouval et al., 2013).

- Artificial Neural networks:

These algorithms are inspired by neuronal learning. We can consider a neuron as a computational unit where weighted inputs are received from other neurons and processed (Shouval et al., 2013). If a certain threshold value is achieved then output is obtained. ANN’s collectively formed a model of neurons that are interconnected. Connections between these units have weights applied during the system training phase. The input nodes are used as attributes for prediction (Shouval et al., 2013). The output nodes deliver the possible outcomes predicted by the network (Shouval et al., 2013).

Merits of ANN’s are that model building is relatively easy as it requires less of statistical knowledge. It can very well seize nonlinearities between predictors and outcomes. It is very well capable of capturing interactions between predictors (Shouval et al., 2013).

Drawbacks of ANN’s are that they lack interpretability (black box model) ie. the model building structure is not transparent to the end-user and is prone to overfitting due to model complexity (Shouval et al., 2013).

3.8 Deep Learning Frameworks

Multiple deep learning frameworks have been developed which utilize the GPU computing power to its full potential to build a neural network with ease and train it efficiently. The most commonly used frameworks include Google-Tensorflow, Apache- MXNet, Theano, Microsoft -CNTK, Caffe, and others. These frameworks support various programming languages.

Caffe is a framework created by the Berkeley Vision and Learning Center (BVLC). In Caffe, a neural network can be easily developed by configuring layers and hyperparameters without any hardcoding (LeCun, Bengio, and Hinton, 2015). For implementing CNN's, Caffe can be one of the options due to its easy model development feature. However, the creation of a recurrent neural network is not supported by Caffe. Caffe supports the deployment of models on the cluster system (LeCun, Bengio, and Hinton, 2015).

CNTK was created by Microsoft Research to develop custom-build neural nets across multiple GPU's and servers. CNTK is a consolidated deep-learning toolkit (LeCun, Bengio, and Hinton, 2015).

TensorFlow which is developed by Google has a growing popularity among deep learning professionals due to its ability to construct flexible design neural networks (LeCun, Bengio, and Hinton, 2015). In TensorFlow, a neural network is nothing but a computational graph made up of nodes and connections among those nodes. It extends implementation not only on CPU's and GPU but also on multiple nodes of a cluster (LeCun, Bengio, and Hinton, 2015).

Theano which is developed by the University of Montreal has similar logic of developing the neural network when compared with Tensorflow as it creates computational graphs similar to TensorFlow. But, its proposition as a Deep learning framework was before TensorFlow. Theano is mostly used for evaluating mathematical expressions in the form of matrix values.

Apache MXNet is another open-source framework for deep learning. MXNet has a Gluon interface used to build neural networks on the cloud.

3.9 Prior research on bioinformatics and data science

Research by (Chang P.,2018) regarding genetic mutation identification and classification undermines to classify mutations in brain tumors with an accuracy of 94%. In this study, a model is devised to predict genetic mutation status in gliomas by using deep learning algorithms such as convolutional neural networks (CNNs). High accuracy was required in prediction which was achieved through CNNs rather than traditional machine learning algorithms (Chang P.,2018). Also, the study discusses using dimensionality reduction techniques such as principal component analysis to extract features of higher importance for classifying MR imaging dataset of gliomas (Chang P.,2018).

In a similar approach by (Wong C.,2017) emphasis is laid on categorizing how different deep learning methodologies can be applied in various areas of health informatics. Research explains different architectures such as Deep Neural Networks (DNN), Deep Belief Network (DBN), Deep Autoencoder (DA), Recurrent Neural Network (RNN) and Convolutional Neural Network (CNN) in deep learning with their pros and drawbacks and its usage in the particular domain of health informatics (Wong C.,2017). Also, it elaborates on concepts such as genetic variants wherein variations in genes are better understood by predicting splicing patterns (Wong C.,2017).

In parallel to splicing (Jenet E.,2017) contributes to the research on the detection of splicing patterns in genetic breast and ovarian cancer. This research throws light on the effects of variants of unknown significance on hereditary cancers

(Jenet E.,2017). The developed pipeline of RNA-Seq and bioinformatics to analyze abnormal splicing joints in targeted genes of cancerous tissues and also observed that the same pipeline can be integrated for analysis of any other gene disorder (Jenet E.,2017).

Research by (Dina Y. Mikhail) elaborates on detecting the pre-cancer stage by examining mutations in the TP53 gene. The approach proposed in this study for handling large databases such as Universal Mutation Database (UMDCell-line) with mutations in gene TP53, is by developing a data mining technique to diagnose the pre-cancer stage. In this research, a model with two stages is built which has the first step that checks for malignant mutations in the TP53 gene if any; using bioinformatics tools such as basic local alignment search tool (BLAST) and CLUSTALW and at the second stage the malignant mutations are classified as per cancer type (Dina Y. Mikhail). For classification, three feed-forward back propagation neural networks are generated and with high accuracy of 85%, the malignant mutations are classified at the pre-cancer stage (Dina Y. Mikhail).

The literature by (Yeager A.,2019), explores on use of Artificial Intelligence (AI) in gene editing using CRISPR technology to predict the type of repair made to DNA sequence cut. The article focuses on developing machine learning repair prediction programs that can be utilized commercially, however, the existing deep learning programs such as inDelphi and FORECasT (favored outcomes of repair events at Cas9 targets) developed by Wellcome Sanger Institute in the UK have different predictions for the similar cuts in identical cells (Yeager A.,2019). Hence those models require improvement in accuracy (Yeager A.,2019).

4. RESEARCH METHODOLOGY

4.1 Research design

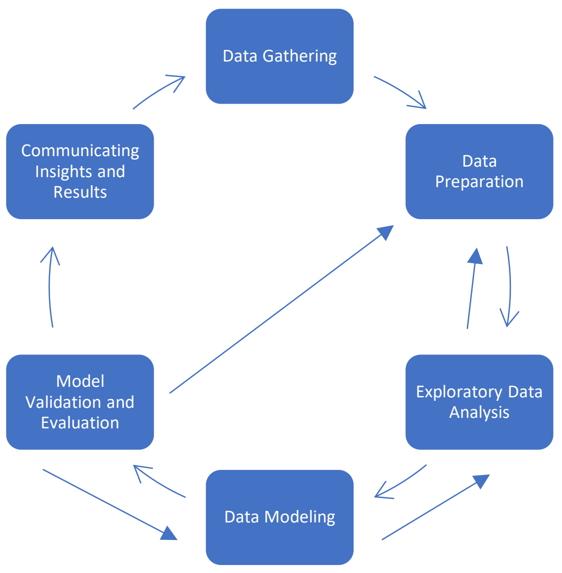

Data Gathering
Data Preparation
Exploratory Data Analysis
Data Modeling
Model Validation and Evaluation
Communicating Insights and Results

Figure 6:This image illustrates the research design framework

We will approach our research question by following a data mining model as elaborated below. The research design initiates with data gathering phase, the next step deals with the preparation of data, followed by exploratory data analysis phase, then we can perform data modeling, once we build a data model we can validate it against performance metrics and evaluate on test data and conclude by communicating insights and results to the desired audience.

Data Requirement and Gathering phase consists of identifying all available databases from which gene expressions and DNA sequences can be retrieved. Then once data is available, we can extract data into a usable format and store it for processing. After loading it we can assess data quality and relevance.

Data Preparation phase comprises of data processing tasks like examining the data at a high level by understanding various types such as qualitative and quantitative data, checking if any significant data is missing, and detecting data which may be an outlier. Data cleaning is part of the data preparation phase in which data can be transformed so that it is suitable for analysis. This involves handling of missing data, treating outliers, and reducing noise (unwanted data) from the dataset. Also, in this phase, we can identify the procedure on how to utilize the qualitative data for designing the model.

In the Exploratory Data Analysis phase, we check for correlation among features in the dataset, create visualizations to generate initial insights about data. From these visualizations, we can also identify patterns in data. If there are numerous features in data then we can perform feature extraction by selecting features of significance for analysis. Feature extraction can include dimensionality reduction techniques such as Principal Component Analysis.

The Data Modelling phase constitutes of examining different classification algorithms to build a data model. Selecting an algorithm and applying it to our dataset. Once models are created then we can assess those and select a good-fit model.

Data Model validation and evaluation phase initiates with evaluating the model performance by running it through industry-standard metrics. Next, we will validate the model on the test dataset. In this phase, if model over-performs or under-performs then we have to move to the data preparation phase and perform data transformation processes on the data and then re-iterate the cycle with prior phases else we can traverse to model building phase and can create another model with different features.

Communicating insights and results phase, we will visualize the insights generated from model performance metrics. These insights need to be effectively communicated to the desired non-technical audience by narrating a clear and actionable story to take business decisions on the problem.

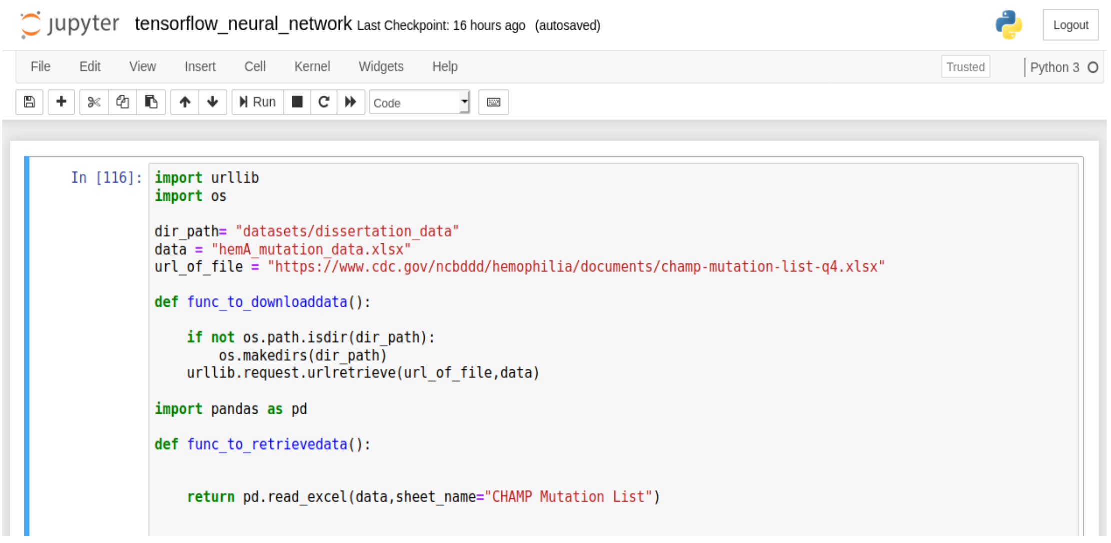
4.2 Data gathering and understanding of the data

Initially, our research question was to predict the type of repair in DNA mutations among hemophilia A patients. Data related to a specific gene, in this research we are interested in HEMA gene and DNA mutations can be searched from one among multiple data sources such as The International Genome Sample Resource, GenBank, the 100,000-genome project, Encyclopedia of DNA Elements (ENCODE) and National Center for Biotechnology Information (NCBI) database. We required data containing mutated DNA sequences and type of repair to correct the mutations among hemophilia A samples. As after exploring the above databases it was observed that historical data related to repair in DNA mutations for the HEMA gene was unavailable and we had a limited time frame to explore databases in depth. Thus, due to the unavailability of historical data to predict DNA repair type based on mutations, we modified our research question to understand DNA mutations in hemophilia A and predict the severity of mutations based on severity indicator from the report of hemophilia A patients.

Data required for analyzing and predicting the severity of mutations was retrieved from the Centers for Disease Control and Prevention (CDC) database. CDC is an organization that works under the U.S. Department of Health and Human Services to protect the health and safety of citizens of the US (Home | Hemophilia | NCBDDD | CDC, 2020). It acts as a health protection agency for the US (Home | Hemophilia | NCBDDD | CDC, 2020).

Under CDC, CDC Hemophilia A/B Mutations Project (CHAMP and CHBMP) keeps track of de novo mutations in hemophilia A and B. Project CHAMP consists of Factor VIII mutations in hemophilia reported worldwide (Home | Hemophilia | NCBDDD | CDC, 2020). Every mutation in the data is uniquely specified as per the Human Genome Variation Society (HGVS) nomenclature for DNA and protein changes (Home | Hemophilia | NCBDDD | CDC, 2020). Also, every single mutation severity is classified depending on the F8 levels in plasma as per the International Society on Thrombosis and Hemostasis (ISTH), depending on the history of antibody effect on the specific mutation and prior references to reports (Home | Hemophilia | NCBDDD | CDC, 2020). This database is updated annually with new mutations published worldwide.

From the CHAMP mutation database, we are interested in retrieving the excel file which contains the mutations data for Hemophilia A. We have automated the process to fetch the excel file and download it in the local system by developing a function in python. So, each year when the excel file is updated with upcoming mutations, we can retrieve the latest data by invoking the function. The python function to download data from the CHAMP website creates a directory if not already created and downloads an excel file by passing arguments, website link, and the name of the excel file to be downloaded. Then, to retrieve data we import from the pandas module, "read_excel" method, and explicitly provide an excel file name and sheet name as parameters. Later we store this data into a pandas data frame. Our data consists of 3756 rows and 20 columns.



```
import urllib
import os

dir_path= "datasets/dissertation_data"
data = "hemA_mutation_data.xlsx"
url_of_file = "https://www.cdc.gov/ncbddd/hemophilia/documents/champ-mutation-list-q4.xlsx"

def func_to_downloaddata():

    if not os.path.isdir(dir_path):
        os.makedirs(dir_path)
    urllib.request.urlretrieve(url_of_file,data)

import pandas as pd

def func_to_retrievedata():


    return pd.read_excel(data,sheet_name="CHAMP Mutation List")
```

Figure 7:This snapshot depicts the code snippet to automatically download and retrieve the data file

Data consists of multiple categorical features such as changes in coding DNA (cDNA) and protein as per HGVS nomenclature, type of mutation (Missense, Nonsense, Synonymous, Frameshift, large structural change, small structural change), type of DNA change(Insertion/Deletion, Substitution, Duplication, Inversion), the exon and codon of factor VIII where a mutation is located, severity indicator as per F8 levels, overall severity published, presence of antibodies among one or more patients of hemophilia A, the year when the mutation was first reported. Now, we observe that there are two features related to mutation severity. One among them is an indicator of F8 levels depending upon the concentration of factor VIII in plasma as compared to a normal person and another feature describes the overall severity of mutation as published.

Changes at coding DNA (cDNA) level and protein level are specified in data according to the HGVS nomenclature. So as per the nomenclature, a mutation can be represented in terms of DNA change and protein change. So, DNA level change can be due to substitution, deletion, duplication, insertion, deletion/insertion, and inversion of nucleotide base pairs. We have already seen the mechanism behind these mutation types in the Literature review. Here we will observe the method to represent these changes at DNA.

To first examine these rules to depict the changes in DNA base pairs, we will look into the concepts of exons and introns. Any gene in the human body is made up of multiple exons that are separated by introns (Exons, Introns & Codons, 2020). Exons are nothing but sequences in a region of a gene where protein is coded and they express these proteins in the form of codons, and thus named exons, whereas introns are intervening sequences that do not express proteins (Exons, Introns & Codons, 2020). The figure below makes this concept clearer.

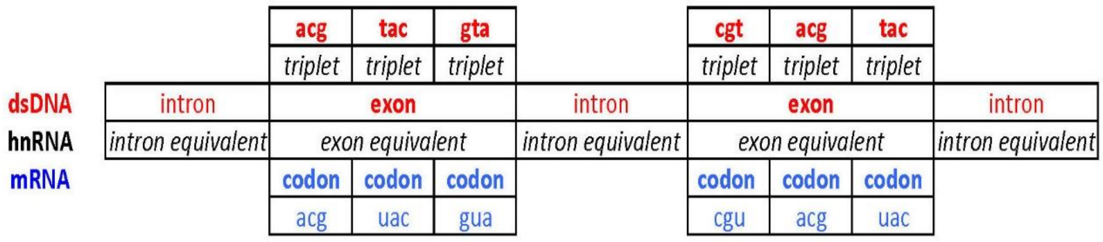


Figure 8:This figure shows exon and intron position in DNA sequence (Exons, Introns & Codons, 2020)

So now, let us first consider for instance substitution of G with T in the DNA sequence at nucleotide position 1009, so this is depicted as c.1009G>T, then substituting G with A at nucleotide position +1 of an intron between 1009 and 1010 nucleotides is shown as c.1009+1G>A, suppose if nucleotide G changes to T for 56 nucleotides at 3'(right)end of the termination codon then it is represented as c.*56G>T (HGVS recommendations: general, DNA level, 2020). Deletions are described in a similar way to substitution except indicating keyword del after specifying nucleotides to be deleted and separated by an underscore ie. deleting C nucleotide base at 1165 position is shown by c.1165delC, to delete nucleotides between uncertain positions we specify by writing c.-1171-?_1271+?del (HGVS recommendations: general, DNA level, 2020). Also, duplications are shown by the dup keyword at the end of nucleotide positions 1171 and 1184 such as c.1171_1184dup (HGVS recommendations: general, DNA level, 2020). If there are two or more consecutive nucleotides deleted and then inserted with other nucleotides then it is termed as deletion/insertion and described as c.1009+1_3delinsATT ie at position +1 intron between 1009 and 1010 deleting three nucleotides and inserting with ATT (HGVS recommendations: general, DNA level, 2020).

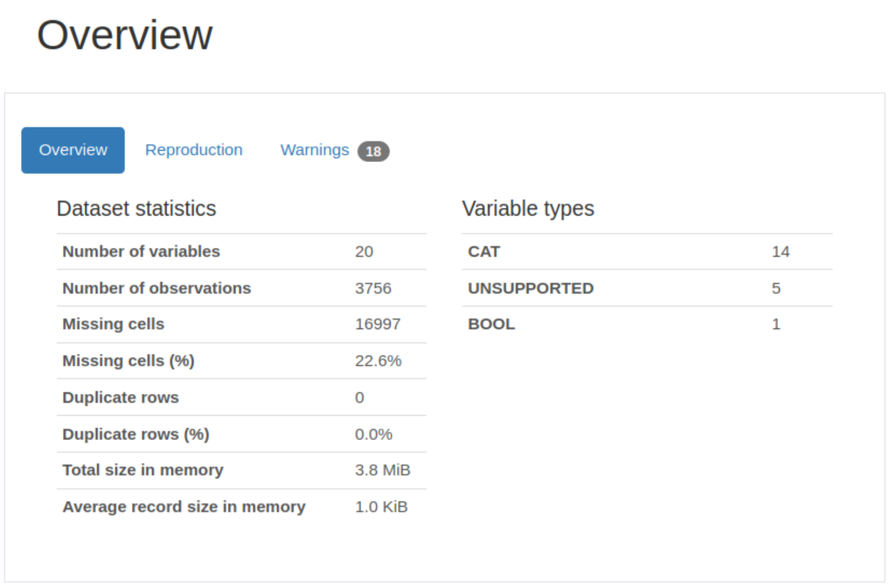
Similarly, sequence change at the protein level is described analogous to DNA level changes but with few modifications. Any protein level change begins with p. followed by amino acids, to left is a protein which has to be encoded as per the codon and to the right is protein to which it has changed. For example p.Lys396Glufs*4 this representation states that lysine protein has changed to glutamate (HGVS recommendations: general, DNA level, 2020).

Let us inspect other categorical features from our data. Type of mutation specifies the type of change in protein due to mutation. Categories in mutation type feature are missense, nonsense, frameshift, synonymous, large structural change (>50 bp), small structural change (<50 bp), and splice site change. Type of change in DNA caused by a mutation is explained by categories in Mechanism feature, these include substitution, deletion, duplication, insertion/deletion, inversion, and duplication/inversion. Severity indicator contains data about factor VIII levels such as severe, mild, and moderate.

Another feature related to the severity of mutation is "reported severity" which has class labels as Mild, Mild/Moderate, Mild/Moderate/Severe, Mild/Severe, Moderate, Moderate/Severe and Severe. These data values are as per mentioned in the publication and not according to F8 levels. This feature is our target feature on which we will apply classification. Also, a feature specifying whether the antibodies were enforced against the mutation has values as Yes, No and Not reported. Every unique mutation has a year associated with it stating the year in which the mutation was first observed in a human patient of hemophilia A.

4.3 Data Preparation and Exploratory data analysis

To get an overview of our data, we perform data profiling using pandas ProfileReport in Jupyter notebook. Pandas – Python data analysis library which is built on top of python programming language can be utilized for data manipulation and analysis. From data profiling, we get to know the number of records, number of variables, missing values, distribution of data for each variable, count of categorical and numerical variables, skewness in data for the entire dataset. We observe that most of our features are categorical (qualitative data) and for analysis, we require transformations and data cleaning on some features.


Overview

Overview | Reproduction | Warnings 18

Dataset statistics

Number of variables	20
Number of observations	3756
Missing cells	16997
Missing cells (%)	22.6%
Duplicate rows	0
Duplicate rows (%)	0.0%
Total size in memory	3.8 MiB
Average record size in memory	1.0 KiB

Variable types

CAT	14
UNSUPPORTED	5
BOOL	1

Figure 9:This snapshot shows data profile report

4.3.1 Data Quality Check

We come across an inconsistency in data values for the “Reported severity” feature. Same class values, for instance, ‘severe’ is mentioned in data as Severe, severe, or SEVERE. We achieve consistent data for this feature using the pandas-map function. Also, similarly, we obtain consistency in other features such as changes in DNA level (HGVS cDNA) and severity levels in plasma as per ISTH.

Completeness of data is another data quality dimension that needs to be investigated in our dataset. From the data profiling report, we can check for the occurrence of missing values in all features. We observe that features such as a change in protein (HGVS Protein) and domain to which the mutation belongs (Domain) have 14% missing values each. Features severe, mild and moderate have “x” marked as an entry against each mutation instance, hence more than 80% missing data is seen but we convert these features to one-hot encoded feature for every level ie. severe, mild and moderate by imputing zeros in place of missing values. Also, comments on mutations contain 96% missing values. Hence, we discard this feature with high null values.

We also check for the uniqueness of data from data profiling report and we come across a few features with high cardinality which suggests that these variables have unique values across all instances in data. The features with high cardinality are change in DNA (HGVS DNA) and change in protein (HGVS Protein).

4.3.2 Exploratory data analysis

After inspecting the data quality and pre-processing the data, we can perform an exploratory analysis of our data. Exploratory data analysis involves creating visualizations to find patterns in data so that we can generate insights. From these insights, we can have a deep understanding of data.

Altair:

For visualizing our data through interactive plots, we will be using Altair. Altair is a visualization library for python, which has API that is simple to use, requires less coding for plotting clear and significant graphs, renders graphs that can be saved in multiple formats, and is built on top of the Vega-lite visualization grammar (Altair: Declarative Visualization in Python — Altair 4.1.0 documentation, 2020).

We pass data in the form of a pandas data frame for visualization. The basic object in Altair is defined by a chart. Once we declare a chart object, next we can specify what kind of visualization we need to plot our data. For instance, a scatter plot, a bar chart, a line graph, a histogram, or a boxplot (Altair: Declarative Visualization in Python — Altair 4.1.0 documentation, 2020). This visualization aspect is handled in Altair with the help of the "mark" attribute of the chart object. For a scatter plot we specify mark_point (Altair: Declarative Visualization in Python — Altair 4.1.0 documentation, 2020). Further, we need to map the columns in data with the encoding channels such as x and y axes, color parameters by using the encode method (Altair: Declarative Visualization in Python — Altair 4.1.0

documentation, 2020). Altair can automatically determine the data type of columns from our data frame so there is no need to explicitly mention it in our visualization grammar (Altair: Declarative Visualization in Python — Altair 4.1.0 documentation, 2020). Altair proves to be much flexible than other visualizing tools with respect to a data aggregation (Altair: Declarative Visualization in Python — Altair 4.1.0 documentation, 2020). It provides us with in-built methods for aggregation of data. Customization of visuals is easy in Altair by just passing parameters such as the color of the graph, title for graph, and many more (Altair: Declarative Visualization in Python — Altair 4.1.0 documentation, 2020). Thus, after generating graphs if we need to publish those it is quite straightforward in Altair. Altair uses the Vega-Embed package for publishing the visualizations as a simple HTML document (Altair: Declarative Visualization in Python — Altair 4.1.0 documentation, 2020).

Initially, we are interested in looking at reported severity of mutations, as this feature is our target variable for classification. By finding the number of mutations as per reported severity can dive us deeper into the analysis of the specific type of severity in mutations among hemophilia patients. Thus, we created a bar chart showing the count of mutations based on reported severity and observed that severe mutations constituted the majority of data ie. 60% of de novo mutations across 1985 to 2019. Also, we plotted a line graph to see the overall trend in mutations for the past thirty-five years. We can observe that there are spikes in the graph from 2008 to 2019.

Later, as we performed an analysis on severe mutations, we were more interested to find out that severe mutations are caused predominantly by which type of mechanism that changes DNA bases and thus changes the underlying proteins specified by the type of mutation. We generated a bar chart depicting the number of severe mutations under each category of mechanism further subgrouping it on mutation type. So, we came up with a graph stating that among severe mutations in hemophilia A majority are substitutional changes in DNA which is nothing but change in nucleotide base pair which results to code different protein than expected (substitution > missense mutation).

The feature HGVS cDNA has values composed of nucleotide position at which change has occurred and change in base pair. Thus, to find out at which nucleotide positions maximum severe substitutional mutations occur in hemophilia A we plotted a graph depicting a range of nucleotide positions and observed that between a range of 0 to 2000 nucleotide position majority severe mutations are seen. Also, between nucleotide position from 5000 to 7000 around 40% severe substitution mutations are located.

To find a trend about change in substitutional base pairs under severe mutations belonging to either missense, nonsense, splice site change, and synonymous mutations, we plot a bar chart illustrating the number of mutations versus change in base pair. So, we have twelve nucleotide base pair changes such as A changes to C (‘A>C’), ‘A>G’, ‘A>T’, ‘C>A’, ‘C>G’, ‘C>T’, ‘G>A’, ‘G>C’, ‘G>T’, ‘T>A’, ‘T>C’ and ‘T>G’. We observe from the graph that base pair change of G to A (16% of total mutations) followed by G to T (13% of total mutations) is higher among other base-pair changes. Also, under severe substitutional base change of G to A majority is due to missense mutation whereas in G to T majority is a nonsense mutation.

4.4 Data Modelling

Once we have performed exploratory data analysis and discovered interesting patterns and trends from our data, next we create a model for classifying mutations as per severity. We observe that our data contains mutation severity with seven categories with the class imbalance of severe mutations accounting to 60% of class labels and the remaining six categories constituting 40% class labels. Hence to avoid class imbalance we convert our multi-class classification problem to binary classification. We convert the class labels other than severe to a category as non-severe. With this, we are also able to achieve a striking distinction between severe and mild/moderate cases in hemophilia A, as patients with severe hemophilia A require medical care of high precision as compared to mild/moderate patients.

4.4.1 Study on Deep Learning

To develop a binary classifier to classify severe and non-severe mutations in hemophilia A we require a model with high accuracy, as misclassification can result in detrimental consequences affecting the hemophilia population worldwide. So, to achieve a highly accurate model we will use artificial neural networks as our underlying technique to develop a good-fit model.

4.4.1.1 Artificial neural networks

The simplest neural network comprising of one or more binary inputs and a single binary output was formulated along the lines of biological neuron network architecture by Warren McCulloch and Walter Pitts (LeCun, Bengio and Hinton, 2015). They suggested that a network of artificial neurons can be built using this simple model to perform logical expression evaluations (LeCun, Bengio and Hinton, 2015).

4.4.1.2 The Perceptron

In 1957, Frank Rosenblatt discovered another simplest artificial neural network (ANN) known as the Perceptron (Géron, n.d.). The Perceptron architecture is slightly different from than model proposed by Warren and Walter in such that, it contains an artificial neuron called as linear threshold unit (LTU) which has numbers as inputs and output instead of binary values (Géron, n.d.). Also, weights are associated with inputs and the LTU calculates the summation of weighted inputs, then passing it through a step function and thereby generating an output (Géron, n.d.). A single layer of LTU's is present in a Perceptron in which every neuron is interlinked with all the inputs. Another component of a Perceptron is a bias neuron which has an output of unit magnitude and the other input neurons that provide an output based on the input that is inserted (Géron, n.d.).

To overcome a few limitations of the Perceptron such as the implementation of exclusive OR (XOR) classifier, multiple Perceptrons can be piled up to form a multi-layer perceptron (MLP) (Géron, n.d.). A multi-layer perceptron is formed with one input layer, one or more hidden layers consisting of LTU's and a single output layer which is the last layer of MLP (Géron, n.d.). Each layer apart from the output layer in MLP is interconnected to the corresponding next layer and contains a bias neuron. When an MLP has two or more hidden layers we can say it is a deep neural network (DNN).

4.4.1.3 Backpropagation algorithm and working of DNN

The working of the backpropagation algorithm begins with making a prediction for each instance of training data (Géron, n.d.). Then the error is calculated by taking the difference between actual and predicted values. Later the algorithm makes a reverse pass in which it traverses in reverse order starting with the layer prior to the output layer to measure the contribution of the error made by each connection (Géron, n.d.). After passing through all layers in reverse order till it reaches the input layer, the last step that the algorithm performs to reduce the error at each connection is to adjust the weights (Géron, n.d.). This last step is similar to gradient descent.

Similar working is observed in deep neural networks (DNN) used for binary classification wherein inputs are fed to a classifier which makes a prediction by learning the data and classifying data instances into one of the classes (LeCun, Bengio and Hinton, 2015). Error is backtracked and each output at the output layer of the DNN corresponds to a different binary class (LeCun, Bengio and Hinton, 2015). In DNN architecture to enhance the performance by tweaking the weights, activation functions are added. These activation functions can be rectified linear unit (ReLU) function and hyperbolic tangent function (LeCun, Bengio and Hinton, 2015). In the case of a multi-class classification problem, the output layer is altered by changing separate activation functions for each class with shared softmax function.

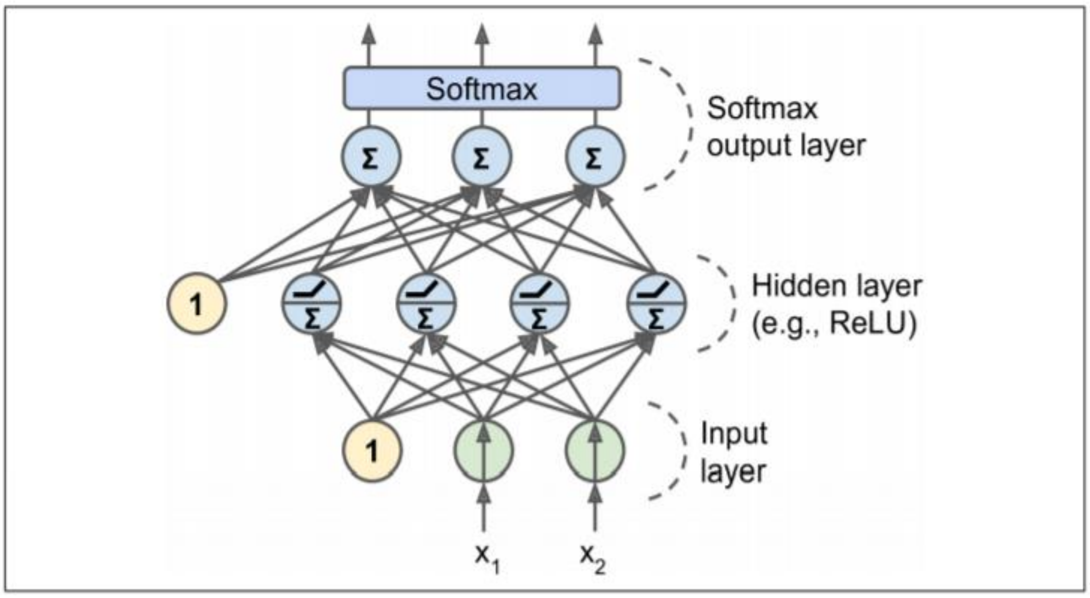


Figure 10:This diagram depicts the architecture of a multi-layer perceptron (Géron, n.d.)

4.4.2 Creating a DNN model

As seen in the above theory on neural networks, all the inputs to any ANN have to be numeric data that can be fed to any classifier. So, we need to convert the categorical features from our data into numeric variables. In machine learning the task to transform the categorical variables to numeric is termed as feature encoding. From data quality inspection we observed that few features in our dataset such as a change in DNA (HGVS cDNA) and change in protein (HGVS protein) have high cardinality. Hence, performing one-hot encoding on these features will generate thousands of features. From a vast set of features, we will have to select a subset by applying dimensionality reduction techniques. But, despite having a subset of features we would require machines with GPU that has the high processing speed to handle an enormous number of features. Instead, we will be using the embedding technique from Natural Language Processing (NLP) to handle high cardinality issues in our dataset.

Word Embedding:

Word embeddings are a mapping technique in which words are mapped to a vector of numbers. The logic behind this grouping is that words that express similar meaning or the ones which go together are associated with analogous numeric vectors and the model can utilize the learned information about few words to develop learning about other similar words (Lopez, Merlino, Rodriguez, 2017). Working of embeddings is such that the context in which a new word is represented is identified by knowing the surrounding words around it and if the word/s in the vicinity appear to be similar to the new word then the new word is added to the list of learned words (Lopez, Merlino, Rodriguez, 2017). Word embeddings are mostly implemented in natural language processing tasks like topic modeling and predicting the syntactically similar words (Lopez, Merlino, Rodriguez, 2017). A substantial research work carried out by Mikolov et al. on word embeddings represents words as multi-dimensional vectors with fewer dimensions (Dekhtyar, Fong, 2017). They developed a model known as Word2Vec based on analyzing the word embeddings. Mikolov et al suggest that using Word2Vec one can perform word embeddings even with text having fewer data without any difficulty in analyzing the relationship among the words (Dekhtyar, Fong, 2017).

So, to build our model we propose an architecture of a dense neural network as shown in the figure below. Initially, we performed label encoding on all the categorical features as we need numerical inputs in the input layer. Next, we apply categorical embeddings to all features of our data. Categorical embedding is the remodeling of word embedding on categorical variables (Loew, Spindler, Brechmann, 2018). Embedding layer is essential in our data model as we have variables such as a change in DNA (HGVS cDNA) and a change in protein

(HGVS Protein) with high cardinality as each value represented in these features is unique. So, the embedding layer is a perfect choice in case of our data as one-hot encoding implementation can result in a lot many parameters and eventually lead to an overfitting issue (Loew, Spindler, Brechmann, 2018). The benefits of adding an embedding layer to our neural network are, it decreases the number of features from our data model and increases the time required for training data to learn about the system and thus drastically improve model results (Loew, Spindler, Brechmann, 2018).

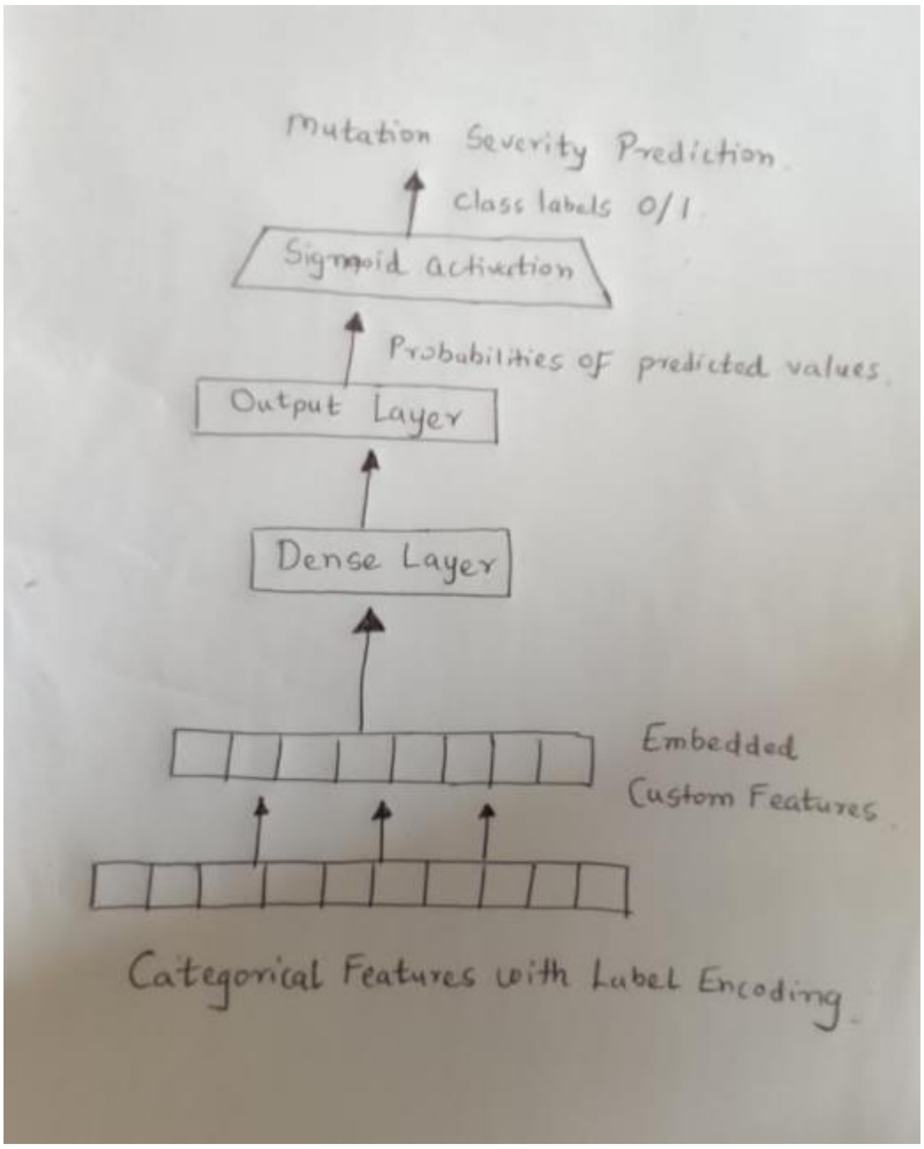
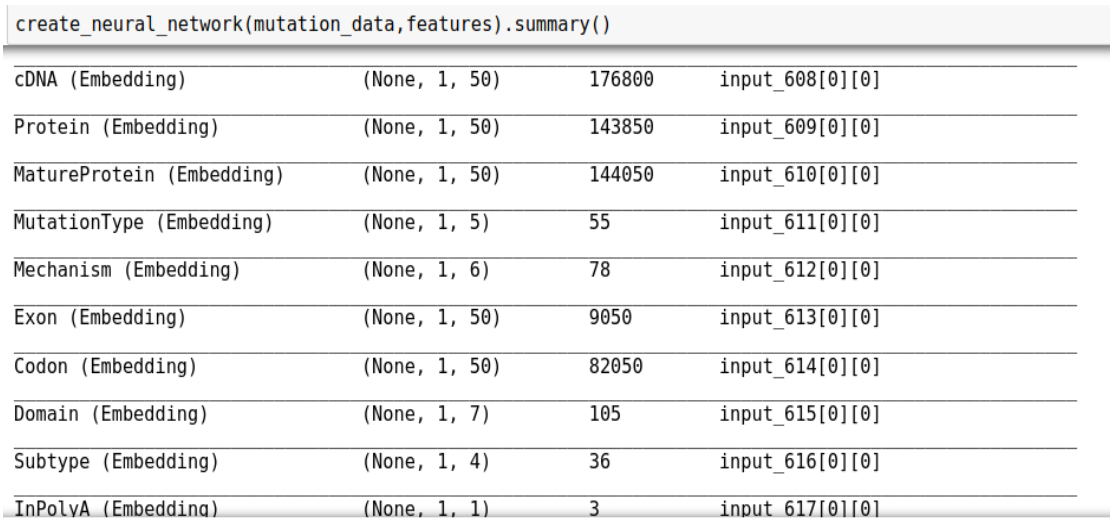

Mutation Severity Prediction
Class labels 0/1
Sigmoid activation
Probabilities of predicted values
Output Layer
Dense Layer
Embedded Custom Features
Categorical Features with Label Encoding

Figure 11: This diagram shows the architecture of a dense neural network implemented in the model

Input to embedding layer is our input layer which has all categorical features with label encoding. Embedding layer is applied to all features of our dataset except the target variable, Reported Severity. We have developed our embedding layer by passing parameters such as the number of unique values in each feature and dimension of the embedding layer. We have calculated the dimension of embeddings for all the features as the minimum value between fifty or half of the number of unique values in a feature. For instance, the feature HGVS cDNA has 3536 unique values then considering half of it which means 1768 and comparing it with 50, the minimum between both the values ie. 50, in this case, is chosen as embedding dimension for a particular feature HGVS cDNA. We have selected 50 randomly, we can choose a higher number for a dataset with millions of records to incorporate more categories into features but ideally, for our dataset with 3536 rows, it is sufficient to assign an upper limit of 50 to embedding dimensionality. Embedding dimensionality reduces the number of categories in our feature, representing feature values as 50-dimension vectors, which in turn reduces the curse of dimensionality.



```
create_neural_network(mutation_data,features).summary()

cDNA (Embedding)              (None, 1, 50)      176800      input_608[0][0]
Protein (Embedding)           (None, 1, 50)      143850      input_609[0][0]
MatureProtein (Embedding)     (None, 1, 50)      144050      input_610[0][0]
MutationType (Embedding)      (None, 1, 5)       55          input_611[0][0]
Mechanism (Embedding)         (None, 1, 6)       78          input_612[0][0]
Exon (Embedding)              (None, 1, 50)      9050        input_613[0][0]
Codon (Embedding)             (None, 1, 50)      82050       input_614[0][0]
Domain (Embedding)            (None, 1, 7)       105         input_615[0][0]
Subtype (Embedding)           (None, 1, 4)       36          input_616[0][0]
InPolyA (Embedding)           (None, 1, 1)       3           input_617[0][0]
```

Figure 12:This snapshot depicts the output of the embedding layer from the neural network

.

After creating an embedding layer for all features, we concatenate all the embedded features into a single output vector. Next, we create a dense layer with 32 neurons and activation function as a rectified linear unit (ReLU). Using ReLU as an activation function is advantageous as it has fast computation speed and in the gradient descent process, it reduces issues faced as it does not have maximum output value. The last layer of our neural network is the output layer which is connected by a dense layer. In the output layer, we pass the activation function as sigmoid as we are dealing with binary classification problem. After passing through the sigmoid activation function we have output values as probabilities of class labels, we convert those probabilities according to threshold value into class label values of 0 or 1 as in case of binary classifier.

4.4.3 Validating Classifier model

We have split our dataset into training and validation data in the ratio 70:30 respectively. After generating a neural network, we assigned it to a model and compiled it. During the model compilation, we aim at minimizing the loss by using binary cross-entropy as a loss function. Once we have compiled and finalized with a good-fit model we validated our model on validation data. Also, we have a test dataset which comprises of de novo mutations among hemophilia A population in the US. This dataset is retrieved from the CDC database itself. In this dataset, we have almost similar features from our training dataset and few extra features such as country of origin, race, ethnicity, and country where the report is generated. But we have removed these additional features from our analysis as they contain a lot of missing and constant values. The target feature which is reported severity is not provided in test data and has to be predicted for all mutations occurred in the US. As per the model, accurate predictions have been achieved.

5. RESULTS AND FINDINGS

First, let us discuss findings from the exploratory analysis phase. As observed in the research methodology, we find interesting trends and patterns in our mutation data.

5.1 Exploratory analysis findings

To explore the overall trend of mutations across the period from 1985-2019, we plotted a line graph showing the period from 1985 to 2019 against the count of de novo mutations reported in each year. From this plot (figure 13) we can observe spikes from 2008 onwards at every alternate year. In 2008, the highest number of new mutations (411) in hemophilia A were discovered whereas in the year 1986 nearly 3 new mutations among hemophilia A patients were found. We can infer from these figures that prior to 2004, the number of hemophilia A patients with new mutations were less except in the year 2002. Later after 2008, the number of cases with de novo mutations increased sharply showing a pattern wherein each alternative year number of new mutations arise and then fall the next year

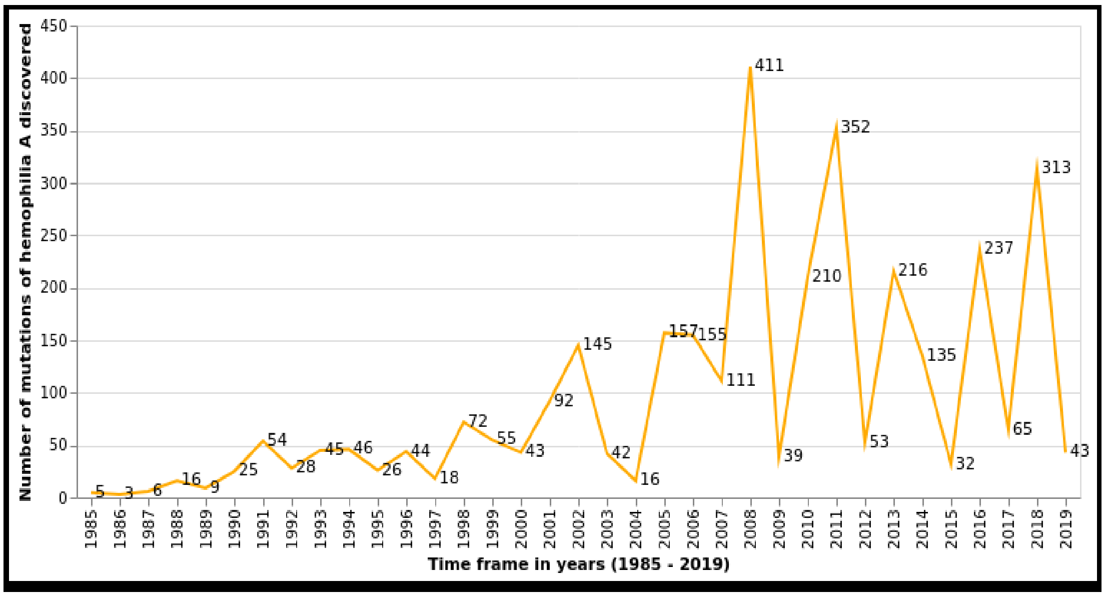


Figure 13:This line graph shows a trend in novel hemophiliac mutations across 1985-2019

We explored data related to mutation severity as reported in publications, as we are interested in predicting its outcome. We created a bar chart depicting the number of new mutations as per the severity reported. We infer from the graph (figure 14) that nearly 60% of mutations are severe whereas 21% are mild mutations and 13% are moderate mutations while remaining 6% mutations are under the category of Mild/Moderate, Mild/Severe, or Moderate/Severe.

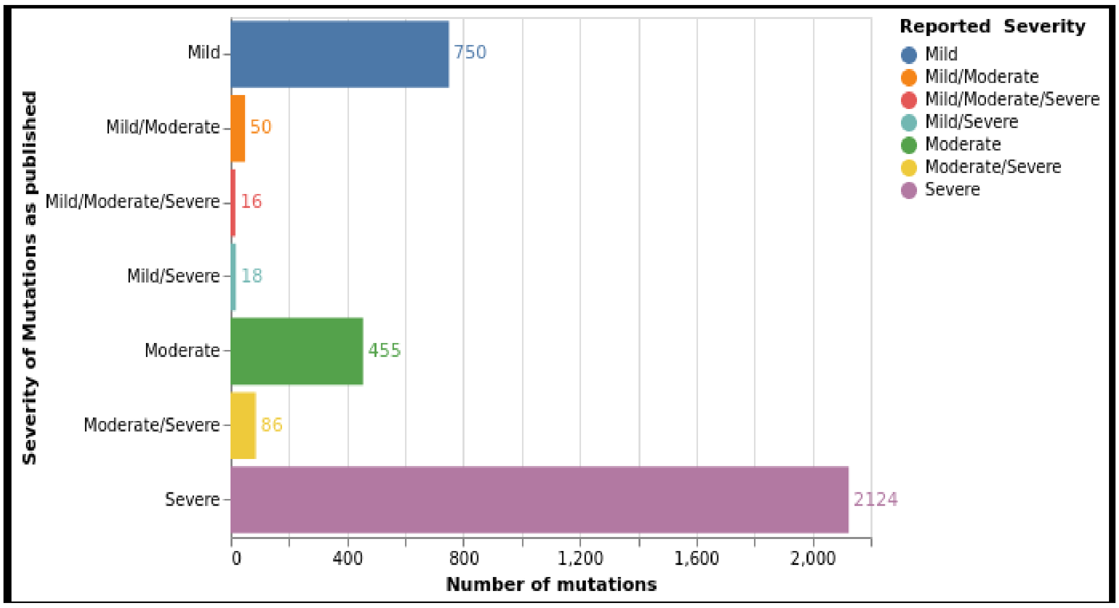


Figure 14:This bar chart shows the count of de novo mutations as per the severity

As we notice in the above graph that severe mutations constitute for the majority, we would like to dive deeper into analysis related to severe mutation types. So, we plot a graph showing number of severe mutations caused due to a particular DNA change mechanism. Also, we further analyze the type of mutation for each DNA change mechanism. It can be seen that almost 90% of severe mutations in hemophilia A are caused due to three mechanisms namely substitution, deletion, and duplication of nucleotides. We observe from the bar chart (figure 15) that 1038 severe mutations out of 2124 (almost 49%) are caused due to the substitution of nucleotide base pair. From these 1038 substitutional mutations, 53% are missense mutations, 32% are nonsense mutations, 14% splice site change, and 1%

synonymous mutations. Whereas 31% (661 out of 2124) new severe mutations are seen because of the deletion of nucleotide/s. Out of these 661, 70% are frameshift mutations, almost 20% large structural change(>50bp), 5.5% small structural change (<50bp) and 4.5% are splice site change mutations. And duplication mechanism accounts for nearly 9% of severe mutations. Out of this 9%, 85% are frameshift mutations, and the remaining 15% are large structural change (>50 bp) mutations. Thus, we can observe that in mechanisms such as deletion, duplication, insertion, and deletion/insertion, frameshift type of mutations mostly occurs. Whereas in the Substitution mechanism, missense mutations are the most.

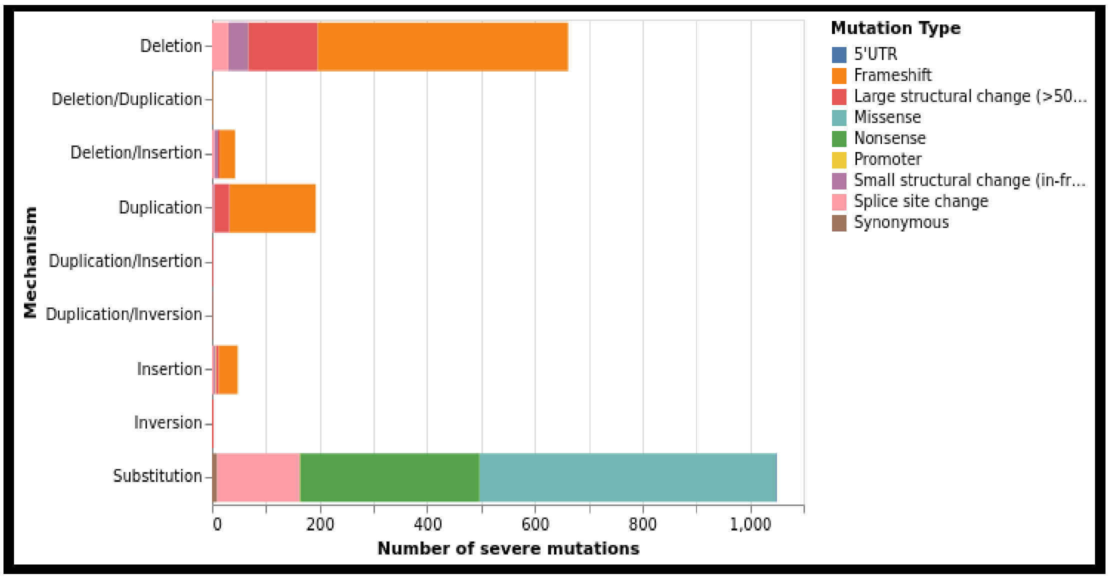


Figure 15:This bar chart depicts the number of severe mutations based on mechanism and mutation type

Next, we would like to examine at which specific nucleotide positions do the substitutional mutations occur. We have extracted the nucleotide position and base pairs from feature HGVS cDNA. So, in this, we found an interesting pattern by generating a bar chart (figure 16) of the count of substitutional severe mutations against a range of nucleotide positions. We discover that nearly half of the substitutions (46%) occur between nucleotide positions 0 to 2111. In the range of nucleotide positions from 4925 to 7035, nearly 38% of substitutional severe mutations are encountered. In the range of 2111 to 4925 fewer substitutions are noticed. By locating the nucleotide positions at which severe mutations take place in the HEMA gene, researchers and medical practitioners can decide on an effective target therapy for hemophilia A.

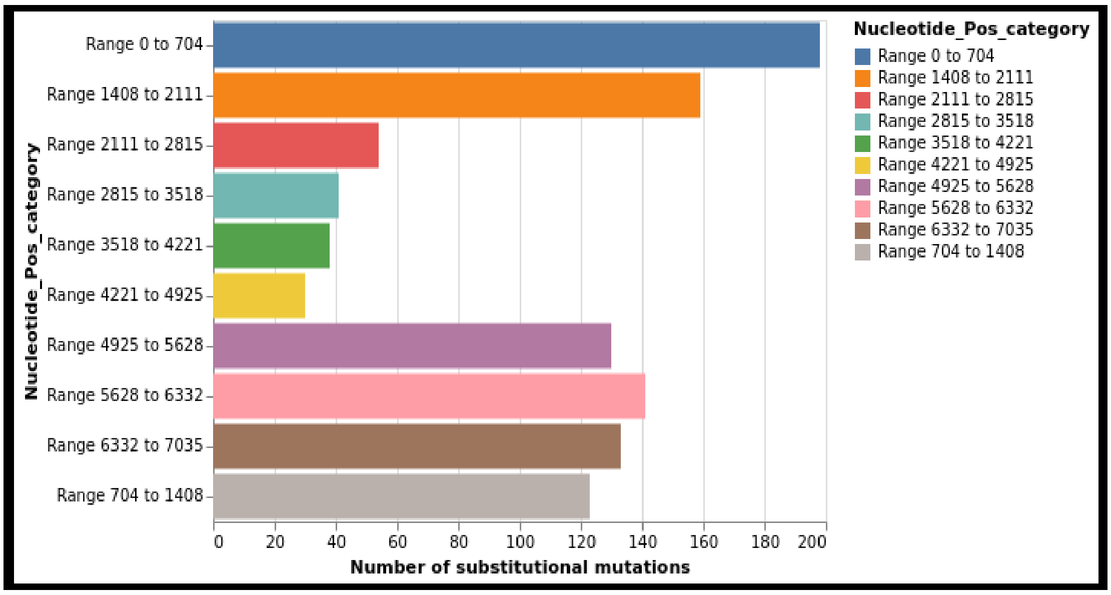


Figure 16:This bar chart depicts the number of severe substitutional mutations and range of nucleotide position at which mutation has occurred

Another striking trend that we explored is about which substitutional base pairs are mostly observed to cause severe mutations. In the plot (figure 17) we map a number of severe mutations due to substitution against the DNA level base pair changes and further subgroup by mutation type. We can observe from the bar chart that severe substitutional mutations are caused mainly when the nucleotide G changes to A leading to code proteins that cause most missense mutations (87), followed by nonsense mutations (49) and then splice site change mutations (36). Also, when G substitutes with T severe mutations are predominantly seen, followed by C-to-T substitution. This drives our attention that mostly nucleotide G changing to A/T is of concern as it causes severe mutations in hemophilia A population. Whereas, nucleotide change of A to C is least expected to cause severe substitutional mutations.

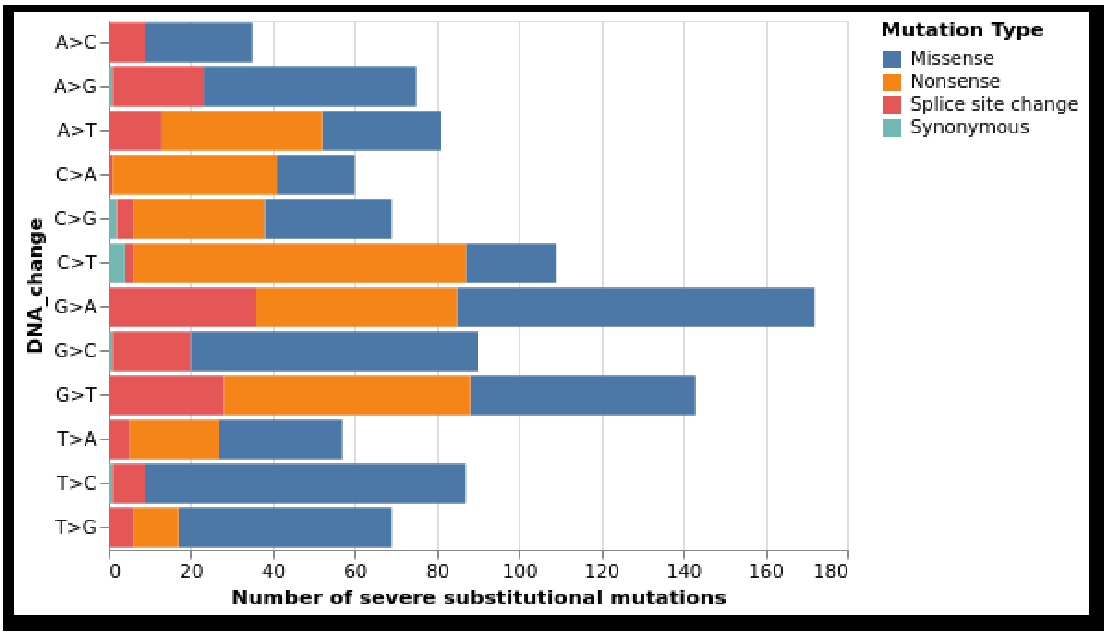


Figure 17: This bar chart depicts the number of severe substitutional mutations as per base pair change

Thus, to sum up on trends found in hemophilia A mutations we can say that 60% de novo mutations are severe. Among these severe mutations, nearly half of the mutations are caused due to substitution. Thus, after inspecting substitutional mutations in detail we noticed that nucleotide position from 0 to 2111 and 4925 to 7035 have a change in base pairs and particular nucleotide G changes to A and G changes to T are predominant substitutional mutations.

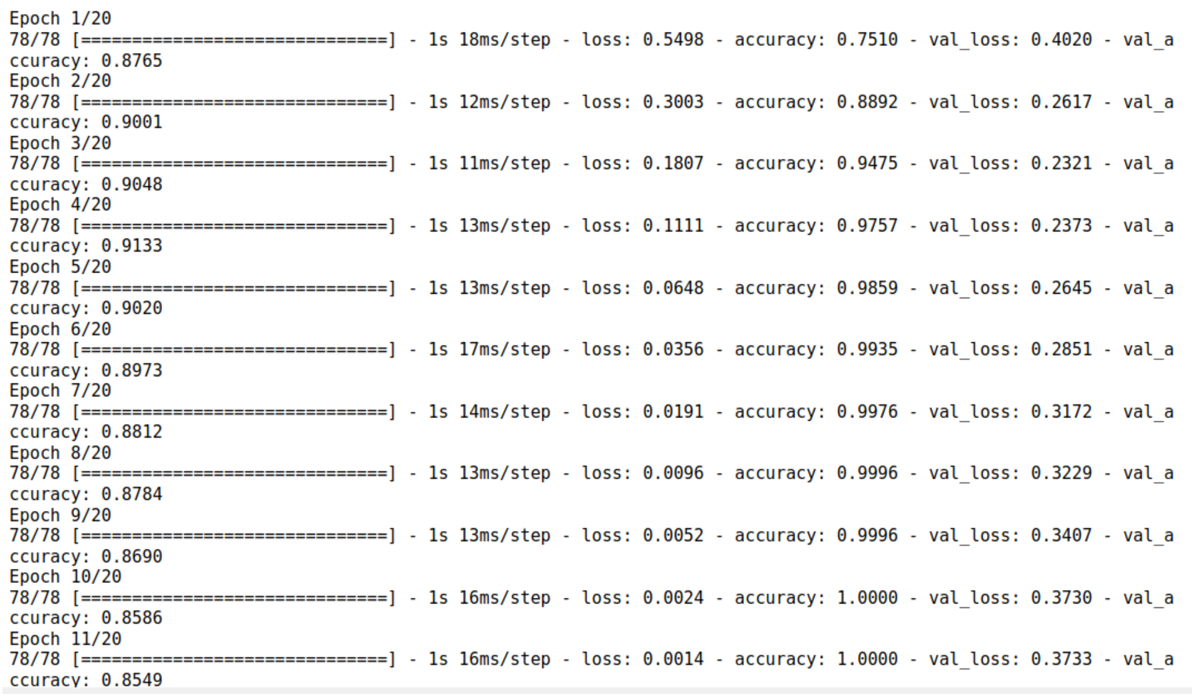
5.2 Binary Classifier Results

Now as we noticed that severe mutations are of concern in hemophilia A, we created a binary classifier to predict if mutations are severe or non-severe based on the change in DNA, change in the protein, mutation type, mechanism of DNA change, the concentration of factor VIII levels and inhibitor history. In our classifier, we have class labels 0 corresponding to non-severe mutations whereas class labels 1 represent severe mutations in hemophilia A.

5.2.1 Model evaluation metrics

5.2.1.1 Classification Accuracy

Once the model was developed using dense neural networks, we validated it on training as well as validation data. We can observe that we achieved an accuracy of 100% on training data whereas 85% accuracy on our validation data.



```
Epoch 1/20
78/78 [==============================] - 1s 18ms/step - loss: 0.5498 - accuracy: 0.7510 - val_loss: 0.4020 - val_a
ccuracy: 0.8765
Epoch 2/20
78/78 [==============================] - 1s 12ms/step - loss: 0.3003 - accuracy: 0.8892 - val_loss: 0.2617 - val_a
ccuracy: 0.9001
Epoch 3/20
78/78 [==============================] - 1s 11ms/step - loss: 0.1807 - accuracy: 0.9475 - val_loss: 0.2321 - val_a
ccuracy: 0.9048
Epoch 4/20
78/78 [==============================] - 1s 13ms/step - loss: 0.1111 - accuracy: 0.9757 - val_loss: 0.2373 - val_a
ccuracy: 0.9133
Epoch 5/20
78/78 [==============================] - 1s 13ms/step - loss: 0.0648 - accuracy: 0.9859 - val_loss: 0.2645 - val_a
ccuracy: 0.9020
Epoch 6/20
78/78 [==============================] - 1s 17ms/step - loss: 0.0356 - accuracy: 0.9935 - val_loss: 0.2851 - val_a
ccuracy: 0.8973
Epoch 7/20
78/78 [==============================] - 1s 14ms/step - loss: 0.0191 - accuracy: 0.9976 - val_loss: 0.3172 - val_a
ccuracy: 0.8812
Epoch 8/20
78/78 [==============================] - 1s 13ms/step - loss: 0.0096 - accuracy: 0.9996 - val_loss: 0.3229 - val_a
ccuracy: 0.8784
Epoch 9/20
78/78 [==============================] - 1s 13ms/step - loss: 0.0052 - accuracy: 0.9996 - val_loss: 0.3407 - val_a
ccuracy: 0.8690
Epoch 10/20
78/78 [==============================] - 1s 16ms/step - loss: 0.0024 - accuracy: 1.0000 - val_loss: 0.3730 - val_a
ccuracy: 0.8586
Epoch 11/20
78/78 [==============================] - 1s 16ms/step - loss: 0.0014 - accuracy: 1.0000 - val_loss: 0.3733 - val_a
ccuracy: 0.8549
```

Figure 18: This snapshot gives the output of a binary classification model as training, validation accuracy, and loss.

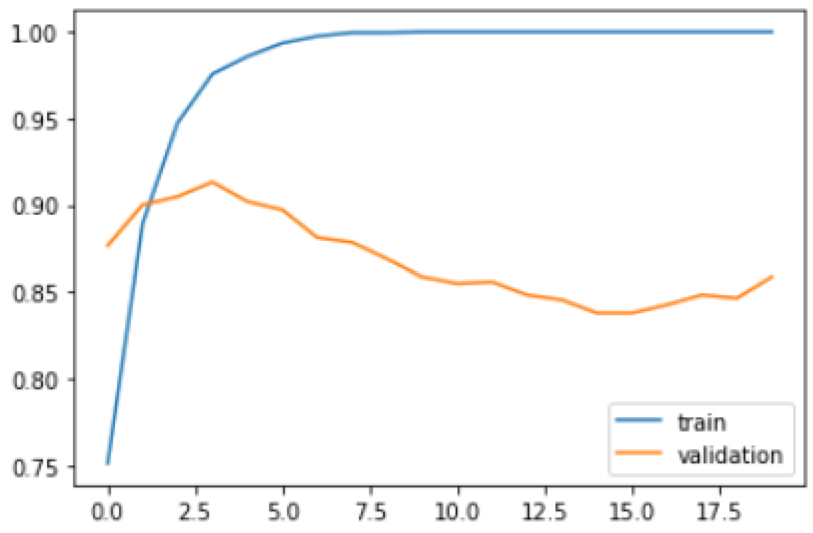


Figure 19:This graph illustrates the training and validation accuracy of the binary classifier with 20 epochs

.

Classification accuracy cannot be considered as one of the strong evaluation metrics as a model can perform well with high accuracy on training data but performs poorly on the validation data. Then it is a sign of overfitting. But in the case of our model, we achieve accuracy of about 85% on validation data as well as compared to the accuracy of 100% on training data. The loss function is minimized severely on training data as compared to validation data.

Few disadvantages of classification accuracy are that it does not specify the types of the error made by the classifier and it does not explain the underlying distribution of predicted values.

5.2.1.2 Confusion matrix

To overcome the limitations of classification accuracy we validate our model against the confusion matrix. To form a confusion matrix for our binary classifier, we require a count of instances correctly predicted to a positive class known as True Positives, the number of instances correctly predicted to a negative class called True Negatives (Evaluating a Classification Model, 2020). Also, we calculate the instances incorrectly predicted to positive and negative classes (False Positives and False Negatives) (Evaluating a Classification Model, 2020). In case of our research, true positives are those mutations which are severe and also predicted as severe, true negatives constitute those mutations which are non-severe and correctly recognized by the classifier as non-severe mutations, false positives are those mutations which fall under non-severe category but predicted as severe mutations whereas false negatives are mutations that are severe but predicted as non-severe. These false positives give us Type I error whereas false negatives specify a Type II error (Evaluating a Classification Model, 2020).

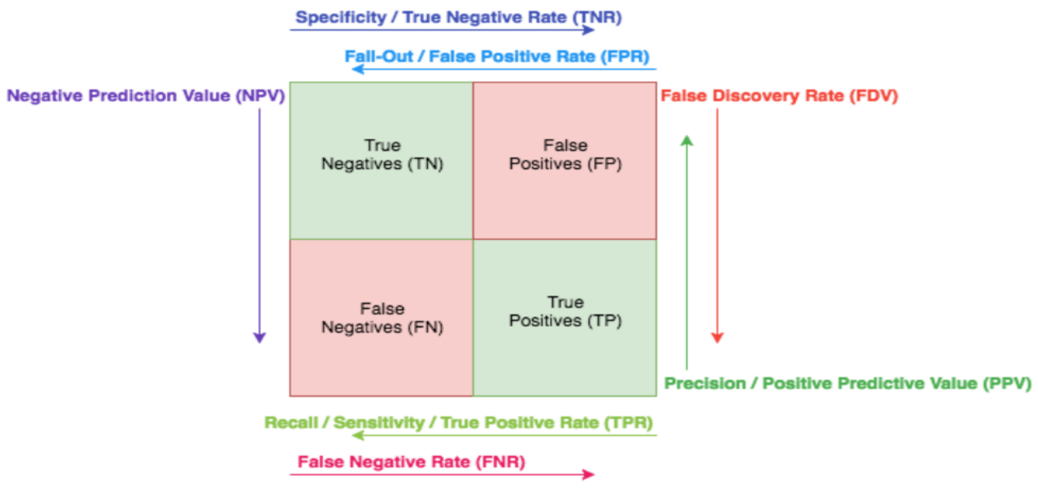
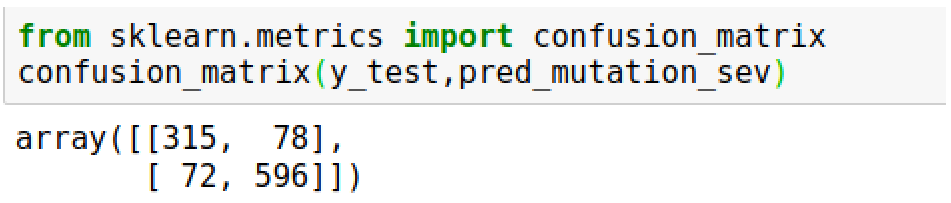

Specificity / True Negative Rate (TNR)
Fall-Out / False Positive Rate (FPR)
Negative Prediction Value (NPV)
False Discovery Rate (FDV)
True Negatives (TN)
False Positives (FP)
False Negatives (FN)
True Positives (TP)
Precision / Positive Predictive Value (PPV)
Recall / Sensitivity / True Positive Rate (TPR)
False Negative Rate (FNR)

Figure 20:This figure depicts a general layout of a confusion matrix (Evaluating a Classification Model, 2020)

Our confusion matrix has 596 true positives and 315 true negatives on the validation data. Hence, we can say that ratio of true positives to total true labels (596/911) is similar to our exploration analysis findings stating almost 60% severe mutations. From the confusion matrix, we can also evaluate other metrics such as precision and recall. Precision is the measure of how often the positive value is predicted correctly. It is calculated as a ratio of true positives to total positives (sum of true positives and false positives). We observe that our model has a precision of 0.88 which is very high. On the other hand, Recall is the measure of how effectively a classifier can identify positive labels (Brownlee, 2020). A recall is also known as Sensitivity. Sensitivity is computed as a ratio of the count of accurately classified positives instances to the total number of positive instances in the data (Brownlee, 2020). In terms of true positives and false negatives, it can be expressed as true positives divided by the sum of true positives and false negatives. Our binary classifier has a sensitivity of 0.89. This tells us that our model is highly sensitive which is a good evaluation parameter as sensitivity should be maximized. Another evaluation metric is specificity. Specificity measures the effectiveness of a classifier to identify negative labels. For our model, we have high specificity, which is expected for a good-fit model.



```
from sklearn.metrics import confusion_matrix
confusion_matrix(y_test,pred_mutation_sev)
```

```
array([[315,  78],
       [ 72, 596]])
```

Figure 21: This output gives an overview of the confusion matrix result

5.2.1.3 The Receiver Operating Curve (ROC) curve

In the case of a binary classifier, predicting the probability of an instance to categorize it into a positive class or negative class can be more flexible than assigning the data instance to a class directly (Brownlee, 2020). These probabilities can be interpreted in numerous ways depending upon the threshold value. The ROC curve can be one of the tools to interpret the probabilities of instances after prediction. The default threshold for classifying data points as per predicted probabilities is 0.5 which means if the probability is in the range of 0.0 to 0.49 then the corresponding data instance is mapped to negative class whereas probability greater than 0.5 and less than 1.0 then that instance is assigned to the positive class (Brownlee, 2020). We assign probabilities of mutations less than 0.5 classified to non-severe category and mutations having probabilities between range 0.5 to1.0 allocated to the severe class. While classifying a binary problem there could be two types of errors encountered. In terms of statistics they are known as the Type I error which are false positives and Type II error generally known as false negatives (Brownlee, 2020). So, choosing an optimal threshold value leads to the task of balancing the issues that arise due to these errors.

The ROC curve is a plot that has a false positive rate on the x-axis against the true positive rate marked on the y-axis for different threshold values. ROC curves prove to be useful to compare models. The overall shape of the curve provides information regarding the false positive rate and true positive rate (Brownlee, 2020). In general, models that are good-fit have a ROC curve resembling a bow extended up to top left position on the curve (Brownlee, 2020). On the other hand, an underperforming model can be observed as a diagonal line passing through the point (0.5,0.5) ie. bottom left to the top right position of the graph. Such models classify instances randomly (Brownlee, 2020). A perfect classifier will have line passing through the point (0,1) which is a line starting from bottom left position till topmost point on the y-axis and then traversing across x-axis on the top until the rightmost point on the curve (Brownlee, 2020). As we can observe in the figure below the ROC curve for our model depicts a bow-shaped arch which makes our model a good fit.

```
from sklearn import metrics
import matplotlib.pyplot as plt

fpr, tpr, thresholds = metrics.roc_curve(y_test, pred_mutation_sev_prob)
plt.plot(fpr, tpr)
```

```
[<matplotlib.lines.Line2D at 0x7f461cd19e10>]
```

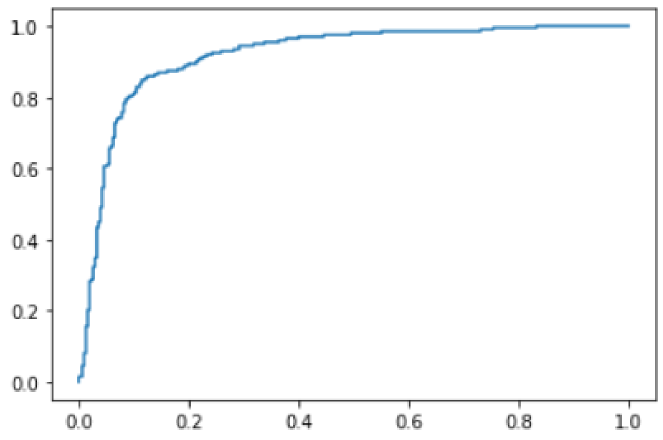


Figure 22:This ROC curve depicts the performance of our model.

5.2.2 Validation of the model on test data

After applying our dense neural network on test data which is a sample of mutations of hemophilia A population in the US. We predict the reported severity of mutations and obtain probabilities of two classes 0 and 1 stating non-severe and severe mutations respectively. We have considered the probability score of class 1 to transform into corresponding class labels. These probabilities we converted into class labels by enforcing a threshold value of 0.5. If the probability was greater than 0.5 then we labeled the instance as severe (1) else it was predicted as a non-severe mutation.

6. EVALUATION

We will evaluate our research questions based on our results and analysis.

1. Can we find patterns in the data to generate insights that can aid medical professionals/researchers to investigate more into discovering a novel approach towards hemophilia A treatment?

   We have uncovered compelling patterns and trends from our DNA mutations data. As per our analysis, we could observe that majority of the mutations occurring in the F8 gene are reported as severe mutations. Nearly fifty percent of severe mutations are caused due to the substitution of nucleotide base pairs. This substitution generally occurs at a nucleotide position between 0 to 2011 and 4925 to 7035. Also, the most common hotspots for base-pair substitutional mutations are observed when G is substituted with A, G changes to T, or C-to-T. Whereas most often when A is replaced with C it does not cause a severe mutation. By examining these patterns researchers working into the genomics domain can streamline their research more inclined towards severe mutations caused due to substitutional base pair change in hemophilia A. Specific mutation analysis work carried out in this research on severe mutations of hemophilia A can act as a strategic tool for preimplantation genetic diagnosis (PGD) of hemophilia A. PGD is a procedure before implantation which aids to detect any genetic disorders at the embryonic stage. This proves to be beneficial by preventing the transfer of genetic disorder to the coming next generation/s.

2. Is it possible to develop a binary classification model with high accuracy to utilize the model into the healthcare sector for the genetic diagnosis of hereditary disorders?

   We have built a binary classifier using a dense neural network to predict the reported severity of mutation in the F8 gene. Our model has a high accuracy of 85% which is required to be achieved in healthcare applications. We validated our model to check its authenticity. We achieved outstanding accuracy on training as well as validation data. Also, we predicted class labels for unseen test data with our model and accomplished outstanding results.

7. CONCLUSION AND RECOMMENDATIONS

7.1 Conclusion

In our study, we tried to recognize patterns from mutations in DNA of hemophilia A and predict whether it is a severe mutation, by using data science technologies.

Initially, in our research, we understand the molecular biological concepts of DNA, RNA, and the central dogma elaborating on the conversion of DNA to proteins. As we are interested in knowing the mutations in DNA, we overview the types of genetic mutations and causes leading to these mutations. We went through the literature work related to hemophilia A, its linkage with inheritance, and the advanced genetic diagnostic techniques.

Later, our study focused on examining various data science technologies helpful in analyzing the mutation data. As we had to develop a model to predict the overall mutation severity, we investigated various machine learning algorithms and their benefits depending on the use case. We explored various research papers on the classification of mutations in brain tumors using deep learning and artificial intelligence, leveraging multiple deep learning technologies in health informatics, and detecting the pre-cancer stage in TP53 gene mutation using convolutional neural networks.

After exploring prior research work carried out in the healthcare domain through the integration of data science and machine learning with bioinformatic tools, our research progressed by generating a framework to achieve a solution to our research problems. So, following the path of research design, we began with the data collection process by gathering data and diving into data to be familiar with data preprocessing tasks. Next, we performed exploratory data analysis and observed interesting patterns in the data. Then after preparing data which could be suitable for machine learning tasks we built a binary classifier using a dense neural network that could predict whether the novel mutations in F8 gene are severe or non-severe taking into consideration factors such as changes in DNA base pairs, changes in protein, type of mutation, the mechanism which causes a change in DNA, exon at the mutated location, codon at the mutated location, and year when the de novo mutation was first reported. We didn't take into consideration the Factor VIII concentration levels in plasma while predicting the overall reported severity as depending upon the F8 levels our model may learn well and outperform on the known data but would fail to perform as expected on the unknown data where F8 levels may be absent. We validated our model to check its accuracy as a highly accurate model is required for medical diagnostic tests analysis.

To conclude with our research work, we evaluated our research questions and with unbiased results, we can state that we came across compelling patterns discovered through data exploratory analysis. These patterns in mutation data may keen interest of genomic researchers to conduct analysis pertaining to severe mutations in hemophilia A arising due to substitutional base pair change. As discussed during the evaluation section, our data analysis findings and predictive model can be used for preimplantation genetic diagnosis (PGD) of hemophilia A. PCR technique used in PGD is quite labour extensive and time consuming to analyze DNA in cells. So, we can utilize our model and insights for analyzing the DNA of the suspect for any mutation in the gene. For hemophilia A disorder, we can evaluate the F8 gene of the mutant carrier. By inspecting the common patterns generated from our analysis we could assess if any similar mutation/s have occurred in the F8 gene of the suspect. If a mutation is observed then we could predict the severity of the mutation with the help of our classifier.

7.2 Strengths and Limitations of the research

7.2.1 Strengths of Research

1. Use of embedding technique for data transformation

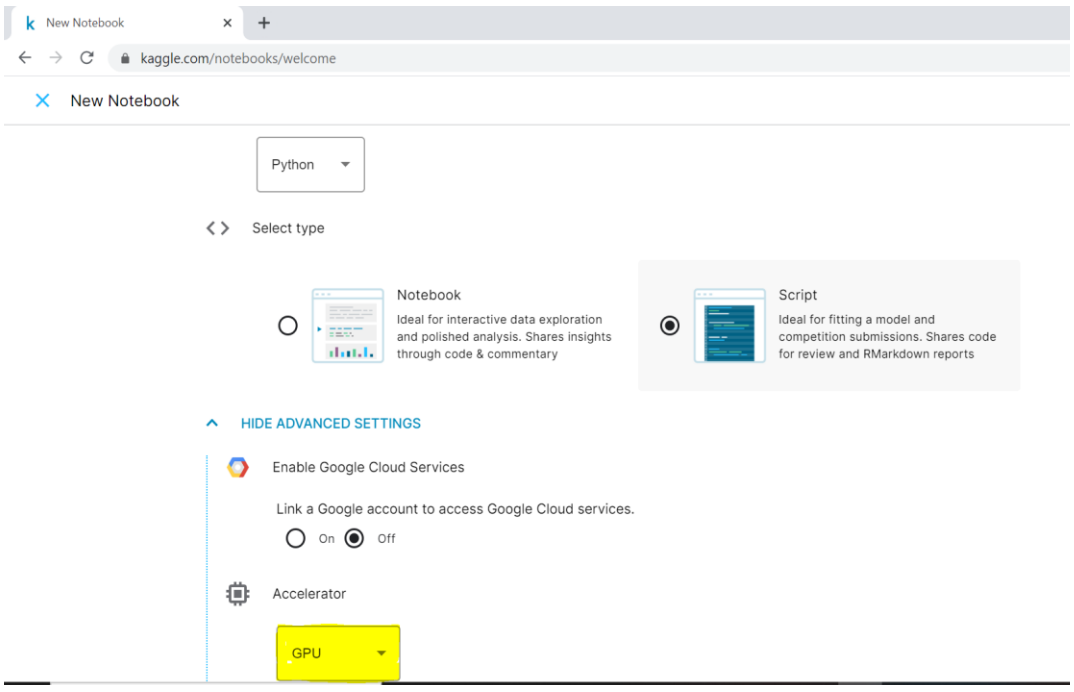
   While developing a dense neural network, our dataset has many categorical features that were required to be converted to numerical variables to be able to feed those features as input into the neural network. So, instead of implementing the one-hot encoding technique which could generate numerous features, we transform our data into low dimensional vectors using the embedding layer in the neural network. Using embedding layer number of features for analysis does not increase and ultimately curse of dimensionality can be reduced.

2. High accuracy model

   In this research, we constructed a binary classifier using a dense neural network from scratch. A binary classifier could be modeled using traditional machine learning algorithms such as logistic regression, decision trees, or support vector machines. But we required a model predicting accurately with high precision hence we chose deep learning as our approach to building a model. Also, feature extraction performed at the embedding layer increased the model accuracy.

3. Utilized Kaggle GPU system for compiling and deployment of the neural network:

   Once the model was constructed using Keras we compiled the model on Kaggle python notebook. This notebook is enabled with GPU for high computational performance so that neural networks can be easily compiled.


New Notebook

kaggle.com/notebooks/welcome

New Notebook

Python

Select type

( ) Notebook
Ideal for interactive data exploration and polished analysis. Shares insights through code & commentary

(●) Script
Ideal for fitting a model and competition submissions. Shares code for review and RMarkdown reports

HIDE ADVANCED SETTINGS

Enable Google Cloud Services

Link a Google account to access Google Cloud services.

On [✓ Off]

Accelerator

GPU

Figure 23:This diagram shows leveraging GPU using Kaggle notebook

.

7.2.2 Limitations of Research

1. Limited data:

We retrieved our data from the Centers for Disease Control and Prevention (CDC) database. The dataset consisted of a limited number of records as only novel mutations in hemophilia A were taken into account. If more data were available for training the classifier then we could have achieved more accuracy but instead of developing a model with higher accuracy, it is better to consider only authentic data for our analysis and research work.

7.3 Further Work and Recommendations

In this research, we have found quite interesting insights and built a predictive classifier that could be integrated with healthcare tools. As mentioned earlier we can integrate our model with genomic diagnostic tools. To avoid the risk of transferring hemophilia A to future generations if any one parent is a carrier of it, we can integrate our model with preimplantation genetic diagnosis technique. In this process, we can check for mutations in DNA that cause hemophilia A in pregnant women before childbirth. If we observe severe mutations at this stage then healthcare professionals can suggest for mutational repair accordingly. Thus, in the future, we can extend this research to integrate results from our research work with bioinformatic tools.

REFERENCES

Chang, P., Grinband, J., Weinberg, B., Bardis, M., Khy, M., Cadena, G., Su, M., Cha, S., Filippi, C., Bota, D., Baldi, P., Poisson, L., Jain, R. and Chow, D. (2018). Deep-Learning Convolutional Neural Networks Accurately Classify Genetic Mutations in Gliomas. American Journal of Neuroradiology, 39(7), pp.1201-1207

Davy, G., Rousselin, A., Goardon, N., Castéra, L., Harter, V., Legros, A., Muller, E., Fouillet, R., Brault, B., Smirnova, A., Lemoine, F., de la Grange, P., Guillaud-Bataille, M., Caux-Moncoutier, V., Houdayer, C., Bonnet, F., Blanc-Fournier, C., Gaildrat, P., Frebourg, T., Martins, A., Vaur, D. and Krieger, S. (2017). Detecting splicing patterns in genes involved in hereditary breast and ovarian cancer. European Journal of Human Genetics, 25(10), pp.1147-1154.

Sennaar, K. (2019). Machine Learning in Genomics – Current Efforts and Future Applications | Emerj. [online] Emerj. Available at: https://emerj.com/ai-sector-overviews/machine-learning-in-genomics-applications

Daniele, R., Wong, C., Deligianni, F., Berthelot, M., Andreu-Perez, J., Benny Lo, and Yang, G. (2017). Deep Learning for Health Informatics.

FierceBiotech. (2019). Making CRISPR-Cas9 gene editing safer with artificial intelligence. [online] Available at: https://www.fiercebiotech.com/research/using-machine-learning-to-predict-crispr-cas9-editing-outcomes

The Scientist Magazine®. (2019). Could AI Make Gene Editing More Accurate?. [online] Available at: https://www.the-scientist.com/the-literature/could-ai-make-gene-editing-more-accurate-65781

Chop.edu. (2019). Genome Editing for Hemophilia: A Next Step in Genetic Therapy | Children's Hospital of Philadelphia. [online] Available at: https://www.chop.edu/pages/genome-editing-hemophilia-next-step-genetic-therapy

van Overbeek, M., Capurso, D., Carter, M., Thompson, M., Frias, E., Russ, C., Reece-Hoyes, J., Nye, C., Gradia, S., Vidal, B., Zheng, J., Hoffman, G., Fuller, C. and May, A., 2016. DNA Repair Profiling Reveals Nonrandom Outcomes at Cas9-Mediated Breaks. Molecular Cell, 63(4), pp.633-646.

Genetics Home Reference. 2020. What Are Genome Editing And CRISPR-Cas9?. [online]Available at: https://ghr.nlm.nih.gov/primer/genomicresearch/genomeediting [Accessed 25 March 2020].

Bhattacharjee, S. and Nandi, S., 2018. Rare Genetic Diseases with Defects in DNA Repair: Opportunities and Challenges in an Orphan Drug Development for Targeted Cancer Therapy. Cancers, 10(9), p.298.

Shouval, R., Bondi, O., Mishan, H., Shimoni, A., Unger, R. and Nagler, A., 2013. Application of machine learning algorithms for clinical predictive modeling: a data-mining approach in SCT. Bone Marrow Transplantation, 49(3), pp.332-337.

International Journal of Science and Research (IJSR), 2016. Comparative Study of K-NN, Naive Bayes and Decision Tree Classification Techniques. 5(1), pp.1842-1845.

Géron, A., n.d. Hands-On Machine Learning with Scikit-Learn, Keras, And Tensorflow.

Lindow, N., Baum, D., Leborgne, M. and Hege, H., 2019. Interactive Visualization of RNA and DNA Structures. IEEE Transactions on Visualization and Computer Graphics, 25(1), pp.967-976.

A Framework for Codon Based Analysis to detect abnormalities responsible for Esophagus Cancer using Soft Computing Tool. 2018 5th International Conference on Signal Processing and Integrated Networks (SPIN)

Modeling and Analysis of DNA Mutation Type Based on Colored Petri Net. Proceedings of the 2017 IEEE International Conference on Information and Automation (ICIA)

Dekhtyar, A., Fong, V., 2017
RE Data Challenge: Requirements Identification with Word2Vec and TensorFlow. 2017 IEEE 25th International Requirements Engineering Conference

Lopez, W., Merlino, J., Rodriguez, P., 2017
Vector representation of Internet Domain Names using a Word Embedding technique.

Loew, L., Spindler, M., Brechmann, E., 2018
A Self-Attention Network for Hierarchical Data Structures with an Application to Claims Management.

Sokolova, M. and Lapalme, G., 2009. A systematic analysis of performance measures for classification tasks. Information Processing & Management, 45(4), pp.427-437.

Omim.org. 2020. OMIM Entry - # 306700 - HEMOPHILIA A; HEMA. [online] Available at:
https://omim.org/entry/306700 [Accessed 27 July 2020].

National Hemophilia Foundation. 2020. Fast Facts. [online] Available at: https://www.hemophilia.org/About-Us/Fast-Facts [Accessed 09 August 2020].

Varnomen.hgvs.org. 2020. Sequence Variant Nomenclature. [online] Available at:
http://varnomen.hgvs.org/history/ [Accessed 3 August 2020].

Khan Academy. 2020. Central Dogma (DNA To RNA To Protein) | Biology Library | Khan Academy. [online] Available at:
https://www.khanacademy.org/science/biology/gene-expression-central-dogma/central-dogma-transcription/ [Accessed 3 August 2020].

Centers for Disease Control and Prevention. 2020. Home | Hemophilia | NCBDDD | CDC. [online] Available at:
https://www.cdc.gov/ncbddd/hemophilia/ [Accessed 24 July 2020].

TensorFlow. 2020. Word Embeddings | Tensorflow Core. [online] Available at: https://www.tensorflow.org/tutorials/text/word_embeddings [Accessed 3 August 2020].

ritchieng.github.io. 2020. Evaluating A Classification Model. [online] Available at:
https://www.ritchieng.com/machine-learning-evaluate-classification-model/
[Accessed 6 August 2020].

Konkle, B., Huston, H. and Fletcher, S., 2020. Hemophilia A. [online] Ncbi.nlm.nih.gov. Available at:
https://www.ncbi.nlm.nih.gov/books/NBK1404/ [Accessed 6 August 2020].

Brownlee, J., 2020. How To Use ROC Curves And Precision-Recall Curves For Classification In Python. [online] Machine Learning Mastery. Available at:
https://machinelearningmastery.com/roc-curves-and-precision-recall-curves-for-classification-in-python/ [Accessed 9 August 2020].

Prof. Dave, (2016). Mechanisms of DNA Damage and Repair. Available at:
https://www.youtube.com/watch?v=sX6LncNjTFU [Accessed 24 July 2020]

Hgvs.org. 2020. HGVS Recommendations: General, DNA Level. [online] Available at: https://www.hgvs.org/mutnomen/recs-DNA.html#:~:text=description%20of%20nucleotides%20at%20DNA,)%2C%20(see%20Standards) [Accessed 9 August 2020].

Mun.ca. 2020. Exons, Introns & Codons. [online] Available at: https://www.mun.ca/biology/scarr/Exons_Introns_Codons.html [Accessed 5 August 2020].

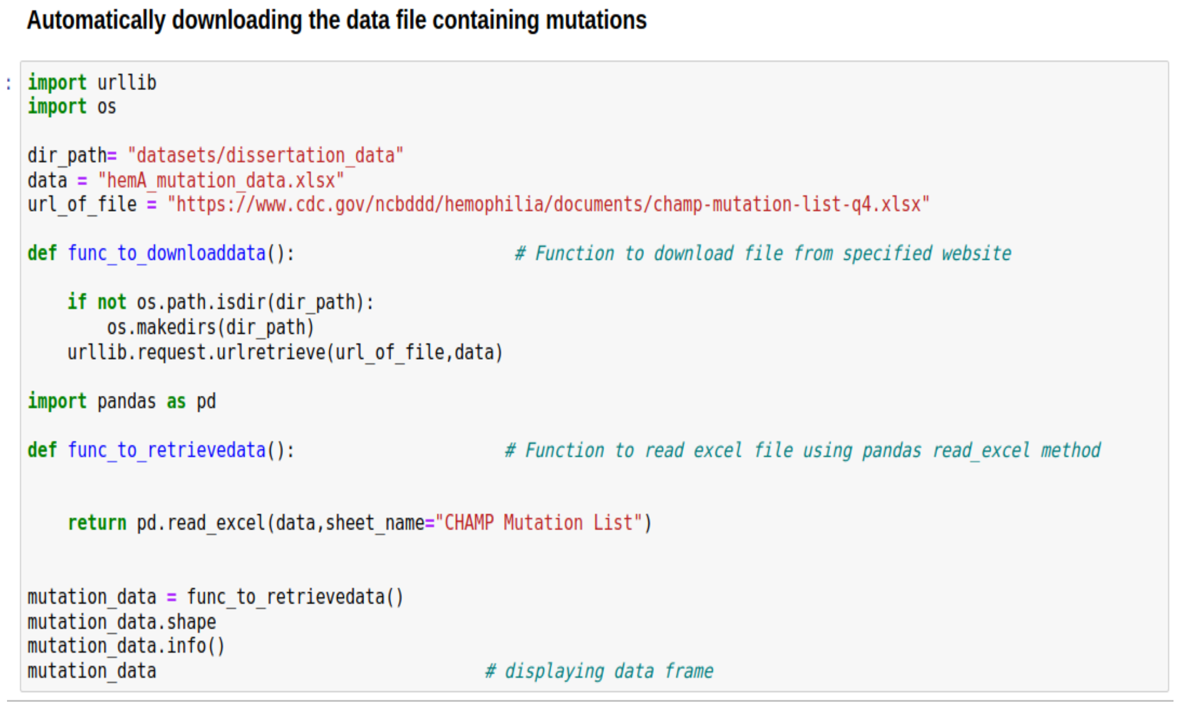
Altair-viz.github.io. 2020. Altair: Declarative Visualization In Python — Altair 4.1.0 Documentation. [online] Available at: https://altair-viz.github.io/ [Accessed 3 August 2020].

LeCun, Y., Bengio, Y. and Hinton, G., 2015. Deep learning. Nature, 521(7553), pp.436-444.

Sermon, K., Van Steirteghem, A. and Liebaers, I., 2004. Preimplantation genetic diagnosis. The Lancet, 363(9421), pp.1633-1641.

APPENDICES

APPENDIX A: CODE SNAPSHOTS



Automatically downloading the data file containing mutations

```
import urllib
import os

dir_path= "datasets/dissertation_data"
data = "hemA_mutation_data.xlsx"
url_of_file = "https://www.cdc.gov/ncbddd/hemophilia/documents/champ-mutation-list-q4.xlsx"

def func_to_downloaddata():                          # Function to download file from specified website

    if not os.path.isdir(dir_path):
        os.makedirs(dir_path)
    urllib.request.urlretrieve(url_of_file,data)

import pandas as pd

def func_to_retrievedata():                          # Function to read excel file using pandas read_excel method


    return pd.read_excel(data,sheet_name="CHAMP Mutation List")


mutation_data = func_to_retrievedata()
mutation_data.shape
mutation_data.info()
mutation_data                                        # displaying data frame
```

Data profiling using pandas profiling

```python
from pandas_profiling import ProfileReport          # importing pandas_profiling

report = ProfileReport(mutation_data,minimal=True)
report.to_file(output_file= 'output.html')
report
```

Data cleaning

```python
remove_notargetclass =  mutation_data[mutation_data['Reported \nSeverity']=='Not reported']
mutation_data = mutation_data.drop(remove_notargetclass.index,axis=0)       # removing rows with incorrect target
                                                                            # values

remove_notargetclass_space =  mutation_data[mutation_data['Reported \nSeverity']=='Not reported ']
mutation_data = mutation_data.drop(remove_notargetclass_space.index,axis=0)

remove_notargetclass_cap =  mutation_data[mutation_data['Reported \nSeverity']=='Not Reported']
mutation_data = mutation_data.drop(remove_notargetclass_cap.index,axis=0)

remove_notargetclass_lower =  mutation_data[mutation_data['Reported \nSeverity']=='not reported']
mutation_data = mutation_data.drop(remove_notargetclass_lower.index,axis=0)


mutation_data
```

```python
mutation_data['Reported \nSeverity'] = mutation_data['Reported \nSeverity']   # removing inconsistencies in data
.map({'Mild':'Mild','Moderate':'Moderate','Mild/Moderate':'Mild/Moderate',
      'Mild/Moderate/Severe':'Mild/Moderate/Severe','Mild/Severe':'Mild/Severe',
      'Moderate/Severe':'Moderate/Severe','Severe':'Severe','severe':'Severe','SEVERE':'Severe'})

mutation_data
```

```python
mutation_data['Reported \nSeverity'] = mutation_data['Reported \nSeverity'].astype('category')
mutation_data
```

```python
remove_notreported = mutation_data[mutation_data['HGVS cDNA']== 'Not reported']

mutation_data = mutation_data.drop(remove_notreported.index,axis=0)
mutation_data
```

Graph displaying number of mutations as per reported severity

```
reported_severity_data = mutation_data.groupby(['Reported \nSeverity'])['HGVS cDNA'].count().reset_index()
reported_severity_data
```

```
import altair as alt
alt.data_transformers.disable_max_rows()

#bar chart depicting number of mutations as per overall reported severity

Plot_Reported_severity = alt.Chart(reported_severity_data).mark_bar().encode(
    x=alt.X(
        "HGVS cDNA",
        title="Number of mutations",
    ),
    y=alt.Y(
        "Reported \nSeverity",
        title="Severity of Mutations as published",
    ),
    color="Reported \nSeverity",

)

text = Plot_Reported_severity.mark_text(

      align='left',
      baseline='middle',
      dx=3
).encode(
    text='HGVS cDNA'

)

(Plot_Reported_severity + text).properties(height=300)
```

```
mutation_data_yearwise_mutations = mutation_data.groupby(['Year\nReported'])['HGVS cDNA'].count().reset_index()
mutation_data_yearwise_mutations
```

```
mutation_data_yearwise_mutations['Year\nReported'] = mutation_data_yearwise_mutations['Year\nReported']
.astype('category')
```

```

# Graph depicting number of novel mutations observed from 1985 - 2019

Plot_yearwise_mutations = alt.Chart(mutation_data_yearwise_mutations).mark_line(color='#FFAA00').encode(
    x=alt.X(
        "Year\nReported",
        title="Time frame in years (1985 - 2019)",
    ),
    y=alt.Y(
        "HGVS cDNA",
        title="Number of mutations of hemophilia A discovered ",
    ),

)
text = Plot_yearwise_mutations.mark_text(

      align='left',
      baseline='middle',
      dx=3
).encode(
    text='HGVS cDNA'

)
```

```python
Reported_severity_severe = mutation_data[mutation_data['Reported \nSeverity'] == 'Severe']
Reported_severity_severe
```

```python
Reported_severity_severe_subs = Reported_severity_severe[Reported_severity_severe['Mechanism'] == 'Substitution']
Reported_severity_severe_subs
```

```python
Reported_severity_severe_subs['cDNA'] = Reported_severity_severe_subs['HGVS cDNA'].str.split(".",expand=True)[1]
Reported_severity_severe_subs
```

```python
Reported_severity_severe_subs['Nucleotide_position'] = Reported_severity_severe_subs['cDNA']
.str.split(">",expand=True)[0]                    # splitting the cDNA

Reported_severity_severe_subs
```

```python
Reported_severity_severe_subs['Nucleotide_position']= Reported_severity_severe_subs['Nucleotide_position']
.map(lambda x: str(x)[:-1])                       # extracting nucleotide position value from cDNA

Reported_severity_severe_subs
```

```python
Reported_severity_severe_subs['Nucleotide_position']= Reported_severity_severe_subs['Nucleotide_position']
.str.split("+",expand=True)[0].replace({None:Reported_severity_severe_subs['Nucleotide_position']})

Reported_severity_severe_subs
```

```python
Reported_severity_severe_subs['Nucleotide_position']= Reported_severity_severe_subs['Nucleotide_position']
.str.split("-",expand=True)[0].replace({None:Reported_severity_severe_subs['Nucleotide_position']})

Reported_severity_severe_subs
```

```python
invalid_year_value = Reported_severity_severe_subs[Reported_severity_severe_subs['Nucleotide_position']==""]
Reported_severity_severe_subs = Reported_severity_severe_subs.drop(invalid_year_value.index,axis=0)
```

Graph depicting nucleotide positions and number of severe substitutional mutations in range of position

```python
groupby_nucl_pos['Nucleotide_Pos_category'] = [x for x in ['Range 0 to 704','Range 704 to 1408',
                                                            'Range 1408 to 2111','Range 2111 to 2815',
                                                            'Range 2815 to 3518','Range 3518 to 4221',
                                                            'Range 4221 to 4925','Range 4925 to 5628',
                                                            'Range 5628 to 6332','Range 6332 to 7035']]
groupby_nucl_pos
```

```python
groupby_nucl_pos = groupby_nucl_pos.drop('Nucleotide_pos_range',axis=1)
groupby_nucl_pos
```

```python
groupby_nucl_pos['Nucleotide_Pos_category'] = groupby_nucl_pos['Nucleotide_Pos_category'].astype('category')
```

```python
plot_nucleotide_pos = alt.Chart(groupby_nucl_pos).mark_bar().encode(
    y='Nucleotide_Pos_category',
    color='Nucleotide_Pos_category',
    x=alt.X(
        'HGVS cDNA',
        title= 'Number of substitutional mutations'
    ),
    tooltip ='HGVS cDNA'

).interactive()
```

Data Cleaning

```
remove_notreported = mutation_data[mutation_data['HGVS cDNA']== 'Not reported']

mutation_data = mutation_data.drop(remove_notreported.index,axis=0)

remove_notargetclass =  mutation_data[mutation_data['Reported \nSeverity']=='Not reported']
mutation_data = mutation_data.drop(remove_notargetclass.index,axis=0)
mutation_data
```

```
mutation_data['In Poly A'] = mutation_data['In Poly A'].map({'Y':1,'N':0})
mutation_data
```

```
mutation_data['Severe \n(<1 U/dL)'] = mutation_data['Severe \n(<1 U/dL)'].map({'X':1})
mutation_data['Moderate \n(1-5 U/dL)'] = mutation_data['Moderate \n(1-5 U/dL)'].map({'X':1})
mutation_data['Mild \n(>5 U/dL)'] = mutation_data['Mild \n(>5 U/dL)'].map({'X':1})


severe_fill = {'Severe \n(<1 U/dL)':0}
moderate_fill = {'Moderate \n(1-5 U/dL)':0}
mild_fill = {'Mild \n(>5 U/dL)':0}

mutation_data = mutation_data.fillna(value=severe_fill)
mutation_data = mutation_data.fillna(value=moderate_fill)
mutation_data = mutation_data.fillna(value=mild_fill)

mutation_data
```

```
mutation_data['Mutation Severity'] = mutation_data['Reported \nSeverity'].map({'Mild':'Not Severe','Moderate':'Not
mutation_data
```

```
mutation_data = mutation_data.rename(columns={'HGVS cDNA':'cDNA','HGVS Protein':'Protein',
                                              'Mature Protein':'MatureProtein','Mutation Type':'MutationType',
                                              'In Poly A':'InPolyA','Severe \n(<1 U/dL)':'SevereIndicator',
                                              'Moderate \n(1-5 U/dL)':'ModerateIndicator',
                                              'Mild \n(>5 U/dL)':'MildIndicator','Reported \nSeverity':'ReportedSev
                                              'History of Inhibitor':'HistoryOfInhibitor','Year\nReported':'YearRep
mutation_data
```

Including selective columns as features for neural network

```
from sklearn import preprocessing

features = [col for col in mutation_data.columns if col not in ['hg19 Coordinates','Unclassified\n(no FVIII level)'
 'ModerateIndicator',
 'MildIndicator','Mutation Severity','ReportedSeverity']]
```

```
features
```

Performing label encoding on all selective features

```
for each_feature in features:
    label_encoder = preprocessing.LabelEncoder()
    mutation_data.loc[:,each_feature] = label_encoder.fit_transform(mutation_data[each_feature].astype(str)
                                                                    .fillna("-1").values)
```

```
mutation_data.head()
```

```
mutation_data['Mutation Severity'] = mutation_data['Mutation Severity'].map({'Not Severe':int(0),'Severe':int(1)})
mutation_data.head()
```

```
mutation_data['Mutation Severity'].fillna("0",inplace=True)
mutation_data.head()
```

```
mutation_data['Mutation Severity'] = mutation_data['Mutation Severity'].astype(int)
mutation_data['Mutation Severity'].head()
```

```
mutation_data
```

Splitting data into training and validation data

```
from sklearn.model_selection import train_test_split
X = mutation_data[mutation_data.columns.difference(['Mutation Severity'])]
y = mutation_data['Mutation Severity'].values.reshape(-1,1)


X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=0)
```

```
import tensorflow as tf
from keras import layers
from keras import optimizers
from keras.models import Model
import numpy as np
from keras.constraints import max_norm
```

Function to develop dense neural network including embedding layer

```
def create_neural_network(dataframe,categorical_columns):
    input_list = []
    output_list = []
    for each_feature in categorical_columns:
        unique_values= int(dataframe[each_feature].nunique())
        embed_dimension =int(min(np.ceil(unique_values/2),50))
        input_layer = layers.Input(shape=(1,))
        output = layers.Embedding(unique_values+1,embed_dimension,name=each_feature)(input_layer)
        output = layers.Reshape(target_shape=(embed_dimension,))(output)
        input_list.append(input_layer)
        output_list.append(output)
    single_vector_output = layers.Concatenate()(output_list)
    single_vector_output = layers.Dense(32,activation='relu',kernel_constraint=max_norm(1.0))(single_vector_output)

    y = layers.Dense(2,activation='sigmoid')(single_vector_output)
    neural_netwrk_model = Model(inputs=input_list,outputs=y)
    return neural_netwrk_model
```

```
create_neural_network(X,features).summary()
```

Compiling model

```
model_train = create_neural_network(X,features)
model_train.compile(loss='binary_crossentropy', metrics=['accuracy'])
```

Model fitting on training and valiadtion data

```
from tensorflow.keras import utils
X_train = [X_train.loc[:, features].values[:, k] for k in range(X_train.loc[:, features].values.shape[1])]
X_test = [X_test.loc[:, features].values[:, k] for k in range(X_test.loc[:, features].values.shape[1])]

history = model_train.fit(X_train,utils.to_categorical(y_train)
         , validation_data = (X_test, utils.to_categorical(y_test)),verbose =1, epochs = 20)
```

```
_, train_acc = model_train.evaluate(X_train,utils.to_categorical(y_train), verbose=0)
_, val_acc = model_train.evaluate(X_test,utils.to_categorical(y_test), verbose=0)
```

```
print('Train: %.3f, Test: %.3f' % (train_acc, val_acc))
```

```
from matplotlib import pyplot

pyplot.plot(history.history['accuracy'],label = 'train')
pyplot.plot(history.history['val_accuracy'],label= 'validation')
pyplot.legend()
pyplot.show()
```

Prediting the class labels by converting probabilities into classes

```
y_pred = model_train.predict(X_test)
list(y_pred)
y_pred = y_pred.tolist()
(y_pred)
```

```

pred_mutation_sev = [pred_severity for y_list in y_pred for pred_severity in y_list]
(pred_mutation_sev)
pred_mutation_sev_prob = pred_mutation_sev[1::2]
pred_mutation_sev = [round(each_pred) for each_pred in pred_mutation_sev_prob]
```

```
(pred_mutation_sev)
```

Model Evaluation : Confusion matrix

```
from sklearn.metrics import confusion_matrix
confusion_matrix(y_test,pred_mutation_sev)
```

Model Evaluation : Recall

```
from sklearn.metrics import recall_score
recall_score(y_test,pred_mutation_sev, average=None)
```

Model Evaluation : Precision

```
from sklearn.metrics import precision_score
precision_score(y_test, pred_mutation_sev, average=None)
```

```
from sklearn import metrics
import matplotlib.pyplot as plt

fpr, tpr, thresholds = metrics.roc_curve(y_test, pred_mutation_sev_prob)
plt.plot(fpr, tpr)
```

Evaluation of model on test data (US hemophilia A mutation data)

```
import pandas as pd
mutation_data_US = pd.read_excel('./Data/champ-mutations-us.xlsx',sheet_name="CHAMP US Database")

mutation_data_US.shape
mutation_data_US.info()
mutation_data_US
```

```
mutation_data_US = mutation_data_US.drop(['RACE','Ethnicity','Country of Origin','Country of Report',
                                          'Source','Source ID'],axis=1)
mutation_data_US
```

```
mutation_data_US = mutation_data_US.rename(columns={'HGV cDNA Name':'cDNA','HGV Protein Name':'Protein',
                                                    'Mutation Type':'MutationType',
                                                    'Reported Severity':'ReportedSeverity',
                                                    'History of Inhibitor':'HistoryOfInhibitor',
                                                    'Nucleotide Change':'NucleotideChange',
                                                    'Predicted AA Change':'PredictedAAChange',
                                                    'Severity from Factor Level':'SeverityFactorLevel'})
mutation_data_US
```

```
from sklearn import preprocessing

features_US = [col for col in mutation_data_US.columns if col not in ['ReportedSeverity','FVIII:C','FVIII:Ag']]
```

```
for each_feature in features_US:
    label_encoder = preprocessing.LabelEncoder()
    mutation_data_US.loc[:,each_feature] = label_encoder.fit_transform(mutation_data_US[each_feature]
                                                                       .astype(str).fillna("-1").values)
```

```
mutation_data_US.head()
```

```
model_US = create_neural_network(mutation_data_US,features_US)
model_US.compile(loss='binary_crossentropy',metrics=['accuracy'])
```

```
y_pred_eval = model_US.predict([mutation_data_US.loc[:,f] for f in features_US])
y_pred_eval
```

```
list(y_pred_eval)
y_pred_eval = y_pred_eval.tolist()
(y_pred_eval)
```

APPENDIX B : TURNITIN SIMILARITY REPORT AND DIGITAL RECEIPT

Pranav Pramod Deogaonkar | thesis

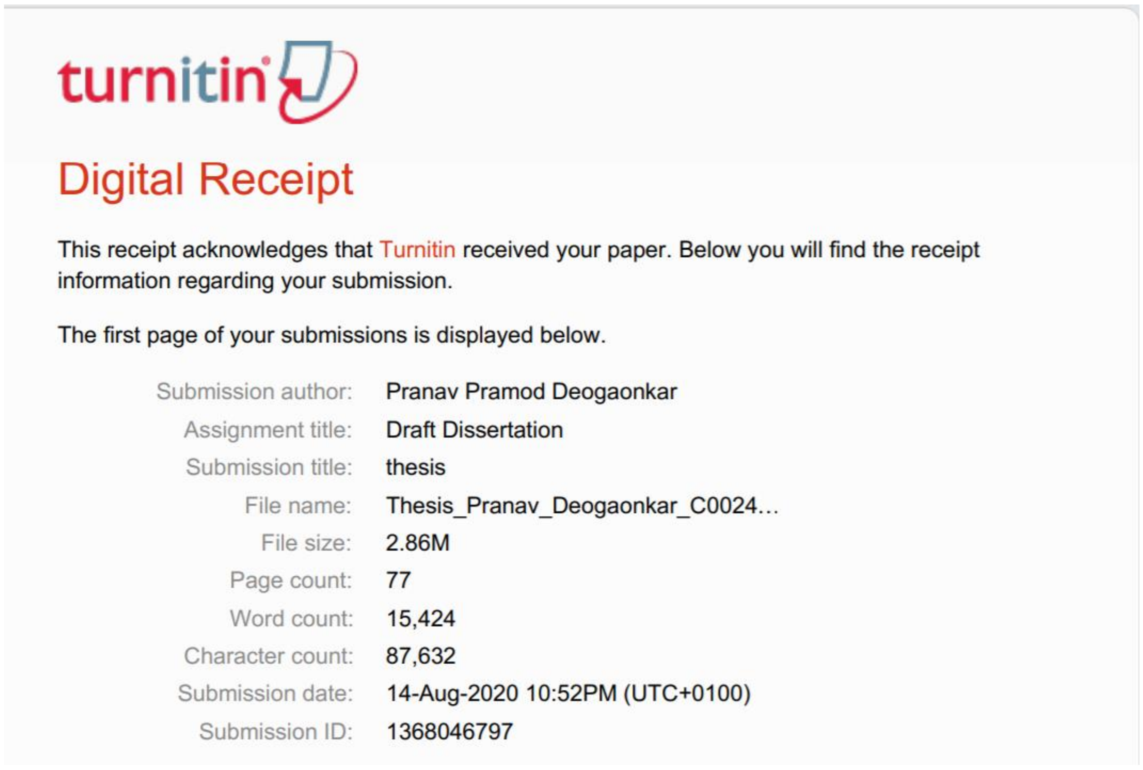
APPLYING DATA SCIENCE TECHNIQUES

TO UNDERSTAND DNA MUTATIONS IN HEMOPHILIA- A

Institiúid Teicneolaíochta Cheatharlach

INSTITUTE of TECHNOLOGY CARLOW

At the Heart of South Leinster



turnitin

Digital Receipt

This receipt acknowledges that Turnitin received your paper. Below you will find the receipt information regarding your submission.

The first page of your submissions is displayed below.

Submission author:	Pranav Pramod Deogaonkar
Assignment title:	Draft Dissertation
Submission title:	thesis
File name:	Thesis_Pranav_Deogaonkar_C0024...
File size:	2.86M
Page count:	77
Word count:	15,424
Character count:	87,632
Submission date:	14-Aug-2020 10:52PM (UTC+0100)
Submission ID:	1368046797

LIFELONG LEARNING CENTRE

Work submitted for assessment which does not include this declaration will not be assessed.

DECLARATION

*I declare that all material in this submission e.g. thesis/essay/project/assignment is entirely my/our own work except where duly acknowledged.

*I have cited the sources of all quotations, paraphrases, summaries of information, tables, diagrams, or other material; including software and other electronic media in which intellectual property rights may reside.

*I have provided a complete bibliography of all works and sources used in the preparation of this submission.

*I understand that failure to comply with the Institute's regulations governing plagiarism constitutes a serious offence.

Student Name: (Printed)	PRANAV DEOGAONKAR
Student Number(s):	C00246393
Programme Title & Yr:	M.Sc. Data Science, 2019-2020
Module :	Dissertation
Signature(s):	PRANAV PRAMOD DEOGAONKAR
Date:	17.08.2020

Please note:

a) Individual declaration is required by each student for joint projects.

b) Where projects are submitted electronically, students are required to type their name under the signature.

c) The Institute regulations on plagiarism are set out in Section 10 of Examination and Assessment Regulations published each year in the Student Handbook