University of Piraeus
Department of Digital Systems

Postgraduate Program “Information Systems & Services”
Big Data and Analytics

Masters diploma thesis

Computational Methods for Improving Genetic Therapies for Duchenne Muscular Dystrophy

Georgios Ch. Kargas

Athens 2021

Supervisor: Associate Professor Michael Filippakis
Department of Digital Systems, University of Piraeus
Supervisor: Dr. Georgios Paliouras
Institute of Informatics and Telecommunications, National Center for Science Research “Demokritos”
Supervisor: Dr. Anastasia Krithara
Institute of Informatics and Telecommunications, National Center for Science Research “Demokritos”

Acknowledgments

First of all, I would like to thank the supervisor of my thesis Professor Michael Filippakis, who accepted to be my supervisor, for his valuable help and guidance during my work. I am also grateful to Professor Georgios Paliouras, Mrs. Anastasia Krithara and Mr. Anastasios Nentidis for carefully reading my work throughout this writing, for their suggestions, as well as for their valuable assistance and cooperation. Above all, I am grateful to my parents, Christos and Paraskevi Karga and for their wholehearted love and support over the years. I devote this work to them.

Georgios Ch. Kargas

Abstract

Genetic engineering involves different techniques to intentionally modify genetic material (primarily deoxyribonucleic acid or DNA) in order to alter, restore, or boost shape or function. Established in the latter half of the twentieth century, recombinant DNA technologies include chemical splicing (recombination) of different strands of DNA, usually using either bacteria (such as Escherichia coli) or bacteriophages (viruses that infect bacteria, such as λ phage) or bacteria (viruses that infect bacteria, such as λ phage) or by way of simple microinjection. In recent years, modern techniques to design and create, literally to engineer, new life forms, typically referred to as synthetic biology, have supplemented these conventional instruments.[1]

A flexible and efficient gene therapy technique is precision genome editing. The area has been subject to continuing developments since the introduction of CRISPR/Cas systems for genome editing. The new biotechnology includes the development of a site-specific DSB accompanied by two key forms of repair mechanisms: end-joining non-homologous and repair-directed homology. The enabled type of the mechanism for molecular repair relies on the cycle of the chromosome, cellular heterogeneity and cell division.[2]

In the current diploma thesis, we tried to break down and understand the main components of prime editing technique. An automated approach is required for advancing our understanding of the evolution and diversity of prime editing and for finding new candidates for genome engineering. We used Duchenne Muscular Dystrophy as a use case for this proposed approach. More specifically, we applied the knowledge of machine learning over prime editing in the genetic disease of Duchenne Muscular Dystrophy.

Table of Contents

Figures

Tables

Abbreviations

DMD	Duchenne Muscular Dystrophy
BMD	Becker's Muscular Dystrophy
gRNA	guide Ribonucleic Acid
tracrRNA	trans-activating crRNA
pegRNA	prime editing guide Ribonucleic Acid
MD	Muscular Dystrophy
CRISPR	Clustered Regularly Interspaced Short Palindromic Repeats
PAM	Protospacer Adjacent Motif
NHEJ	Non-homologous end joining
HDR	Homology directed repair
Nick	A break in only one strand of DNA
Nickase	Enzyme that is capable of cleaving only one strand of target DNA.
Coding DNA strand	The double-stranded chain of the DNA of the gene used as a template for the RNA synthesis.
Non coding DNA strand	The DNA sequence of the gene that has a sequence similar to the RNA synthesized from it.
Bp	Base pair, unit consisting of two nucleobases bound to each other by hydrogen bonds

1 Introduction

The ability to make site-specific alterations to the human genome has been a goal of medicine since the gene was established as the fundamental unit of heredity. Thus, gene therapy is understood as the ability to change biology by the reversal of modified (mutated genes or site-specific changes that target clinical treatment. This treatment was made possible by the advances in genetics and bioengineering that made it possible to modify vectors for the transmission of extrachromosomal material to target cells. One of the key interests of this strategy is the optimization of delivery vehicles (vectors) which are often plasmids, nanostructured or viruses. Viruses are most frequently studied due to their excellence of infecting cells and the inclusion of their genetic material. However, there is great concern about exacerbated immune responses and modification of the genome, especially in germ line cells. In vivo somatic cell experiments have shown satisfactory outcomes with accepted protocols in clinical trials.[3]

This innovation in genetic methodologies lead to the development of vast quantities of data that require the help of mathematical and analytical tools to be properly analysed. Thus, an automated approach is therefore needed to advance our knowledge of the evolution and diversity of these methodologies and to identify new candidates for genome engineering in eukaryotic models. [4] [5]

Duchenne muscular dystrophy (DMD), caused by mutations in the dystrophin gene, is characterized by cardiovascular and skeletal muscle degeneration, lack of ambulation and premature death. Dystrophin is a huge protein (>3600 amino acids) that stabilizes muscle membranes by binding the actin cytoskeleton to the inner surface of the sarcolemma. Thousands of mutations that inhibit the development of dystrophin have been identified in patients with DMD. These mutations cluster in hotspot regions of the genome that can in theory, be bypassed by various exon skipping strategies to return the dystrophin to an open reading frame. To date, however, there has been no appropriate long-term medication for this condition, and the only medication approved by the Food and Drug Administration for the treatment of DMD allows for the recovery of <1 per cent of the normal level of dystrophin protein after sustained treatment. There is however a significant unmet medical need for new strategies to correct the root cause of DMD—genetic mutations in the dystrophin gene.[6]

The rapid advancement of gene editing techniques derived from the invention of CRISPR/Cas9 now enables the alteration of the human genome and opens up the prospect of developing treatments for genetic diseases such as DMD.

The main contribution of this thesis is an automated way of progressing from Crispr Cas 9 gene editing to prime editing. With the aid of machine learning, we propose the most efficient pegRNAs for editing DMD exon 44 mutation in order to reframe the open reading frame of the exon 45, and finally to restore the dystrophin gene.

2 Biological background

2.1 CRISPR Gene Editing

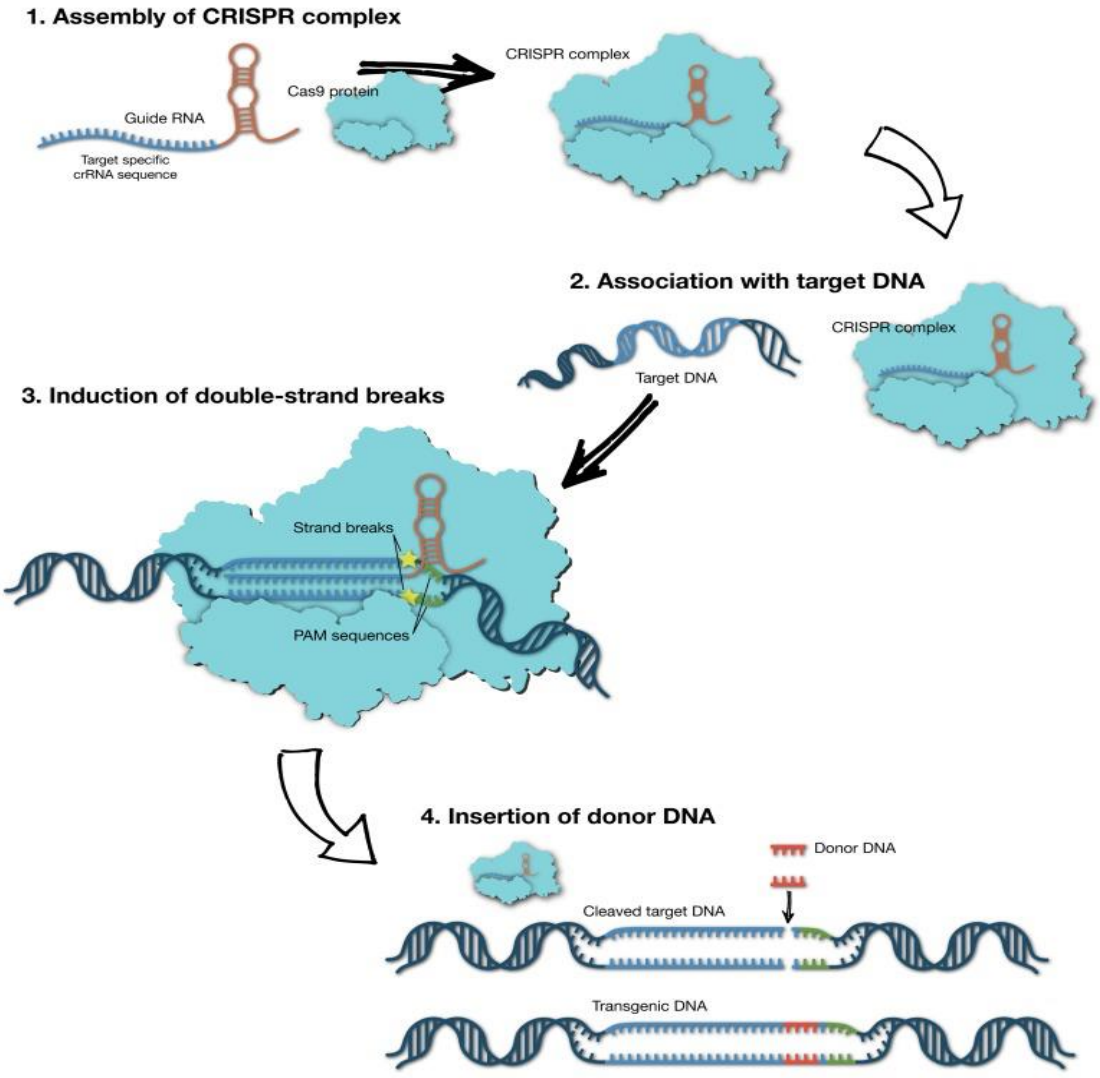


Figure 1: How does CRISPR work. (reproduced from *Joana R. Costa, Bruce E. Bejcek et al., Genome Editing Using Engineered Nucleases and Their Use in Genomic Screening, Assay Guidance Manual, 2004*)

CRISPR stands for clustered regularly interspaced short palindromic repeats. CRISPR "spacer" sequences are transcribed into short RNA sequences ("CRISPR RNAs" or "crRNAs") capable of guiding the system to matching sequences of DNA. When the target DNA is found, the Cas9, which is one of the enzymes produced by the CRISPR system, binds to the DNA and cleaves it. This results in shutting the targeted gene off. Using modified versions of Cas9, researchers have the ability to activate gene expression instead of cutting directly the DNA. These techniques allow researchers to study the gene's function and also modify it.[7]

Engineered CRISPR systems consist of two components: a guide RNA (gRNA or sgRNA) and a CRISPR-associated endonuclease (Cas protein). The gRNA is a short synthetic RNA composed of a scaffold sequence necessary for Cas-binding and a pre-defined ~20 nucleotide spacer that defines the genomic target to be modified. Thus, the genomic target of

the Cas protein can be modified by simply changing the target sequence present in the gRNA.[8]

2.2 The Cas9 nuclease

The Cas9 protein, an RNA-coordinated DNA endonuclease, is an amazing asset for controlling the genome. The simplicity of programming Cas9 (CRISPR-related protein 9) has empowered CRISPR (bunched, routinely interspaced, short palindromic rehashes)- based hereditary screens, recognizing settled qualities and giving novel knowledge into quality capacity for different phenotypes. Introductory libraries were planned with little information on sgRNA movement runs, a basic structure parameter, as deciphering screening information requires consistency among numerous sgRNAs focusing on a similar quality to recognize genuine hits from bogus positives. Dormant and vague sgRNAs lessen the powerful quality inclusion of the library and the exactness of the hit rundown. Numerous examinations show that Cas9 off-target action relies upon both sgRNA succession and trial conditions. These investigations have given subjective however inadequate comprehension of particularity determinants.

Finding generalizable examples is very testing, requiring huge informational collections to sufficiently test the tremendous number of conceivable blemished sgRNA-DNA communications to uncover grouping highlights for forecast of askew action. Here, we present the plan and portrayal of human and mouse genome-wide sgRNA libraries dependent on our recently distributed standards for foreseeing on-target efficiency. Expanding on screening information created with the new libraries and enormous scale appraisal of off-target movement, we create improved calculations for on-and off-target action expectation, permitting further advancement of our genome-wide libraries.[9]

2.3 Applications of CRISPR

2.3.1 Gene knockout using CRISPR

CRISPR can be used to generate knockout cells or animals by co-expressing an endonuclease like Cas9 or Cas12a (also known as Cpf1) and a gRNA specific to the targeted gene. The genomic target can be any ~20 nucleotide DNA sequence, provided it meets two conditions:

1. The sequence is unique compared to the rest of the genome.
2. The target is present immediately adjacent to a Protospacer Adjacent Motif (PAM).

The PAM sequence serves as a binding signal for Cas9, but the exact sequence depends on which Cas protein is used. The most popular nuclease used is *S. pyogenes* Cas9 (SpCas9). Once expressed, the Cas9 protein and the gRNA form a ribonucleoprotein complex through interactions between the gRNA scaffold and surface-exposed positively charged grooves on Cas9. Cas9 undergoes a conformational change upon gRNA binding that shifts the molecule from an inactive, non-DNA binding conformation into an active DNA-binding conformation. Importantly, the spacer region of the gRNA remains free to interact with target DNA.

Cas9 will only cleave a given locus if the gRNA spacer sequence shares sufficient homology with the target DNA. Once the Cas9-gRNA complex binds a putative DNA target, the seed sequence (8-10 bases at the 3′ end of the gRNA targeting sequence) will begin to anneal to the target DNA. If the seed matches the target DNA sequence, the gRNA will

continue to anneal to the target DNA in a 3′ to 5′ direction. Thus, mismatches between the target sequence in the 3′ seed sequence completely abolish target cleavage, whereas mismatches toward the 5′ end distal to the PAM often still permit target cleavage.

Cas9 undergoes a second conformational change upon target binding that positions the nuclease domains, called RuvC and HNH, to cleave opposite strands of the target DNA. The end result of Cas9-mediated DNA cleavage is a double-strand break (DSB) within the target DNA (~3-4 nucleotides upstream of the PAM sequence).

The resulting DSB is then repaired by one of two general repair pathways:

1. The efficient but error-prone non-homologous end joining (NHEJ) pathway
2. The less efficient but high-fidelity homology directed repair (HDR) pathway

The NHEJ repair pathway is the most active repair mechanism, and it frequently causes small nucleotide insertions or deletions (indels) at the DSB site. The randomness of NHEJ-mediated DSB repair has important practical implications, because a population of cells expressing Cas9 and a gRNA will result in a diverse array of mutations (for more information, jump to Plan Your Experiment). In most cases, NHEJ gives rise to small indels in the target DNA that result in amino acid deletions, insertions, or frameshift mutations leading to premature stop codons within the open reading frame (ORF) of the targeted gene. The ideal end result is a loss-of-function mutation within the targeted gene. However, the strength of the knockout phenotype for a given mutant cell must be validated experimentally.[8]

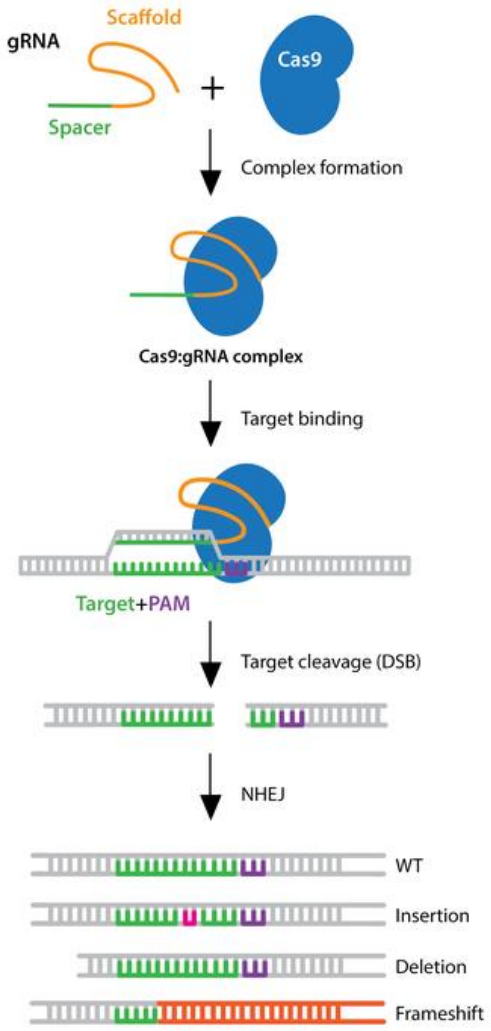


Figure 2: Gene knockout using CRISPR (reproduced from *https://www.addgene.org/guides/crispr/)*

2.4 Single guide RNA

sgRNA is an abbreviation for "single guide RNA." The sgRNA is a single RNA molecule that contains both the custom-designed short crRNA sequence fused to the scaffold tracrRNA sequence. sgRNA can be made either by synthetically generation or in vitro or in vivo from a DNA template.

Different genetic manipulations require different CRISPR components. A good way of selecting a specific genetic manipulation could narrow down the are appropriate reagents for a given experiment. The table below shows some of the different CRISPR components.

Table 1: Different CRISPR components (reproduced from *https://www.addgene.org/guides/crispr/)*

Genetic Manipulation	Application	Cas9	gRNA	Additional considerations
Knockout	Permanently disrupt gene function in a particular cell	Cas9 (or Cas9 nickase)	Single (or dual) gRNA targeting 5′	High-fidelity Cas enzymes increase specificity. Dual-nickase approach increases

	type or organism without a specific preferred mutation		exon or essential protein domains	specificity but is less efficient. Each putative knockout allele must be experimentally verified.
Edit	Generate a specific user-defined sequence change in a particular gene, such as generating a point mutation or inserting a tag	Cas9 (or Cas9 nickase) Base editor	Single (or dual) gRNA targeting the region where the edit should be made	HDR requires a repair template and displays reduced efficiency compared to NHEJ knockout. Base editors can make a limited set of mutations.

2.5 Applications of Crispr to DMD

The most recent developments in DMD therapy do not provide DMD with permanent therapy. As an appealing tool for DMD gene therapy, the CRISPR/Cas technology is both reliant and independent of the particular mutation.

CRISPR/Cas technologies can be used for modulating disease modifiers irrespective of the patient mutation. Using a single sgRNA strategy, full length dystrophin may be restored with respect to DMD replication mutations. The open reading frame can be restored for DMD deletion and point mutations by deleting or reframing exon(s) to create a shorter version of dystrophin. Using homologous recombination, full-length wild type dystrophin may also be recovered.[10]

2.6 Duchenne Muscle Dystrophy

Duchenne muscular dystrophy (DMD) is a severe form of muscular dystrophy that occurs primarily in males, though in rare cases may affect females. DMD causes progressive weakness and loss (atrophy) of skeletal and heart muscles.[11]

The DMD gene is responsible for the encoding of the dystrophin protein (***Error! Reference source not found.***). Dystrophin is a rod-shaped cytoplasmic protein found in muscle cells and is one of a group of proteins that work together to strengthen muscle fibers. Also, it protects them from injury as muscles contract and relax.[12]

The sequence of the gene can be found in various databases like UniProt and Ensembl. The id of the gene in the UniProt database is P11532 and in the Ensembl database is ENSG00000198947. The structure of the dystrophin gene is presented below.

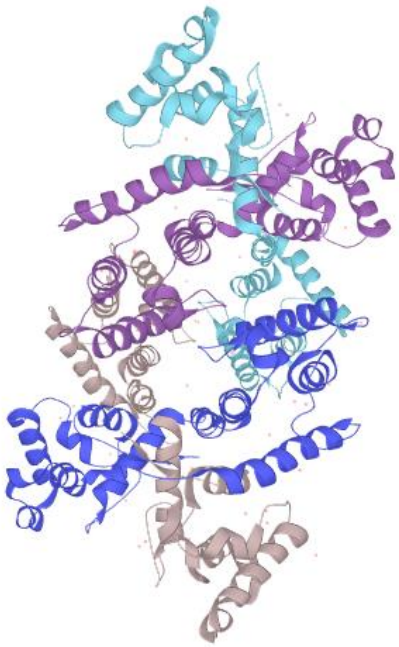


Figure 3: The dystrophin gene. *(reproduced from https://www.uniprot.org)*

DMD was first described by the French neurologist Guillaume Benjamin Amand Duchenne in the 1860. In 1986, MDA-supported researchers identified a particular gene on the X chromosome that, when flawed (mutated), leads to DMD. In 1987, the protein associated with this gene was identified and named *dystrophin*. Lack of the dystrophin protein in muscle cells causes them to be fragile and easily damaged. DMD has an X-linked recessive inheritance pattern and is passed on by the mother, who is referred to as a *carrier*.

DMD is characterized by progressive muscle degeneration and weakness due to the alterations of a protein called *dystrophin* that helps keep muscle cells intact. It is one of four conditions known as dystrophinopathies. The other three diseases that belong to this group are:

I. Becker Muscular dystrophy (BMD, a mild form of DMD)
II. an intermediate clinical presentation between DMD and BMD
III. and DMD-associated dilated cardiomyopathy (heart-disease) with little or no clinical skeletal, or voluntary, muscle disease.

DMD symptom onset is in early childhood, usually between ages 2 and 3. The disease primarily affects boys, but in rare cases it can affect girls. As the disease progresses, muscle weakness and atrophy spread to affect the trunk and forearms and gradually progress to involve additional muscles of the body. [13] [14]

2.6.1 The dystrophin gene

The dystrophin gene spans a genomic range of greater than 2 Mb and encodes a large protein (dystrophin) containing an N-terminal actin-binding domain and multiple spectrin repeats. The protein that is encoded forms a component of the dystrophin-glycoprotein complex (DGC). This bridges the inner cytoskeleton and the extracellular matrix. Duchenne muscular dystrophy, Becker muscular dystrophy or cardiomyopathy may be caused by deletions, duplications, and point mutations at the gene. Alternative promoter usage and alternative splicing result in numerous distinct transcript variants and protein isoforms for this gene.

As stated in the National Center for Biotechnology Information (NCBI), the cytogenetic location of dystrophin is Xp21.2-p21.1, which is the short (p) arm of the X chromosome between positions 21.2 and 21.1. The molecular location is made up of base pairs 31,119,219 to 33,339,460 on the X chromosome.

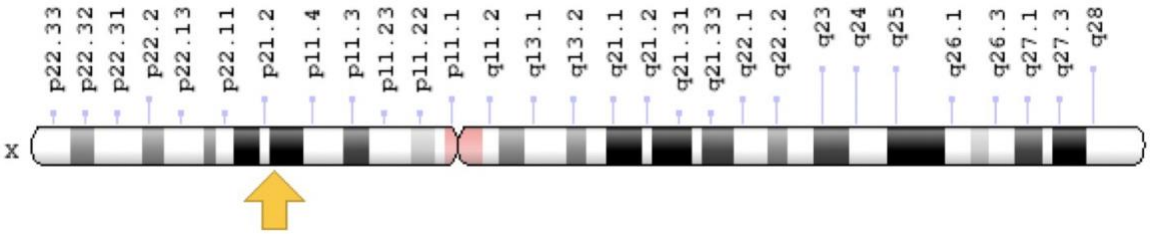


Figure 4: Chromosomal location. *(reproduced from https://ghr.nlm.nih.gov)*

The major DMD deletion is found between exons 45 and 53, thus making this a 'hot spot' area for 70% of DMD patients. Notably, skipping exon 51 is predicted to ameliorate the dystrophic phenotype in the greatest number of patients. Currently the *mdx* mouse is the most widely used animal model of DMD, although its mild phenotype limits its suitability in clinical trials.[15]

Mutations to the dystrophin gene are the main cause of DMD. Most commonly, one or more exons are missing, and the remaining exons don't fit together properly (exons are a portion of the gene). Because of this error in the genetic instructions, cells cannot make dystrophin, the protein that muscles need in order to working properly.[12]

2.6.2 Mutations

The four bases of adenine, cytosine, guanine and thymine are the fundamental building block s of DNA. They are commonly known by their letters A, C, G, and T respectively.
Three billion of these letters form the entire manual for the construction and maintenance of t he human body, but apparently tiny errors (mutations) can cause a disease.

Thousands of different mutations have been reported within the dystrophin gene. It is crucial to remember that everyone contains mutations in some of our genes, although we commonly do not comprehend it because the mutations do not have an effect on us in any substantial way.

The different types of mutations that can happen in the dystrophin gene are deletions, duplications and point mutations.

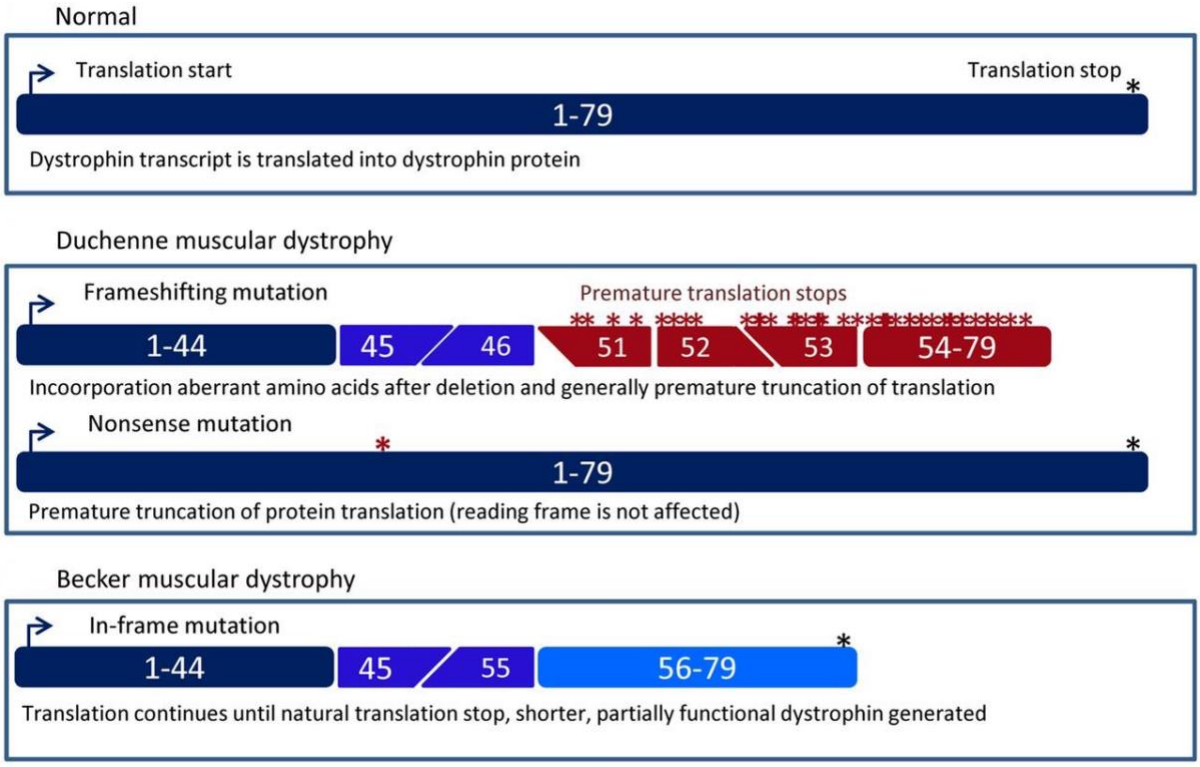


Figure 5: Schematic depiction of dystrophin transcripts in healthy, Duchenne muscular dystrophy (DMD) and Becker muscular dystrophy (BMD) individuals *(reproduced from Annemieke Aartsma-Rus, Ieke B. Ginjaar et al., Journal of Medical Genetics, 53, 3, 2)*

Deletions arise when pieces of the gene (known as exons) are missing. Small mutations can result in a premature stop signal (stop mutations), or they can disrupt the genetic code (a small deletion of duplication within and exon).[16]
Deletions of 1 or more exons are the most common sort of mutation. Since there are a complete of 79 exons within the dystrophin gene, there are many specific deletions that can arise. However, there are sure areas of the gene which might be more likely to have a deletion, and these areas are referred to as “hot spots”. Exons 44-55 are a hot-spot vicinity for deletions.

Duplications occur while one or more exons in the gene are doubled. Duplications are not as commonplace as deletions. Like deletions, duplications might occur in the course of all seventy-nine exons of the dystrophin gene.

Point mutations are smaller changes in the gene that do not involve a whole exon. Sometimes just one letter inside the DNA code is missing (deleted), doubled (duplicated), or changed. One of the most common factor mutations is called a nonsense mutation. Nonsense mutations motive a premature stop in the gene which leads to little or no dystrophin protein production.[17]

Genetic therapy is promising due to the fact that the dystrophin gene is inserted. However, several obstacles have been met along the way. The size of the dystrophin gene is the main obstacle in gene therapy. Thus, smaller genes, micro or mini dystrophin, have been developed, which can be inserted into a vector. The most suitable vector found so far is a virus associated with the adenovirus, a non-pathogenic parvovirus, but it has been shown to cause an immunological response. In order to assess the response, *mdx mice* dys-/dys- have been created, and there is evidence that when the gene is injected, the dystrophin is partially expressed, and muscular strength is improved. However, in preliminary studies on humans, 90

days after treatment initiation this gene expression was not observed. Results suggest that the success of this therapy is inhibited by cellular immunity.[18]

Exon-skipping therapy is one of the most promising therapeutic approaches aiming to restore the expression of a shorter but functional dystrophin protein. The antisense field has remarkably progress over the last years with recent accelerated approval of the first antisense oligonucleotide-based therapy for DMD, Exon 51, though the therapeutic benefit remains to be proved in patients. Despite clinical advances, the poor effective delivery to target all muscle remains the main hurdle for antisense drug therapy.[19]

In the current thesis the main focus will be in genetic therapy, and more precisely in the method of CRISPR using the enzyme Cas9.

2.6.3 Existing Crispr methods for potential DMD treatment

Below is a brief presentation of related works for the treatment of DMD using CRISPR Cas9:

Table 2: Related works targeting the DMD gene

Target Gene Region	**Protospacer sequence (5'-3')**	**Reference**
DMD exon 44	CTTACAGGGAGAACTCCAGGA	[6]
DMD introns 45-55	CTGGACGGAGCTGGTTTATCT	[20]
various DMD exon 23 indel mutations	CTATCTGAGTGACACTGTGA	[21]
DMD exon 45	GGCTTACAGGAACTCCAGGA	[22]

Yi-Li Min. et. al. developed a patient-derived induced pluripotent stem cell (iPSCs) from a DMD patient without exon 44 of the dystrophin gene (DMD). The regular dystrophin gene from the sibling of the same patient was set as a healthy monitoring system. Deletion of exon 44 (Ex44) disrupts the open reading frame of dystrophin by inducing splicing in exon 43 to exon 45 and the implementation of early termination codon. Using CRISPR-Cas9 gene editing to skip exon 43, which allows splicing between exons 42 and 45, or to skip exon 45, which allows splicing between exons 43 and 46, the reading frame can be restored. Alternatively, by adding one nucleotide (+3n+1 insertion) or removing two nucleotides (+3n−2 deletion), the reframing of exon 43 or 45 may restore the protein reading frame.[6]

Courtney S. Young et. Al. developed a novel dystrophic mouse model by using CRISPR/Cas9 to delete exon 45 in the human DMD gene in DMD mice, which makes DMD out of frame. They used this model to show that their clinically applicable CRISPR/Cas9 interface, which targets the removal of human DMD exons 45–55, can be directly used in vivo to restore dystrophin.[20]

Taeyoung Koo. et. al. used Cas9 which was extracted from S. Pyogens in order to produce Dmd knockout mice with a frameshift mutation in the DMD gene. Then, they expressed CjCas9, its single-guide RNA, and the EGFP gene in the tibialis anterior muscle of the Dmd knockout mice using an all-in-one adeno-associated virus (AAV) vector. CjCas9 effectively cut the target site in the DMD gene in vivo and caused minor insertions or deletions at the target site. Their study resulted in the conversion of the damaged DMD reading frame from out of frame to in frame, leading to the expression of dystrophin in the sarcolemma. Importantly, muscle strength was improved in CjCas9-treated muscles, with no off-target mutations, suggesting high efficiency and specificity of CjCas9. That study showed that CjCas9-mediated in vivo DMD frame correction has significant potential for the treatment of DMD and other neuromuscular diseases.[21]

Yu Zhang et. Al. loaded a Cas9 nuclease in a single-stranded AAV (ssAAV) and CRISPR single-stranded AAV (scAAV) guide RNAs and supplied that dual AAV system to a DMD mouse model. The dose of scAAV needed for effective genome editing was at least 20 times lower than that of ssAAV. Mice undergoing systemic therapy demonstrated preserved expression of dystrophin and strengthened muscle contractility. Their results indicated that the performance of CRISPR-Cas9-mediated genome editing can be greatly increased by using the scAAV method.[22]

2.7 Prime Editing

2.7.1 Prime editing strategy

Some genetic variants leading to various diseases are difficult to correct effectively and witho ut surfeit by-products. Prime editing is a versatile and precise genome editing method that directly writes new genetic information into a specified DNA site using a catalytically impaired Cas9 endonuclease fused to an engineered reverse transcriptase, programmed with a prime editing guide RNA (pegRNA) that both specifies the target site and encodes the desired edit. *Anzalone et. al*, performed more than 175 edits in human cells, including targeted insertions, deletions, and all 12 types of point mutation, without requiring double-strand breaks or donor DNA templates.

Prime editing guide Ribonucleic Acid (pegRNA) is an important component of the Prime Editing system due to the fact that it specifies the target site but also encodes the desired edit and prime reverse transcription. The edit from the pegRNA is transferred into the target site while, a branched intermediate is formed with two single strand DNA flaps; an unedited 5′ flap and a 3′ flap with the edited sequence from the pegRNA.[23]

Prime editing was used in human cells to correct, efficiently and with few byproducts, the primary genetic causes of sickle cell disease (requiring a transversion in *HBB*) and Tay–Sachs disease (requiring a deletion in *HEXA*); to install a protective transversion in *PRNP*; and to

insert various tags and epitopes precisely into target loci. Four human cell lines and primary post-mitotic mouse cortical neurons support prime editing with varying efficiencies. Prime editing presented higher or similar efficiency and fewer byproducts than homology-directed repair, had complementary strengths and weaknesses compared to base editing, and induced much lower off-target editing than Cas9 nuclease at known Cas9 off-target sites.

Prime editing substantially could expand the scope and capabilities of genome editing, and in principle could correct up to 89% of known genetic variants associated with human diseases. Cas9 targets DNA employing a direct RNA containing a spacer arrangement that hybridizes to the target DNA site. Engineered guide RNAs were used in order to specify the DNA target. Also, they contained new genetic information that replaces target DNA nucleotides. To exchange data from these built direct RNAs to target DNA, the genomic DNA had to be utilized, scratched at the target location to uncover a 3'-hydroxyl gather, to prime the turnaround translation of an edit-encoding expansion on the built direct RNA (in the future alluded to as the prime altering direct RNA, or pegRNA) straightforwardly into the target location.[2]

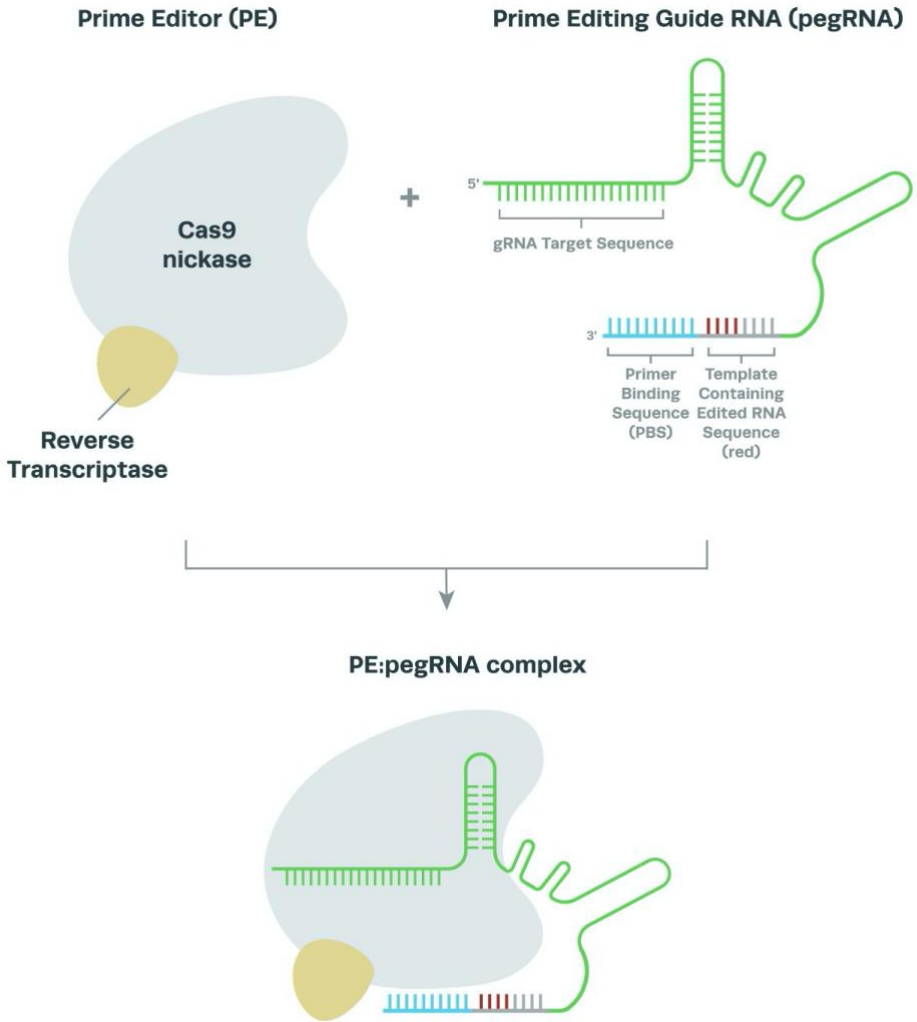


Figure 6: Components of Prime Editing (reproduced from *https://www.synthego.com/blog/prime-editing*)

The main advantage of prime editing over CRISPR is that the first's enzymes do not have to break both strands of DNA to create changes, liberating researchers from depending on the

cell's DNA repair framework, which they cannot control to create the alters that they need. This implies that prime editing might empower the advancement of medicines for hereditary maladies caused by changes that are not effectively tended to by existing gene-editing tools.[24]

This gene editing method shows high accuracy levels. Off-target cuts were below 10 percent, and less than one-tenth of edited cells had unwanted changes to their genome. On the contrary off-target cuts were up to 90 percent for first-gen CRISPR systems.[25]

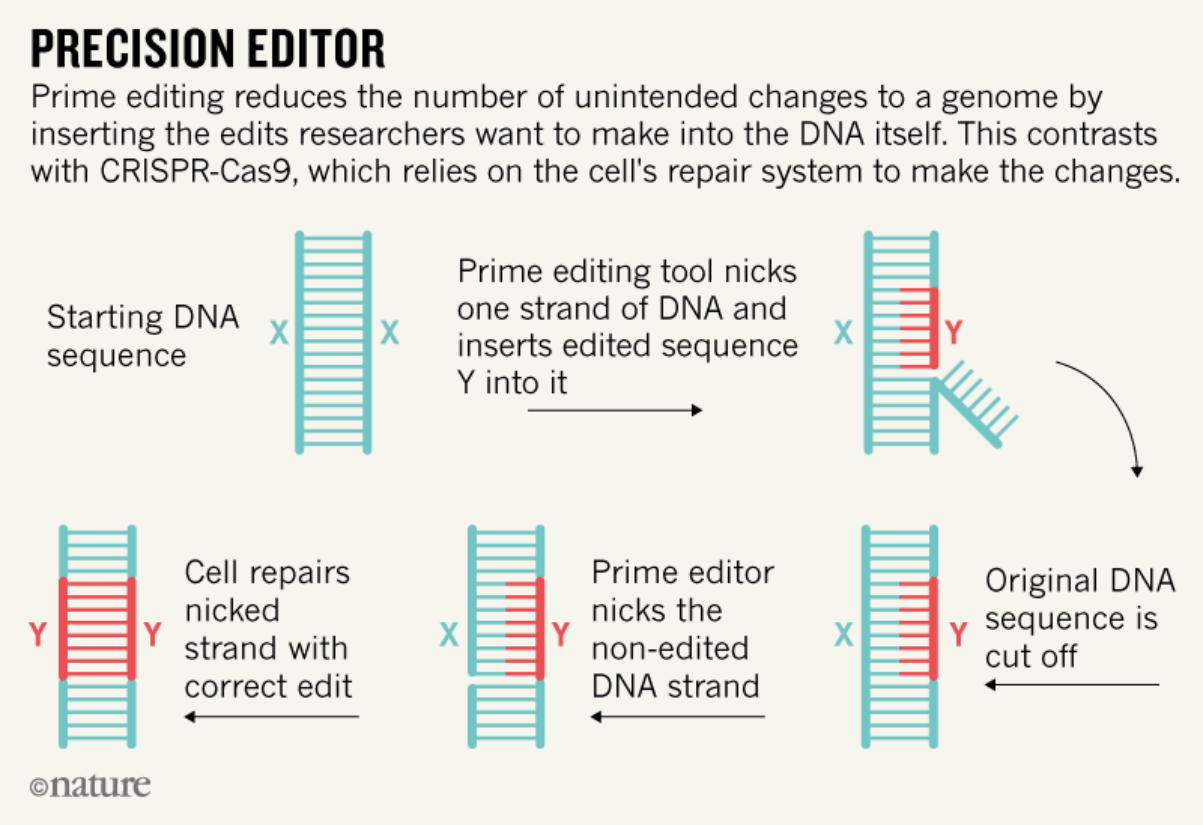


Figure 7: Prime editing relies on precision editing (reproduced from *Heidi Ledford, Super-precise new CRISPR tool could tackle a plethora of genetic diseases, NLM (Medline), 2019*).

This gene editing method shows high accuracy levels. Off-target cuts were below 10 percent, and less than one-tenth of edited cells had unwanted changes to their genome. On the contrary off-target cuts were up to 90 percent for first-gen CRISPR systems.[25]

2.7.2 Genes – Diseases correlations

We searched through bibliography given the gene names for the associated diseases described in *Anzalone et. al.,* in order to understand if there was a connection between the various protospacers used in CRISPR experiments and the spacers used in Prime Editing.
All genes from 'Search-and-replace genome editing without double-strand breaks or donor DNA' by *Anzalone et. al,* were bibliographically inspected in order to find each associated disease.

Table 3: Genes and related diseases.

Abbreviation	Related Gene	Associated Disease
HBB	Hemoglobin	Sickle cell disease
HEXA	Hexosaminidase A	Tay–Sachs disease
HEK3	Ephrin type-A receptor 8	Epithelial ovarian cancer
HEK4	EPH receptor A3	Lung cancer
EMX1	Homeobox protein EMX1	Kallmann Syndrome and Epileptic Encephalopathy
FANCF	Fanconi anemia group F protein	Fanconi Anemia
DNMT1	DNA (cytosine-5)-methyltransferase 1	Certain human tumors and developmental abnormalities
RUNX1	Runt-related transcription factor 1	Several types of leukemia
VEGFA	Vascular Endothelial Growth Factor A	Microvascular Complications of Diabetes 1 and Poems Syndrome
RNF2	Ring Finger Protein 2	Angelman Syndrome

The gene names as well as the associated diseases to each gene or coding protein were found using GeneCards. GeneCards is a searchable, integrative database that provides comprehensive, user-friendly information on all annotated and predicted human genes. The knowledgebase automatically integrates gene-centric data from ~150 web sources, including genomic, transcriptomic, proteomic, genetic, clinical and functional information.[17]

The following table shows the protospacers that were used in various CRISPR experiments.

Table 4: Protospacers used in various CRISPR experiments

Gene	Protospacer used in CRISPR	Reference
HBB	GTAACGGCAGACTTCTCCAC	[26]

HEK3	GGCCCAGACTGAGCACGTGA	[27]
HEK4	GGCACTGCGGCTGGAGGTGG	[28]
EMX1	GAGTCCGAGCAGAAGAAGAA	[29]
FANCF	GGAATCCCTTCTGCAGCACC	[29]
DNMT1	GATTCCTGGTGCCAGAAACA	[30]
RUNX1	GCATTTTCAGGAGGAAGCGA	[29]
VEGFA	GATGTCTGCAGGCCAGATGA	[31]
RNF2	GTCATCTTAGTCATTACCTG	[27]

The protospacers of the CRISPR experiments were the same with the spacers used in the prime editing experiments described in *Anzalone et. al*.

Table 5: Protospacers used in CRISPR and spacers used in Prime Editing.

Gene	**Protospacers used in CRISPR**	**Spacer used in Prime Editing**
HBB	GTAACGGCAGACTTCTCCAC	GTAACGGCAGACTTCTCCAC
***HEXA**	-	GATCCTTCCAGTCAGGGCCAT
HEK3	GGCCCAGACTGAGCACGTGA	GGCCCAGACTGAGCACGTGA
HEK4	GGCACTGCGGCTGGAGGTGG	GGCACTGCGGCTGGAGGTGG
EMX1	GAGTCCGAGCAGAAGAAGAA	GAGTCCGAGCAGAAGAAGAA
FANCF	GGAATCCCTTCTGCAGCACC	GGAATCCCTTCTGCAGCACC
DNMT1	GATTCCTGGTGCCAGAAACA	GATTCCTGGTGCCAGAAACA
RUNX1	GCATTTTCAGGAGGAAGCGA	GCATTTTCAGGAGGAAGCGA
VEGFA	GATGTCTGCAGGCCAGATGA	GATGTCTGCAGGCCAGATGA

RNF2	GTCATCTTAGTCATTACCTG	GTCATCTTAGTCATTACCTG

For the HEXA gene, there was not found any CRISPR protospacer identical to the one described in Anzalone et. al. The largest portion of the 3' extension is common to the same groups of pegRNAs (same pegRNAs refer to the same genomic group as HBB, HEXA etc).

Also, within this common sequence is the PBS (Primer Building Site) and the RT template. So, the sequence that was changed every time in the 3' extension among the same group of pegRNAs was the RT template. Thus, *Anzalone et. al.* experimented on different templates in order to find out which made the best repair.[2]

The complementary sequences were found using the online tool http://arep.med.harvard.edu/labgc/adnan/projects/Utilities/revcomp.html

In the paper 'Search-and-replace genome editing without double-strand breaks or donor DNA' of *Anzalone et. al*, are presented the pegRNAs that had the highest biological efficiency in the conducted experiments for each target gene. In the table below, those pegRNAs are given along with the target gene.

2.8 Dystrophin gene and exon 44 mutation

The second most commonplace mutational hotspot in the dystrophin gene consists of exon 44, which disrupts the open reading frame in surrounding exons. Deletion of exon 44 disrupts the open analyzing frame of dystrophin through causing splicing of exon 43 to exon 45 and introducing a premature termination codon.[6]

This would result in patients with DMD not having exon 44 in their mature mRNA. DNA coding goes from exon 43 straight to 45, resulting in out of frame mutation. So, the DNA deletion mutation would be corrected either at exon 43 or exon 45. In either case it can be done by insertion of one bp or the deletion of two bps.

The sequence of both the coding and the non-coding strand were found using Benchling.[32]

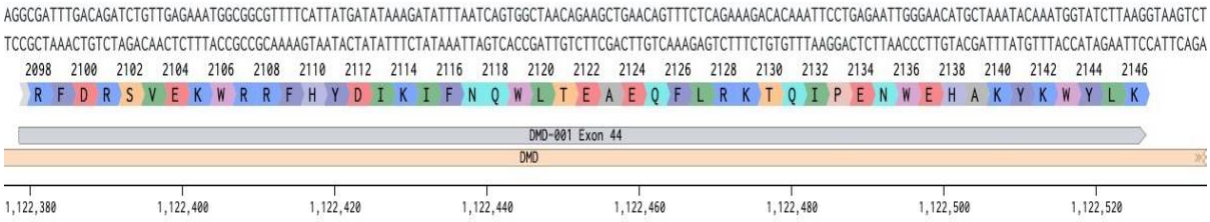


Figure 8: Sequence of exon 44

Benchling is an online resource that helps researchers manage and coordinate laboratories and experimental results by offering experimental design and data analysis software tools for the molecular biology. Benchling provides tools for functions such as priming design and colony counting, as well as CRISPR design guides and automated cloning of Gibson and Golden Gate. Users can take data notes in line with data, link data through entries, keep files and data in one place, and manage and monitor team progress.[33]

3 The dataset

Anzalone et. al. shared with us the data that were produced from the experiments for both HBB and HEXA genes. Both datasets included data for those genes from multiple experiments. Those data included each pegRNA with the corresponding spacer sequence, the 3' extension, along with the PBS length and the RT template length. In order to use those data, we took and trained the machine learning algorithms with the mean value of each correct edit.

The data described in the following sections were taken from *Anzalone et. al*. From the original dataset, only data for the genes HBB and HEXA were kept.

Also, we kept the columns that provided the information for the 3' extension sequence along with the columns with the correct edit for each of the three replications. During the preprocess of the dataset we computed '*the mean of correct edit*', the percentage that showed the correct edit of each 3' extension.

Table 6: Sample of the dataset

pegRNA	Spacer sequence	3' extension	PBS length	RT template length	Mean of % correct edit (w/o indels)
HEXAs 1	GATCCTTCCAGTCAGGGCCAT	ATATCTTATGGCCCTGACTGGAA	13	14	0,24436087
HEXAs 2	GATCCTTCCAGTCAGGGCCAT	TATATCTTATGGCCCTGACTGGAA	13	15	0,200435593
HEXAs 3	GATCCTTCCAGTCAGGGCCAT	GTATATCTTATGGCCCTGACTGGAA	13	16	0,97492582
HEXAs 4	GATCCTTCCAGTCAGGGCCAT	ACCGTATATCTTATGGCCCTGACTGGAA	13	19	3,501457357
HBB 3.7	GCATGGTGCACCTGACTCCTG	AGACTTCTCCTCAGGAGTCAGGTGCAC	13	14	38,342347846
HBB 5.2	GCATGGTGCACCTGACTCCTG	TAACGGCAGACTTCTCCTCAGGAGTCAGGTGCAC	13	19	30,649821844
HBB 5.3	GCATGGTGCACCTGACTCCTG	ACGGCAGACTTCTCCTCAGGAGTCAGGTGCAC	13	17	34,232669115
HBB 5.4	GCATGGTGCACCTGACTCCTG	GGCAGACTTCTCCTCAGGAGTCAGGTGCAC	13	16	43,945128711

4 Computational approach for pegRNAs creation

The protospacer that was used in the paper 'CRISPR-Cas9 corrects Duchenne muscular dystrophy exon 44 deletion mutations in mice and human cells' was taken to be part of the produced pegRNA. After an extensive bibliographic research, we concluded that all of the protospacers used in various CRISPR-Cas 9 experiments, were part of the pegRNAs introduced in *Anzalone et. al.* There were three steps followed to produce the pegRNA targeting the dystrophy exon 44:

1. The protospacer mentioned in the paper above was taken in order to be part of the produced pegRNA.

2. The position of the mutation was detected within the gene. From bibliographic research it resulted that the mutation was a small deletion. As a result, the cell skips the exon 43 and 45.

3. The protospacer position was found within the gene and it was placed precisely in the same position as the 3' extension on the complementary chain.

Two rules applied in the creation of the pegRNA: as mentioned in the work of *Anzalone et. al.*, the PBS length had to be between twelve to fifteen nucleotides and the RT template length had to be equal or greater than seven nucleotides. Also, according to *Anzalone et. al.*, the total length of the pegRNA would be:

- For insertions 1bp to ≥ 44bp
- For deletions 1bp to ≥ 80bp

4.1 pegRNAs creation for reframing exon 45

In our approach, we created pegRNAs for both the insertion and deletion of bps in order to repair the small deletion occurring in exon 44. In the case of insertion, the goal of the produced pegRNAs was to insert +1bp in the exon 45, in order to reframe the genetic information. In the case of deletion, we deleted -2 bp in order to achieve the reframing of the same exon.

As one could see from the image below, *Yi-Li Min et. al*. presented 34 different candidates for CRISPR editing of exon 45 along with the wildcard gene (HC). The image presents an analysis of the sgRNAs that target the splice acceptor or donor sites for exons 43 and 45, along with the corresponding base modifications and the result in the dystrophin gene.

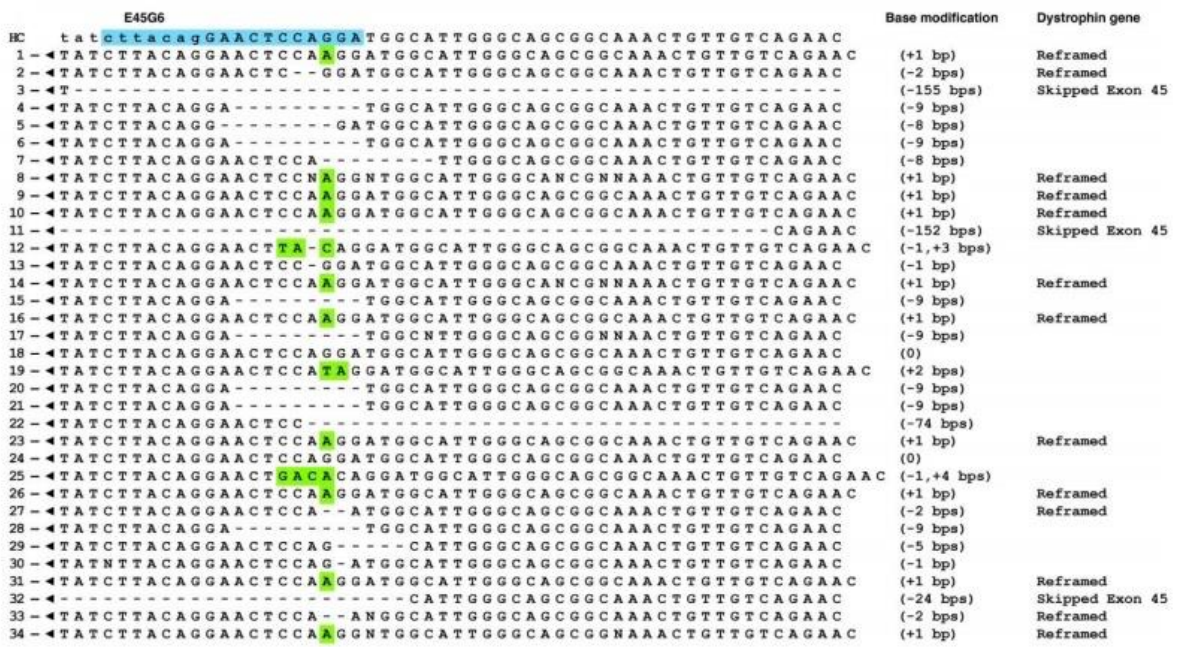


Figure 9: Analysis of sgRNAs that target the splice acceptor or donor sites for exons 43 and 45 (reproduced from *Yi-Li Min et. al. Supplementary Materials for CRISPR-Cas9 corrects Duchenne muscular dystrophy exon 44 deletion mutations in mice and human cells, Science Advances, 06 Mar 2019)*

We chose to reproduce the candidates sgRNAs 1 and 2 through Prime Editing. The reason for choosing only these two was the restrictions that are applied as described in *Anzalone et. al.* For instance the candidate 11 was excluded, as we mentioned in the previous section the maximum number of a pegRNAs for the case of a deletion would be of a maximum of 80bp. Another example of an excluded candidate would be the sgRNA 32, as another restriction of pegRNAs is that the edit should occur inside the PBS. As we can see the sgRNA 32 lacks most of the bps that would be needed for creating the proportional PBS.

4.1.1 Insertion and deletion for reframing exon 45

Initial steps were followed so that the pegRNAs targeting exon 44 could be created:

1. Detection of the mutation in the mutant gene.

2. Detection of PAM site (NGG) next to the mutation site.

3. If there is a PAM site nearby, according to *Anzalone et. al.*, the complementary to PAM strand would be used for the creation of the RT template.

4. Selection of the length of both the RT template and the PBS.

More specifically, we started by searching Benchling for the DMD gene by giving the chromosomal location and the desired organism (homo sapiens in our case study). Although, the edited gene described in the paper by *Min et. al.*, referred to the genomic loci of mus musculus, we searched for the human DMD gene as that same genomic region is conserved by a hundred percent among mus musculus and homo sapiens. After locating the exon 45, we verified that the base pairs of the protospacer used in the Crispr experiment from *Min et. al.* targeting the homonymous exon, were found in the beginning of the same exon. Having

assumed that the top chain is the coding strand and the bottom is the non-coding strand the protospacer orientation was on the left 5′ and on the right 3′. Counter to the coding strand, the non-coding was 3′ on the left and 5′ on the right.

It is of vital importance to add that we tested our approach for an automated way of creating the desired pegRNAs for both cases of exon 45, after testing the method on the data provided by *Anzalone et. al.* As a result we produced the same pegRNAs mentioned in the supplementary information of the paper "*Search-and-replace genome editing without double-strand breaks or donor DNA*", thus making the assumption that the procedure that we followed was correct.

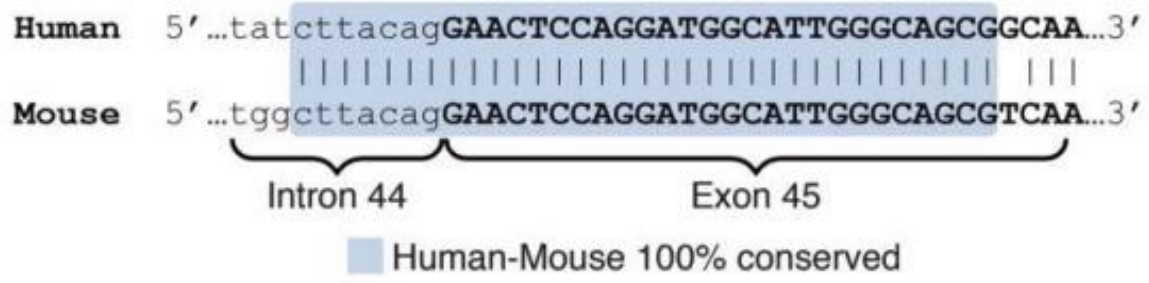


Figure 10: Conserved region between the two species. (*reproduced Yi-Li Min et. al., Supplementary Materials for CRISPR-Cas9 corrects Duchenne muscular dystrophy exon 44 deletion mutations in mice and human cells, Science Advances, 06 Mar 2019)*

In order to understand how to locate the base pairs for the creation of the different RT templates, we studied the HBB and HEXA spacers used in *Anzalone et. al.* Our study resulted in a pattern where the first three bps of the minimum RT template would be the last three complementary bps of the protospacer that are right next to the PAM site. As in our case study we wanted to edit the gene by adding a +A for the reframing to happen, we added a +T at the very beginning of the RT template. This was done, because as we concluded from studying *Anzalone et. al.*, the correct edit must be included in the RT template.

The next three bps added, resulting in a total of seven bp for the minimum RT template, were the complementary bases moving downstream to the 5′ end of the non-coding strand. We created a total of twenty-three different RT templates by adding each time one bp to the minimum RT template until the total number of bps of the maximum in length RT template was twenty-nine. Anzalone et. al. had not mentioned anything about the total number of bps for the RT template, but after studying thoroughly the supplementary material provided with the main article, we concluded that the specific number of bps for the maximum in length RT would be the desired one.

The location of the RT template revealed the bps of the minimum PBS, after studying the same supplementary material we concluded that the first bp of the PBS would be right next to the site where the RT template was found. *Anzalone et. al.* suggested that the PBS would be in total length of minimum of 8 bps to a maximum of 15 bps. First, we created a minimum PBS by adding seven more bps moving towards to the 3′ end of the same strand, resulting in a total of eight bps. In order to generate, the other PBS up to the maximum length, we added each one bp moving again towards to the 3′ end.

One hundred eighty-four different 3' extension were produced for reframing the dystrophin gene by inserting a +A bp. All sequences are shown in 5' to 3' orientation. The 3' extensions contain both the PBS and the RT template.

It is very important to point out that *Anzalone et. al*. recommends designing pegRNAs such that the first base of the 3' extension is not C. As, a result we did not keep any 3' extensions that the first bp was C.

4.1.2 Pseudocode for generating the 3' extensions for insertion

A more detailed guide with the initial steps for generating the 3' extensions is presented below:

1. Search Benchling for the DMD gene by giving the chromosomal location and the desired organism (homo sapiens).

2. Locate the target exon.

3. Verify that the base pairs of the protospacer used in the Crispr experiment. targeting the homonymous exon, were found in the beginning of the same exon.

4. The first three bps of the minimum RT template would be the last three complementary bps of the protospacer that are right next to the PAM site.

5. **Inputs:**

    a. Coding strand of the targeted exon e.g., "*AAAAAGACATGGGGCTTCATTTTTGTTTTGCCTTTTTGGTATCTTACAGGAACTCCAGG ATGGCATTGGGCAGCGGCAAACTGTTGTCAGAACATTGAATGCAACTGGGGAAGAAATA ATTCAGCAATCCTCAAAAACAGATGCCAGTATTCTACAGGAAAAATTGGGAAGCCTGAAT CTGCGGTGGCAGGAGGTCTGCAAACAGCTGTCAGACAGAAAA*"
    b. PAM sequence e.g., "*TGG*"
    c. Protospacer sequence e.g., "*CTTACAGGAACTCCAGGA*"
    d. Place the correct base or bases that will be inserted e.g., "*T*"

6. **Calculations:**

    a. Find the PAM sequence
    b. Find the protospacer sequence
    c. Create the minimum RT Template by adding the next three bps of the PAM, resulting in a total of seven bp
    d. Generate multiple RT templates by adding each time one bp to the minimum RT template until the total number of bps of the maximum in length RT template is 29.
    e. Based on the location of the RT template find the bps of the minimum PBS (the first bp of the PBS would be right next to the site where the RT template was found).
    f. Create the minimum PBS by adding seven more bps moving towards to the 3′ end of the same strand.
    g. Merge all the produced PBS and RT Templates, to generate the 3′ extensions

All the same steps apply for the case of deletion with the only difference, that one does not have to input any base or bases, as the protospacer that would be inputted carries the deleted base or bases.

5 Selection of the most promising pegRNAs using machine learning

Due to the huge volume of data produced by the in vivo and in vitro experiments, the need for in silico experiments arises. Machine learning could potentially help eliminate the need for making all of those experiments happen. Our goal is to produce a tool that could help Biologists chose the most efficient pegRNAs for gene editing by using mainly computational methods. This could be done by finding the optimal algorithms for each specific biological problem and each time tuning the algorithmic parameters so that scientists could achieve the same results but without the need of exhausting all of the laboratory resources and time needed.

Different algorithms were tested on the data in order to find the one that could predict the target variable more precisely. Those algorithms were: Multiple Linear Regression, Support Vector Regression, Random Forest Regressor and Gradient Boosting Regressor.

5.1 Multiple Linear Regression

Multiple linear regression (MLR), also known simply as multiple regression, is a statistical technique that predicts the outcome of a response variable using many explanatory variables. Multiple linear regression (MLR) is intended to model the causal relationship between the explanatory (independent) variables and the variable response (dependent).[34]

5.2 Support Vector Regression

Support Vector Machines are very different algorithm types, distinguished by the use of kernels, lack of local minima, sparse solution and capacity control obtained by acting on the edge, or number of support vectors, etc. They were conceived by Vladimir Vapnik and his colleagues, and first presented with the paper at the 1992 Conference on Computational Learning Theory (COLT). All these nice features were however already present in machine learning since the 1960s: use of kernels by wide margin hyper planes, geometric representation of kernels as internal products in a feature space. Different methods of optimization were used in pattern recognition and there was widespread discussion of sparseness methods. The use of slack variables was implemented in the 1960s to address noise in the data and non-separability.[35]

5.3 Random Forest Regressor

A random forest is a meta estimator that matches a number of decision trees on different dataset sub-samples to be listed. This algorithm makes use of average to improve predictive accuracy and over-fitting power. The sub-sample size is always the same as the original input sample size but the samples are drawn with replacement if bootstrap is True (default).[36]

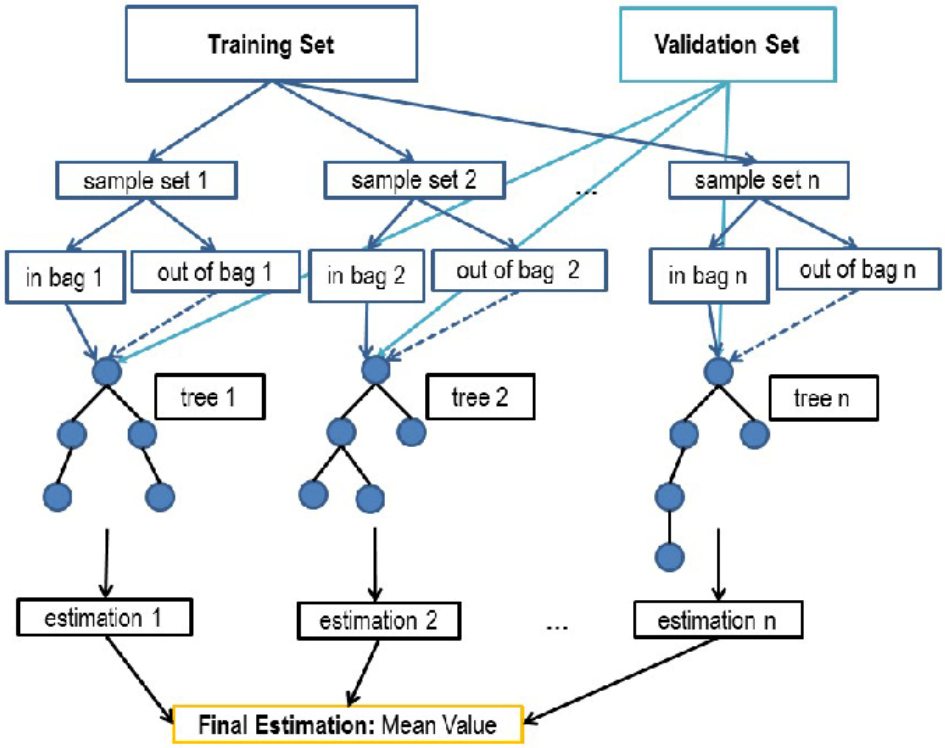


Figure 11: Structure of the Random Forests regressor *(reproduced from Liarokapis et al., Learning the Post-Contact Reconfiguration of the Hand Object System for Adaptive Grasping Mechanisms, Conference: IEEE/RSJ)*

5.4 Gradient Boosting Regressor

The logic behind gradient boosting is basic, as it could be seen instinctively, without utilizing scientific documentation. A basic assumption of linear regression is that sum of its residuals is 0, i.e. the residuals should be spread randomly around zero.
Now think of those residuals as errors committed by our model of predictors. While tree-based models are not based on these assumptions, if we think about this assumption logically, we might say that if we can see any residual pattern about 0, we can use that pattern to match a standard.[37]

5.5 Experimental settings

For applying the machine learning algorithms on the dataset, we tested the data on various train / test splits, and we concluded that the ideal percentage, was 70% for training the algorithms and 30% for testing. To decide the optimal values to be used for our model hyperparameters from a given range of values, we then used the grid search cross validation method from the Scikit-Learn library.

For the Gradient Boosting Regressor algorithm, we chose to optimize the four hyperparameters: learning rate, max-depth, max-features and min-samples-leaf.

- Max_depth = overall depth of the tree
- Learning_rate = shrinks the contribution of each tree by learning_rate
- Max-features = the number of features to consider when looking for the best split

- Min_samples_leaf = the minimum number of samples required to be at a leaf node. A split point at any depth will only be considered if it leaves at least min_samples_leaf training samples in each of the left and right branches.[38]

Although model parameters such as the slope and intercept in a linear regression are learned during training, hyperparameters must be set by the data scientist before training. Hyperparameters include, in the case of a random forest, the number of decision trees in the forest and the number of characteristics considered by each tree when splitting a node. (The variables and thresholds used to break every node learned during training are the parameters of a random forest). For all models, Scikit-Learn introduces a set of rational default hyperparameters, but these are not guaranteed to be optimal for a problem. In general, it is difficult to decide the best hyperparameters in advance, and tuning a model is where machine learning transforms from science into trial-and-error-based engineering. Hyperparameter tuning depends more on experimental results than on theory, so trying several different combinations to test the output of and model is the best way to decide the optimal settings. In machine learning, however, testing each model only on the training set can lead to one of the most fundamental problems: overfitting. The hyperparameters that were searched were:

- n_estimators = number of trees in the forest
- max_features = max number of features considered for splitting a node
- max_depth = max number of levels in each decision tree
- min_samples_split = min number of data points placed in a node before the node is split
- min_samples_leaf = min number of data points allowed in a leaf node
- bootstrap = method for sampling data points (with or without replacement)

5.5.1 Parameters tuning – Grid Search with Cross Validation

Grid Search was used for finding the best Model parameters for the dataset. Although model parameters are learned during training — like the slope and intercept in a linear regression — one must set hyperparameters prior to training. In the case of a random forest, the hyperparameters include the number of decision trees in the forest and the number of characteristics that each tree considers when dividing a node. (The variables and thresholds used to split each node learned during training are the parameters of a random forest). For all models, Scikit-Learn implements a set of sensible default hyperparameters but these are not guaranteed to be optimal for a problem. It is generally difficult to decide the best hyperparameters in advance, and tuning a model is where machine learning transforms from a science into trial-and - error dependent engineering.

Hyperparameter tuning depends more on experimental tests than on theory, and so the best way to assess the optimal settings is to try to test the output of each model in several different combinations. Evaluating each model only on the training set, however, will result in one of the most basic machine learning problems: overfitting. If model is optimized for the training data, it will perform very well on the training set, but it will not be able to generalize to new data as in a test set for example. This is known as overfitting, or simply designing a model that knows the training set very well but cannot be applied to new problems when a model performs strongly on the training set but poorly on the test set.[39]

First, GridSearchCV was imported from the Scikit-Learn library, which is a python machine learning library. GridSearchCV's estimator parameter includes the model that was used for the

method of tuning the hyper-parameter. We used the GridSearchCV for the Random Forest Regressor and the Gradient Boosting Regressor. Also, we further divided our training set into K=10 numbers of subsets, called folds, in K-Fold CV. We then fit the model 10 times iteratively, training the data on K-1 of the folds each time and evaluating on the 10th fold (called the validation data) each time. We averaged the results on and of the folds at the very end of training to come up with final validation metrics for the model.

5.6 Feature representation

In order to train a machine learning algorithm on the data, the input DNA sequences are required to be encoded as numerical values and represented as either vectors or multi-dimensional matrices.[40]

There are many approaches for Machine Learning with DNA sequence data, in the current work we used three different ways of encoding the sequence information:

1. ordinal vectors
2. one-hot encoding
3. k-mer counting

When a sequence is coded based on its nucleotide composition, it essentially results in a multidimensional vector which dimensions depend on the original sequence length. The problem that we had to deal with was that when converting sequences of different length using ordinal vectors and one hot encoding, the generated vectors had different sizes, thus resulting to NaN values between the vectors mismatches. To overcome this, we tried two different approaches. The first was to replace the NaN values with zeros and the second was to convert the vectorized sequences to same length before training the algorithms on the data.

In the following sections, the different nucleotide representations will be analyzed thoroughly.

5.6.1 Encoding DNA sequence data

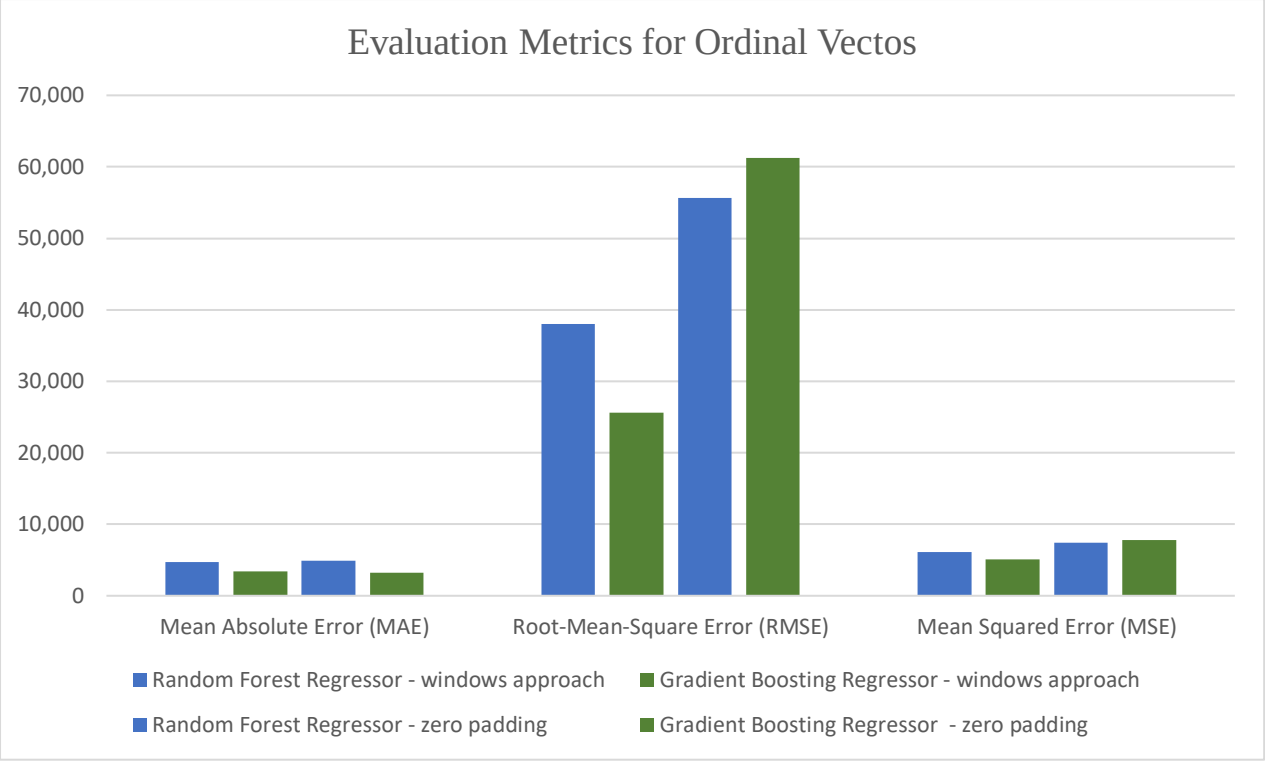
5.6.1.1 Ordinal Vectors

Ordinal encoding is to encode each nucleotide characters as an ordinal value. For example, “ATGC” becomes [0.25, 0.5, 0.75, 1.0]. Any other base such as “N” can be a 0. According to the paper *"Evaluation of Convolutionary Neural Networks Modeling of DNA Sequences using Ordinal versus one-hot Encoding Method"* by Allen Chieng, Hoon Choong and Nung Kion Lee, was shown that this way of encoding of the nucleotides characters worked well.[40]

Table 7: Performance of the algorithms using the window approach and the zero padding.

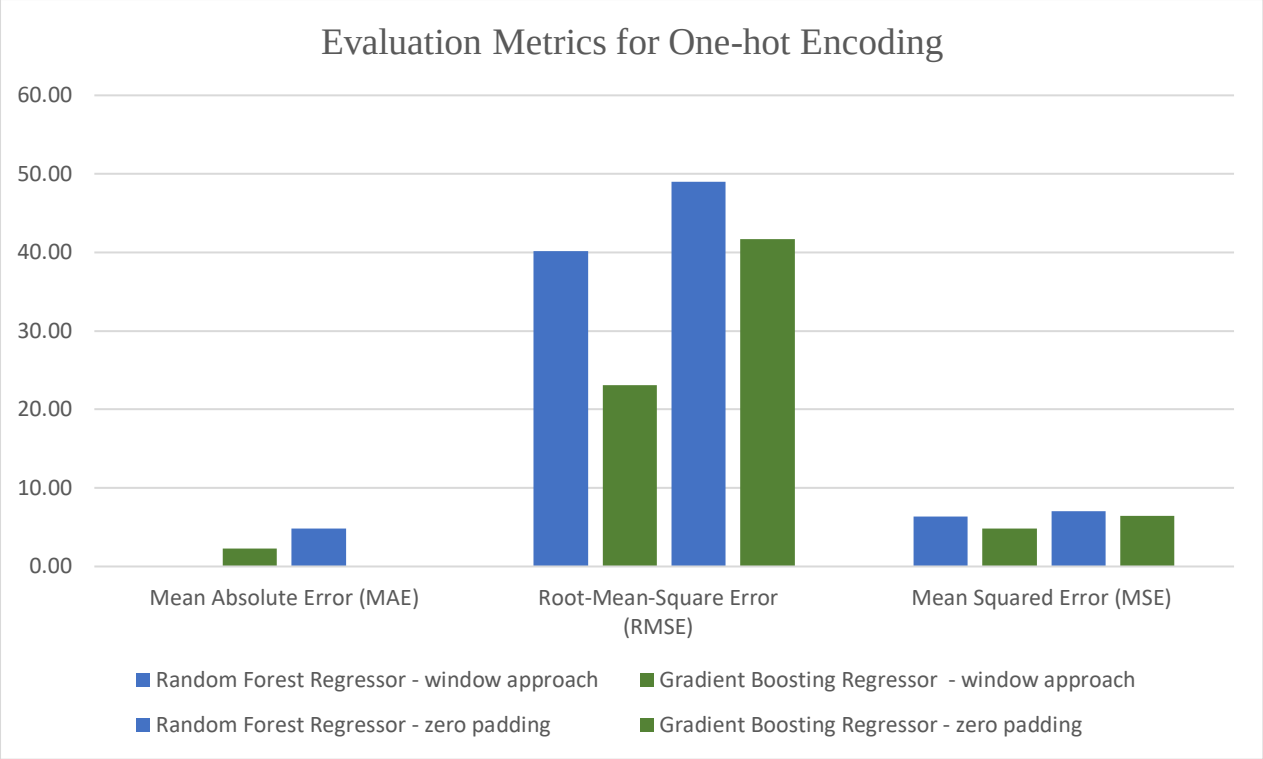
	Vectorized sequences – window approach		**Vectorized sequences – zero padding**	
Metrics	**Random Forest Regressor**	**Gradient Boosting Regressor**	**Random Forest Regressor**	**Gradient Boosting Regressor**

Mean Absolute Error (MAE)	4.731	3.431	4.940	3.202
Root-Mean-Square Error (RMSE)	37.979	25.567	55.647	61.254
Mean Squared Error (MSE)	6.162	5.056	7.459	7.826



5.6.1.2 One hot encoding

One-hot encoding is widely used in deep learning methods and lends itself well to algorithms like convolutional neural networks. In this example, “ATGC” would become [0,0,0,1], [0,0,1,0], [0,1,0,0], [1,0,0,0]. And these one-hot encoded vectors can either be concatenated or turned into 2 dimensional arrays.[41]



5.6.1.3 K-mer counting

A problem that remains is that none of these approaches lead to uniform length vectors, and that is a prerequisite to feed data to a classification or regression algorithm. For the ordinal vector approach, you need to resort to stuff like truncating sequences or padding for "n" or "0 "to get uniform length vectors.

DNA and protein sequences can be interpreted as the language of creation, metaphorically. The language encodes instructions, as well as work for the molecules present in all types of life.

The comparison of sequence language continues with the genome as the terms text, subsequences (genes and gene families) are sentences and chapters, k-mers and peptides (motifs), and nucleotide bases and amino acids are the alphabets.

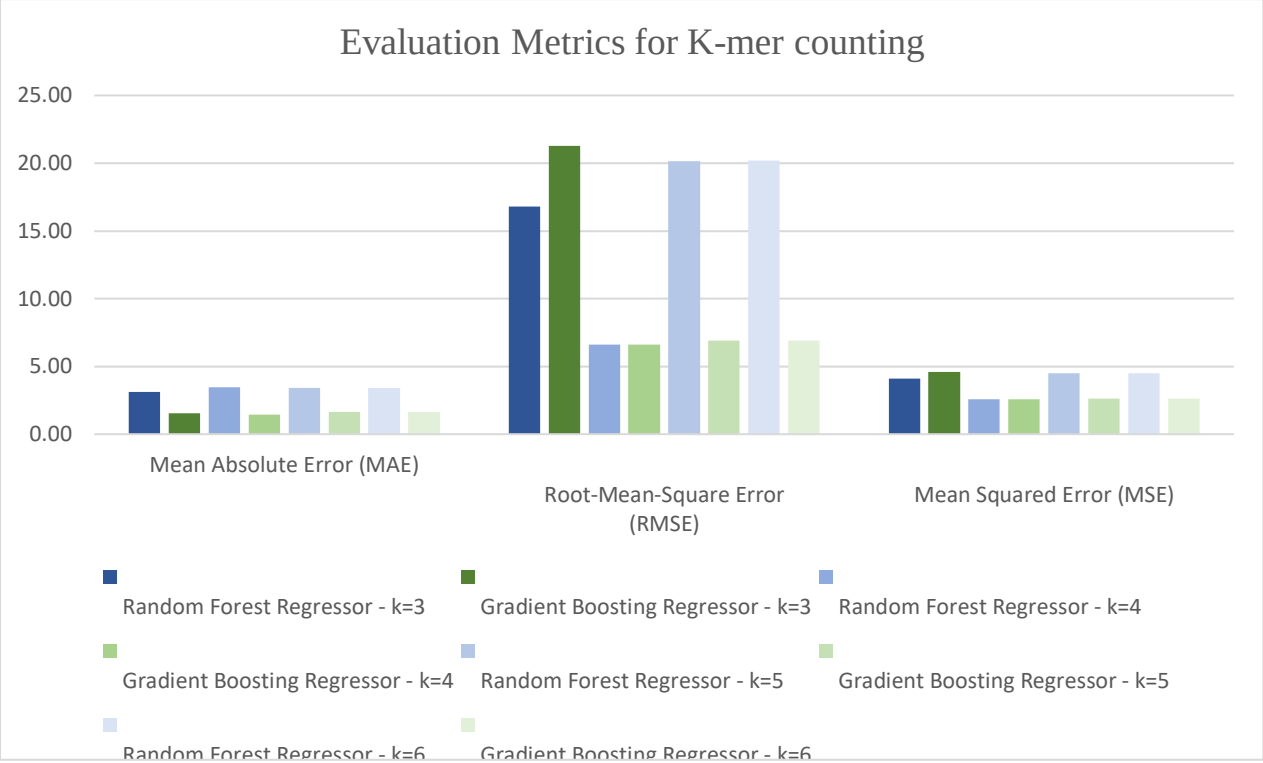
The method that was used takes the long biological sequence first and splits it into k-mer length overlapping “words”. For example, if “words” of length 6 (hexamers) are used, “AGACTTCT” becomes: ‘AGACTT’, ‘GACTT’, ‘ACTTCT’.

In genomics such types of manipulations as are referred as "k-mer counting," or counting each possible k-mer sequence occurrences. Specialized tools are available for this but the natural language processing tools from Python make it simple.[41]

For training thee algorithms the dataset was split at a ratio of 70/30 for training and testing respectively. Grid search was used to find the best parameters for both the Random Forest Regressor and the Gradient Boosting Regressor. ***(Error! Reference source not found.)***

For the purpose of finding the optimal performance for both the algorithms used, we tested the algorithms with k-mers of length 3,4,5 and 6. After creating the k-mers along with the corresponding vocabulary, the count vectorizing module was used from Scikit-Learn Natural Language Processing library.

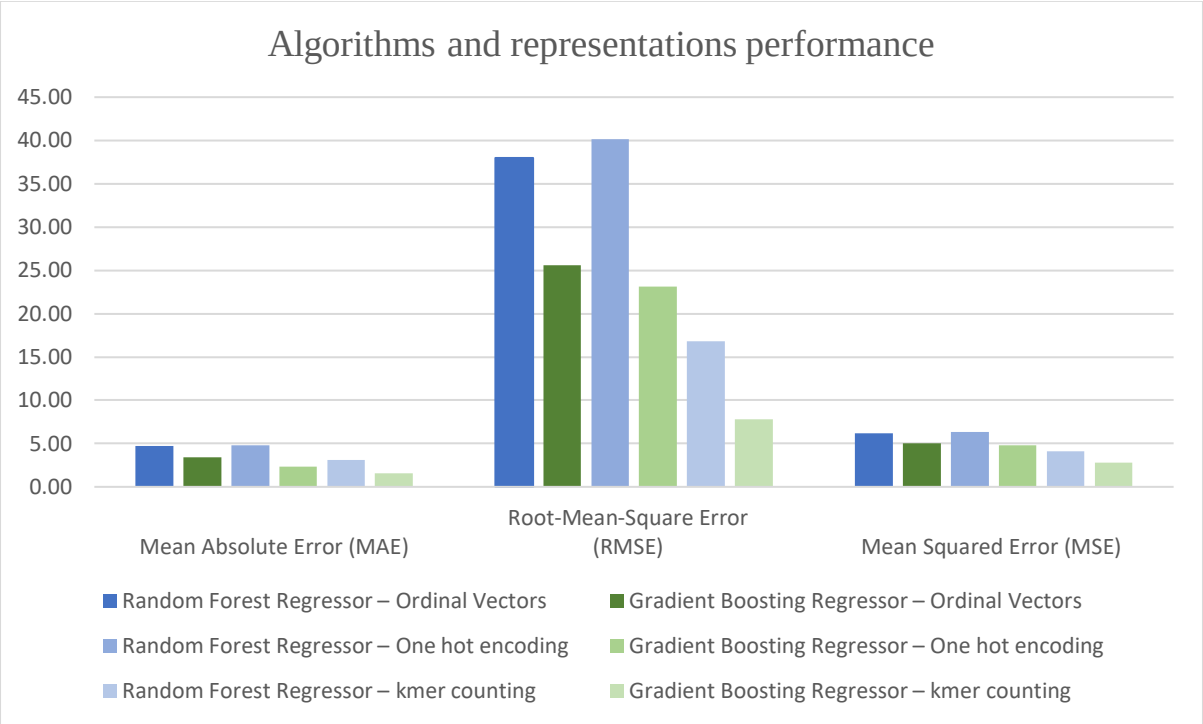
The last step before training the algorithms was to create vectors that had a dimensionality equal to the size of the vocabulary created. If the text data featured that vocab word, a one was put in that dimension. Every time that word was encountered again, the count would increase, leaving 0s everywhere the word was not found even once.



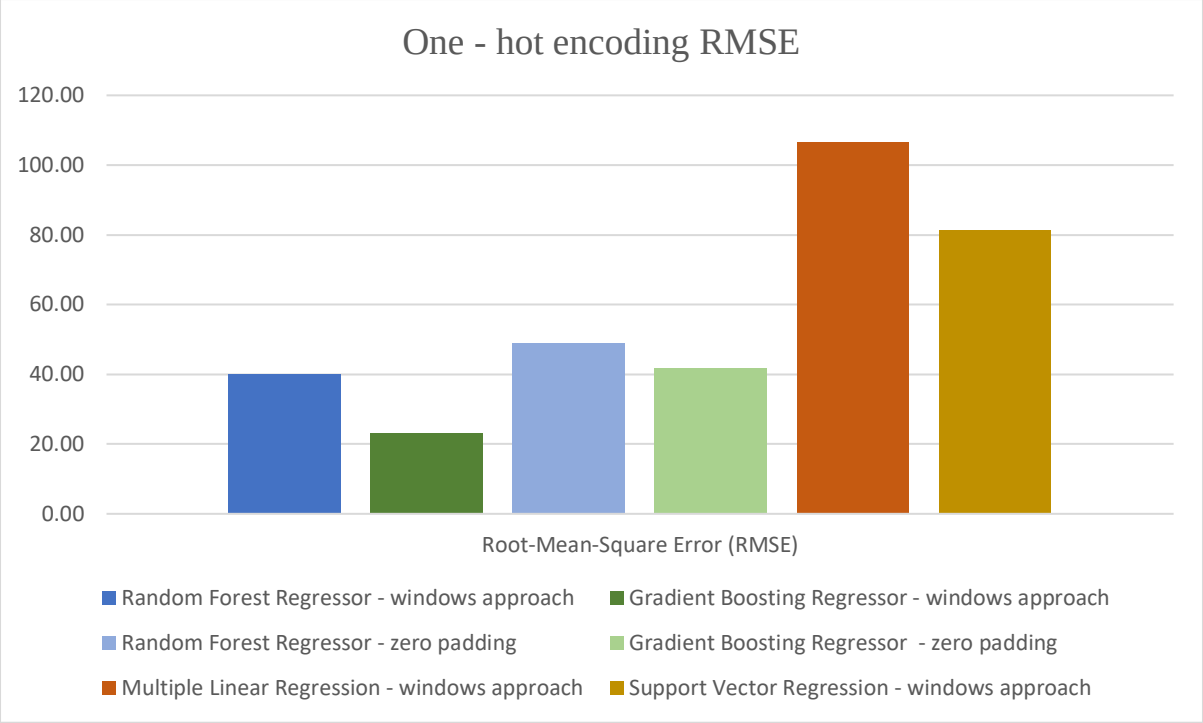
6 Experimental results

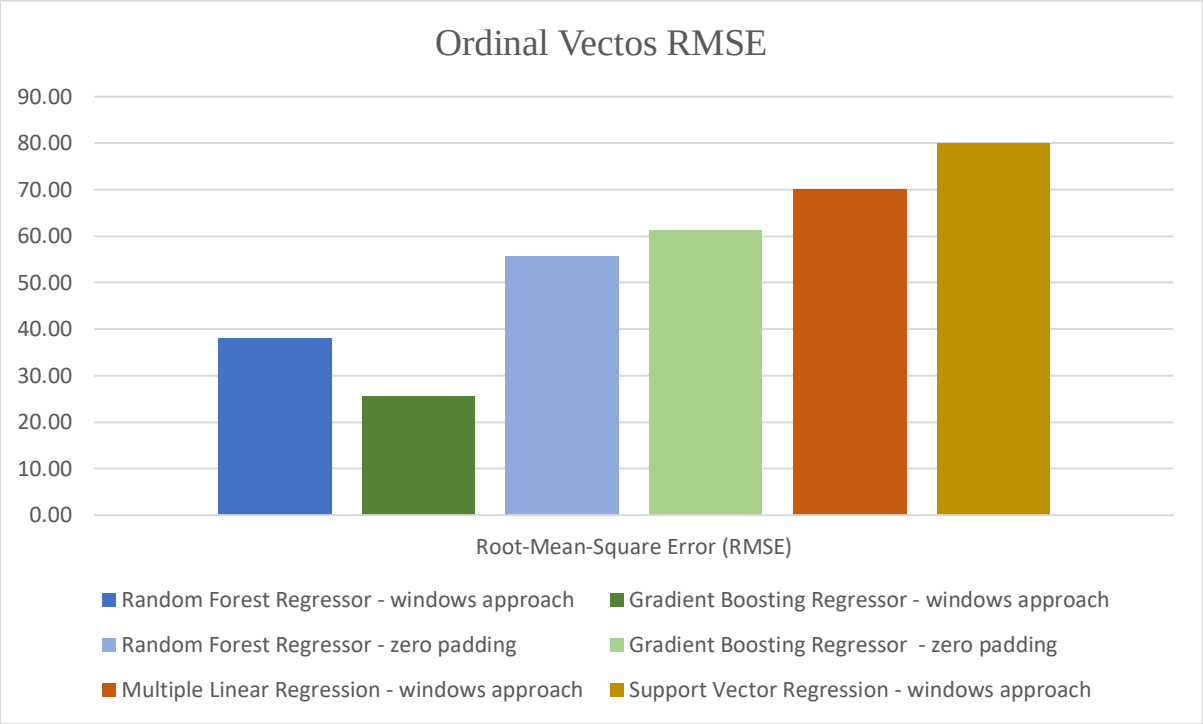
6.1 Algorithms performance

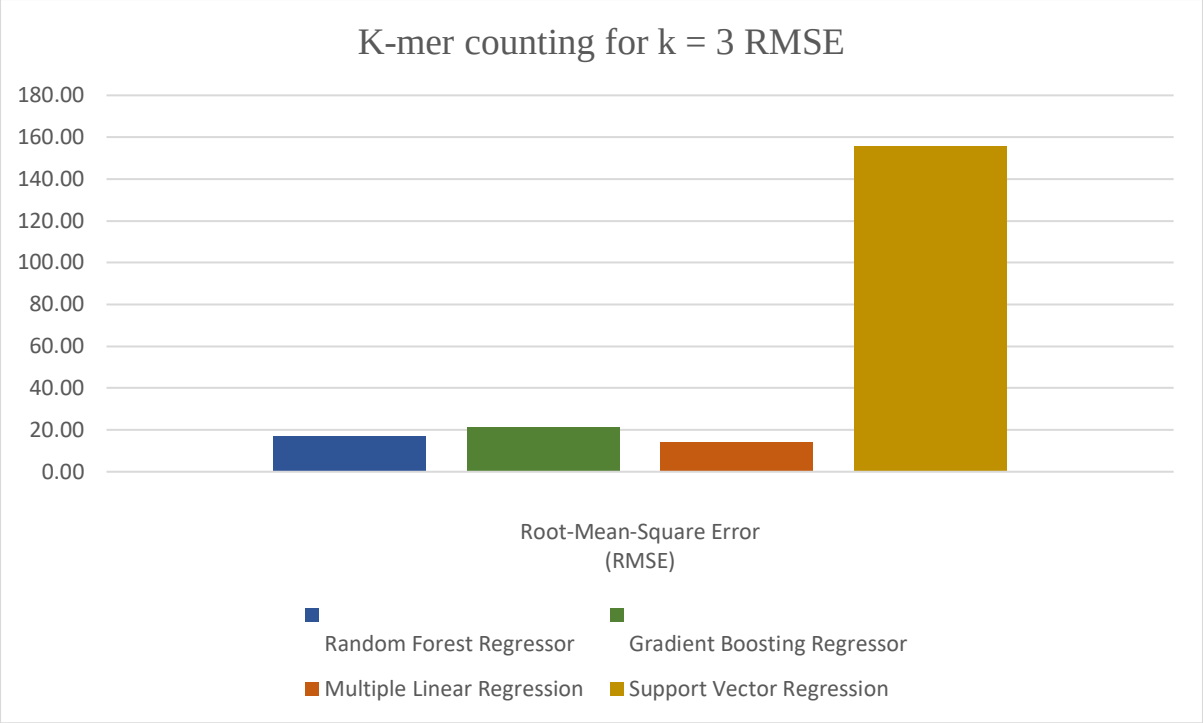
In general, Gradient Boosting Regressor had the best performance on the data in comparison with the Random Forest Regressor, regardless of the metric or the number of k. For the window method, Multiple Linear Regression and Support Vector Regression had the worst performance on the dataset. Multiple Linear Regression showed to have the best overall performance for k-mers, and specifically for k=3.

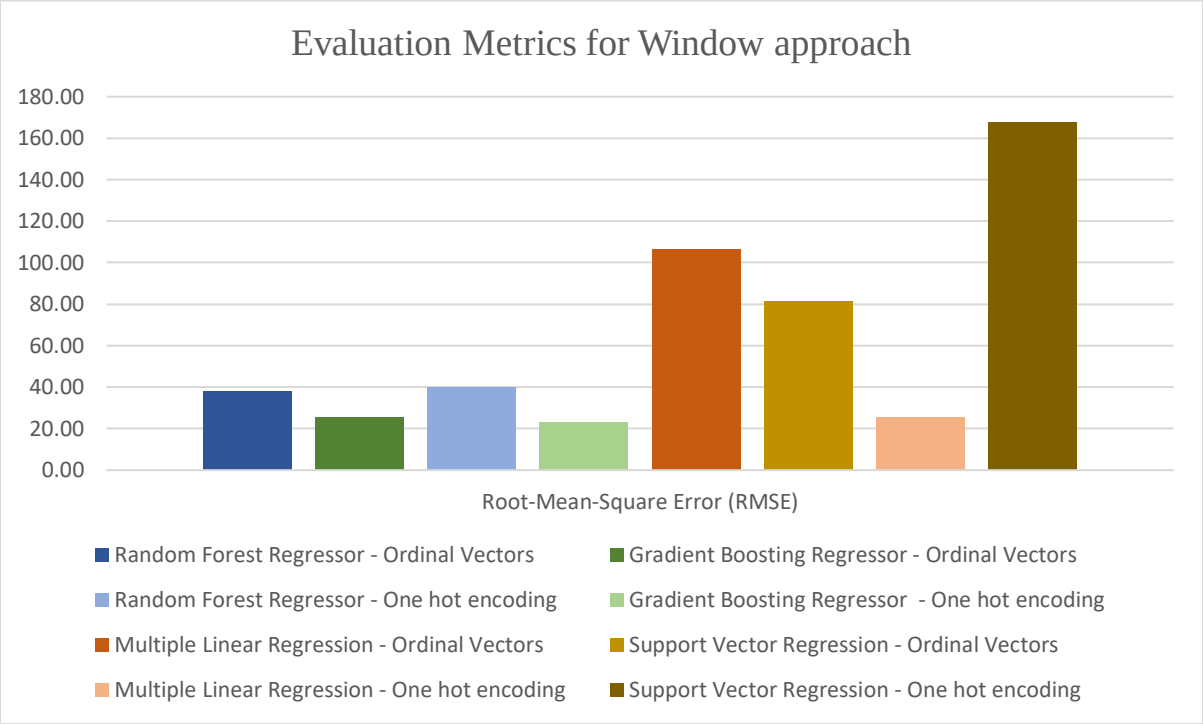


Also, from the above plot we could see that regardless of the algorithm used, k-mer counting appeared to have the best performance.


One - hot encoding RMSE
120.00
100.00
80.00
60.00
40.00
20.00
0.00
Root-Mean-Square Error (RMSE)
Random Forest Regressor - windows approach
Gradient Boosting Regressor - windows approach
Random Forest Regressor - zero padding
Gradient Boosting Regressor - zero padding
Multiple Linear Regression - windows approach
Support Vector Regression - windows approach


Ordinal Vectos RMSE
90.00
80.00
70.00
60.00
50.00
40.00
30.00
20.00
10.00
0.00
Root-Mean-Square Error (RMSE)
Random Forest Regressor - windows approach
Gradient Boosting Regressor - windows approach
Random Forest Regressor - zero padding
Gradient Boosting Regressor - zero padding
Multiple Linear Regression - windows approach
Support Vector Regression - windows approach


K-mer counting for k = 3 RMSE
180.00
160.00
140.00
120.00
100.00
80.00
60.00
40.00
20.00
0.00
Root-Mean-Square Error
(RMSE)
Random Forest Regressor
Gradient Boosting Regressor
Multiple Linear Regression
Support Vector Regression


Evaluation Metrics for Window approach
180.00
160.00
140.00
120.00
100.00
80.00
60.00
40.00
20.00
0.00
Root-Mean-Square Error (RMSE)
Random Forest Regressor - Ordinal Vectors
Gradient Boosting Regressor - Ordinal Vectors
Random Forest Regressor - One hot encoding
Gradient Boosting Regressor - One hot encoding
Multiple Linear Regression - Ordinal Vectors
Support Vector Regression - Ordinal Vectors
Multiple Linear Regression - One hot encoding
Support Vector Regression - One hot encoding

7 Conclusions and future work

In the current diploma thesis, we tried to create pegRNAs for use in prime editing, for the potential treatment of Duchene Muscular Dystrophy, and more specifically the mutation at exon 44 described in *Yi-Li Min et. al*. We took the protospacers that were used in CRISPR experiments and found through an extensive bibliographic research that the same protospacers are part of the pegRNAs used in prime editing. Also, we suggested an automated way of creating pegRNAs and we tried to evaluate them with machine learning algorithms.

As we mentioned in previous sections, we trained the algorithms on the data for HBB and HEXA provided to us by *Anzalone et. al.* The same algorithms were tested on the produced pegRNAs, in order to find those that would result in high biological efficiency. The most promising pegRNAs for both the case of insertion and deletion were kept. Likewise, the case that we studied for HBB and HEXA, we kept the pegRNAs that its mean of correct edit exceeded the scaffold of 14%.

As shown in the previous section multiple algorithms were tested as well as three approaches of encoding the data. Of all the algorithms tested, the Random Forest Regressor using the kmers approach, showed the best overall performance in predicting the efficiency of the pegRNAs.

We suggest that the current thesis would help researchers to automate the creation and evaluation of pegRNAs even for other more complex diseases. Through our computational work we concluded that every protospacer used in CRISPR could be used as part of the pegRNA that could potentially be used in prime editing. Future researchers should bear in mind that the total length of each pegRNA should not exceed the number of 44 bps for insertion and 80 bps for deletions, and the first bp should not be a Cytosine as highlighted in *Anzalone et. al.* Also, we suggest that additional experiments should be made with data from other experiments using CRISPR gene-editing, utilizing machine learning in order to find the best candidates for transitioning from CRISPR to prime editing.

Appendix

Table 8: Complete list of produced pegRNAs for reframing the exon 45, insertion.

pegRNA	spacer	3’ extension (5' to 3')
DMD 1	CTTACAGGAACTCCAGGA	AACAGTTTGCCGCTGCCCAATGCCATCCTTGGAGTTC
DMD 2	CTTACAGGAACTCCAGGA	ACAGTTTGCCGCTGCCCAATGCCATCCTTGGAGTTC
DMD 3	CTTACAGGAACTCCAGGA	CAGTTTGCCGCTGCCCAATGCCATCCTTGGAGTTC
DMD 4	CTTACAGGAACTCCAGGA	AGTTTGCCGCTGCCCAATGCCATCCTTGGAGTTC
DMD 5	CTTACAGGAACTCCAGGA	GTTTGCCGCTGCCCAATGCCATCCTTGGAGTTC
DMD 6	CTTACAGGAACTCCAGGA	TTTGCCGCTGCCCAATGCCATCCTTGGAGTTC
DMD 7	CTTACAGGAACTCCAGGA	TTGCCGCTGCCCAATGCCATCCTTGGAGTTC
DMD 8	CTTACAGGAACTCCAGGA	TGCCGCTGCCCAATGCCATCCTTGGAGTTC
DMD 9	CTTACAGGAACTCCAGGA	GCCGCTGCCCAATGCCATCCTTGGAGTTC
DMD 10	CTTACAGGAACTCCAGGA	CCGCTGCCCAATGCCATCCTTGGAGTTC
DMD 11	CTTACAGGAACTCCAGGA	CGCTGCCCAATGCCATCCTTGGAGTTC
DMD 12	CTTACAGGAACTCCAGGA	GCTGCCCAATGCCATCCTTGGAGTTC
DMD 13	CTTACAGGAACTCCAGGA	CTGCCCAATGCCATCCTTGGAGTTC
DMD 14	CTTACAGGAACTCCAGGA	TGCCCAATGCCATCCTTGGAGTTC
DMD 15	CTTACAGGAACTCCAGGA	GCCCAATGCCATCCTTGGAGTTC
DMD 16	CTTACAGGAACTCCAGGA	CCCAATGCCATCCTTGGAGTTC
DMD 17	CTTACAGGAACTCCAGGA	CCAATGCCATCCTTGGAGTTC
DMD 18	CTTACAGGAACTCCAGGA	CAATGCCATCCTTGGAGTTC
DMD 19	CTTACAGGAACTCCAGGA	AATGCCATCCTTGGAGTTC
DMD 20	CTTACAGGAACTCCAGGA	ATGCCATCCTTGGAGTTC
DMD 21	CTTACAGGAACTCCAGGA	TGCCATCCTTGGAGTTC
DMD 22	CTTACAGGAACTCCAGGA	GCCATCCTTGGAGTTC
DMD 23	CTTACAGGAACTCCAGGA	CCATCCTTGGAGTTC
DMD 24	CTTACAGGAACTCCAGGA	AACAGTTTGCCGCTGCCCAATGCCATCCTTGGAGTTCC
DMD 25	CTTACAGGAACTCCAGGA	ACAGTTTGCCGCTGCCCAATGCCATCCTTGGAGTTCC
DMD 26	CTTACAGGAACTCCAGGA	CAGTTTGCCGCTGCCCAATGCCATCCTTGGAGTTCC
DMD 27	CTTACAGGAACTCCAGGA	AGTTTGCCGCTGCCCAATGCCATCCTTGGAGTTCC
DMD 28	CTTACAGGAACTCCAGGA	GTTTGCCGCTGCCCAATGCCATCCTTGGAGTTCC
DMD 29	CTTACAGGAACTCCAGGA	TTTGCCGCTGCCCAATGCCATCCTTGGAGTTCC
DMD 30	CTTACAGGAACTCCAGGA	TTGCCGCTGCCCAATGCCATCCTTGGAGTTCC
DMD 31	CTTACAGGAACTCCAGGA	TGCCGCTGCCCAATGCCATCCTTGGAGTTCC
DMD 32	CTTACAGGAACTCCAGGA	GCCGCTGCCCAATGCCATCCTTGGAGTTCC
DMD 33	CTTACAGGAACTCCAGGA	CCGCTGCCCAATGCCATCCTTGGAGTTCC
DMD 34	CTTACAGGAACTCCAGGA	CGCTGCCCAATGCCATCCTTGGAGTTCC
DMD 35	CTTACAGGAACTCCAGGA	GCTGCCCAATGCCATCCTTGGAGTTCC
DMD 36	CTTACAGGAACTCCAGGA	CTGCCCAATGCCATCCTTGGAGTTCC
DMD 37	CTTACAGGAACTCCAGGA	TGCCCAATGCCATCCTTGGAGTTCC
DMD 38	CTTACAGGAACTCCAGGA	GCCCAATGCCATCCTTGGAGTTCC
DMD 39	CTTACAGGAACTCCAGGA	CCCAATGCCATCCTTGGAGTTCC
DMD 40	CTTACAGGAACTCCAGGA	CCAATGCCATCCTTGGAGTTCC
DMD 41	CTTACAGGAACTCCAGGA	CAATGCCATCCTTGGAGTTCC
DMD 42	CTTACAGGAACTCCAGGA	AATGCCATCCTTGGAGTTCC
DMD 43	CTTACAGGAACTCCAGGA	ATGCCATCCTTGGAGTTCC
DMD 44	CTTACAGGAACTCCAGGA	TGCCATCCTTGGAGTTCC
DMD 45	CTTACAGGAACTCCAGGA	GCCATCCTTGGAGTTCC

DMD 46	CTTACAGGAACTCCAGGA	CCATCCTTGGAGTTCC
DMD 47	CTTACAGGAACTCCAGGA	AACAGTTTGCCGCTGCCCAATGCCATCCTTGGAGTTCCT
DMD 48	CTTACAGGAACTCCAGGA	ACAGTTTGCCGCTGCCCAATGCCATCCTTGGAGTTCCT
DMD 49	CTTACAGGAACTCCAGGA	CAGTTTGCCGCTGCCCAATGCCATCCTTGGAGTTCCT
DMD 50	CTTACAGGAACTCCAGGA	AGTTTGCCGCTGCCCAATGCCATCCTTGGAGTTCCT
DMD 51	CTTACAGGAACTCCAGGA	GTTTGCCGCTGCCCAATGCCATCCTTGGAGTTCCT
DMD 52	CTTACAGGAACTCCAGGA	TTTGCCGCTGCCCAATGCCATCCTTGGAGTTCCT
DMD 53	CTTACAGGAACTCCAGGA	TTGCCGCTGCCCAATGCCATCCTTGGAGTTCCT
DMD 54	CTTACAGGAACTCCAGGA	TGCCGCTGCCCAATGCCATCCTTGGAGTTCCT
DMD 55	CTTACAGGAACTCCAGGA	GCCGCTGCCCAATGCCATCCTTGGAGTTCCT
DMD 56	CTTACAGGAACTCCAGGA	CCGCTGCCCAATGCCATCCTTGGAGTTCCT
DMD 57	CTTACAGGAACTCCAGGA	CGCTGCCCAATGCCATCCTTGGAGTTCCT
DMD 58	CTTACAGGAACTCCAGGA	GCTGCCCAATGCCATCCTTGGAGTTCCT
DMD 59	CTTACAGGAACTCCAGGA	CTGCCCAATGCCATCCTTGGAGTTCCT
DMD 60	CTTACAGGAACTCCAGGA	TGCCCAATGCCATCCTTGGAGTTCCT
DMD 61	CTTACAGGAACTCCAGGA	GCCCAATGCCATCCTTGGAGTTCCT
DMD 62	CTTACAGGAACTCCAGGA	CCCAATGCCATCCTTGGAGTTCCT
DMD 63	CTTACAGGAACTCCAGGA	CCAATGCCATCCTTGGAGTTCCT
DMD 64	CTTACAGGAACTCCAGGA	CAATGCCATCCTTGGAGTTCCT
DMD 65	CTTACAGGAACTCCAGGA	AATGCCATCCTTGGAGTTCCT
DMD 66	CTTACAGGAACTCCAGGA	ATGCCATCCTTGGAGTTCCT
DMD 67	CTTACAGGAACTCCAGGA	TGCCATCCTTGGAGTTCCT
DMD 68	CTTACAGGAACTCCAGGA	GCCATCCTTGGAGTTCCT
DMD 69	CTTACAGGAACTCCAGGA	CCATCCTTGGAGTTCCT
DMD 70	CTTACAGGAACTCCAGGA	AACAGTTTGCCGCTGCCCAATGCCATCCTTGGAGTTCCTG
DMD 71	CTTACAGGAACTCCAGGA	ACAGTTTGCCGCTGCCCAATGCCATCCTTGGAGTTCCTG
DMD 72	CTTACAGGAACTCCAGGA	CAGTTTGCCGCTGCCCAATGCCATCCTTGGAGTTCCTG
DMD 73	CTTACAGGAACTCCAGGA	AGTTTGCCGCTGCCCAATGCCATCCTTGGAGTTCCTG
DMD 74	CTTACAGGAACTCCAGGA	GTTTGCCGCTGCCCAATGCCATCCTTGGAGTTCCTG
DMD 75	CTTACAGGAACTCCAGGA	TTTGCCGCTGCCCAATGCCATCCTTGGAGTTCCTG
DMD 76	CTTACAGGAACTCCAGGA	TTGCCGCTGCCCAATGCCATCCTTGGAGTTCCTG
DMD 77	CTTACAGGAACTCCAGGA	TGCCGCTGCCCAATGCCATCCTTGGAGTTCCTG
DMD 78	CTTACAGGAACTCCAGGA	GCCGCTGCCCAATGCCATCCTTGGAGTTCCTG
DMD 79	CTTACAGGAACTCCAGGA	CCGCTGCCCAATGCCATCCTTGGAGTTCCTG
DMD 80	CTTACAGGAACTCCAGGA	CGCTGCCCAATGCCATCCTTGGAGTTCCTG
DMD 81	CTTACAGGAACTCCAGGA	GCTGCCCAATGCCATCCTTGGAGTTCCTG
DMD 82	CTTACAGGAACTCCAGGA	CTGCCCAATGCCATCCTTGGAGTTCCTG
DMD 83	CTTACAGGAACTCCAGGA	TGCCCAATGCCATCCTTGGAGTTCCTG
DMD 84	CTTACAGGAACTCCAGGA	GCCCAATGCCATCCTTGGAGTTCCTG
DMD 85	CTTACAGGAACTCCAGGA	CCCAATGCCATCCTTGGAGTTCCTG
DMD 86	CTTACAGGAACTCCAGGA	CCAATGCCATCCTTGGAGTTCCTG
DMD 87	CTTACAGGAACTCCAGGA	CAATGCCATCCTTGGAGTTCCTG
DMD 88	CTTACAGGAACTCCAGGA	AATGCCATCCTTGGAGTTCCTG
DMD 89	CTTACAGGAACTCCAGGA	ATGCCATCCTTGGAGTTCCTG
DMD 90	CTTACAGGAACTCCAGGA	TGCCATCCTTGGAGTTCCTG
DMD 91	CTTACAGGAACTCCAGGA	GCCATCCTTGGAGTTCCTG
DMD 92	CTTACAGGAACTCCAGGA	CCATCCTTGGAGTTCCTG
DMD 93	CTTACAGGAACTCCAGGA	AACAGTTTGCCGCTGCCCAATGCCATCCTTGGAGTTCCTGT
DMD 94	CTTACAGGAACTCCAGGA	ACAGTTTGCCGCTGCCCAATGCCATCCTTGGAGTTCCTGT
DMD 95	CTTACAGGAACTCCAGGA	CAGTTTGCCGCTGCCCAATGCCATCCTTGGAGTTCCTGT
DMD 96	CTTACAGGAACTCCAGGA	AGTTTGCCGCTGCCCAATGCCATCCTTGGAGTTCCTGT

DMD 97	CTTACAGGAACTCCAGGA	GTTTGCCGCTGCCCAATGCCATCCTTGGAGTTCCTGT
DMD 98	CTTACAGGAACTCCAGGA	TTTGCCGCTGCCCAATGCCATCCTTGGAGTTCCTGT
DMD 99	CTTACAGGAACTCCAGGA	TTGCCGCTGCCCAATGCCATCCTTGGAGTTCCTGT
DMD 100	CTTACAGGAACTCCAGGA	TGCCGCTGCCCAATGCCATCCTTGGAGTTCCTGT
DMD 101	CTTACAGGAACTCCAGGA	GCCGCTGCCCAATGCCATCCTTGGAGTTCCTGT
DMD 102	CTTACAGGAACTCCAGGA	CCGCTGCCCAATGCCATCCTTGGAGTTCCTGT
DMD 103	CTTACAGGAACTCCAGGA	CGCTGCCCAATGCCATCCTTGGAGTTCCTGT
DMD 104	CTTACAGGAACTCCAGGA	GCTGCCCAATGCCATCCTTGGAGTTCCTGT
DMD 105	CTTACAGGAACTCCAGGA	CTGCCCAATGCCATCCTTGGAGTTCCTGT
DMD 106	CTTACAGGAACTCCAGGA	TGCCCAATGCCATCCTTGGAGTTCCTGT
DMD 107	CTTACAGGAACTCCAGGA	GCCCAATGCCATCCTTGGAGTTCCTGT
DMD 108	CTTACAGGAACTCCAGGA	CCCAATGCCATCCTTGGAGTTCCTGT
DMD 109	CTTACAGGAACTCCAGGA	CCAATGCCATCCTTGGAGTTCCTGT
DMD 110	CTTACAGGAACTCCAGGA	CAATGCCATCCTTGGAGTTCCTGT
DMD 111	CTTACAGGAACTCCAGGA	AATGCCATCCTTGGAGTTCCTGT
DMD 112	CTTACAGGAACTCCAGGA	ATGCCATCCTTGGAGTTCCTGT
DMD 113	CTTACAGGAACTCCAGGA	TGCCATCCTTGGAGTTCCTGT
DMD 114	CTTACAGGAACTCCAGGA	GCCATCCTTGGAGTTCCTGT
DMD 115	CTTACAGGAACTCCAGGA	CCATCCTTGGAGTTCCTGT
DMD 116	CTTACAGGAACTCCAGGA	AACAGTTTGCCGCTGCCCAATGCCATCCTTGGAGTTCCTGTA
DMD 117	CTTACAGGAACTCCAGGA	ACAGTTTGCCGCTGCCCAATGCCATCCTTGGAGTTCCTGTA
DMD 118	CTTACAGGAACTCCAGGA	CAGTTTGCCGCTGCCCAATGCCATCCTTGGAGTTCCTGTA
DMD 119	CTTACAGGAACTCCAGGA	AGTTTGCCGCTGCCCAATGCCATCCTTGGAGTTCCTGTA
DMD 120	CTTACAGGAACTCCAGGA	GTTTGCCGCTGCCCAATGCCATCCTTGGAGTTCCTGTA
DMD 121	CTTACAGGAACTCCAGGA	TTTGCCGCTGCCCAATGCCATCCTTGGAGTTCCTGTA
DMD 122	CTTACAGGAACTCCAGGA	TTGCCGCTGCCCAATGCCATCCTTGGAGTTCCTGTA
DMD 123	CTTACAGGAACTCCAGGA	TGCCGCTGCCCAATGCCATCCTTGGAGTTCCTGTA
DMD 124	CTTACAGGAACTCCAGGA	GCCGCTGCCCAATGCCATCCTTGGAGTTCCTGTA
DMD 125	CTTACAGGAACTCCAGGA	CCGCTGCCCAATGCCATCCTTGGAGTTCCTGTA
DMD 126	CTTACAGGAACTCCAGGA	CGCTGCCCAATGCCATCCTTGGAGTTCCTGTA
DMD 127	CTTACAGGAACTCCAGGA	GCTGCCCAATGCCATCCTTGGAGTTCCTGTA
DMD 128	CTTACAGGAACTCCAGGA	CTGCCCAATGCCATCCTTGGAGTTCCTGTA
DMD 129	CTTACAGGAACTCCAGGA	TGCCCAATGCCATCCTTGGAGTTCCTGTA
DMD 130	CTTACAGGAACTCCAGGA	GCCCAATGCCATCCTTGGAGTTCCTGTA
DMD 131	CTTACAGGAACTCCAGGA	CCCAATGCCATCCTTGGAGTTCCTGTA
DMD 132	CTTACAGGAACTCCAGGA	CCAATGCCATCCTTGGAGTTCCTGTA
DMD 133	CTTACAGGAACTCCAGGA	CAATGCCATCCTTGGAGTTCCTGTA
DMD 134	CTTACAGGAACTCCAGGA	AATGCCATCCTTGGAGTTCCTGTA
DMD 135	CTTACAGGAACTCCAGGA	ATGCCATCCTTGGAGTTCCTGTA
DMD 136	CTTACAGGAACTCCAGGA	TGCCATCCTTGGAGTTCCTGTA
DMD 137	CTTACAGGAACTCCAGGA	GCCATCCTTGGAGTTCCTGTA
DMD 138	CTTACAGGAACTCCAGGA	CCATCCTTGGAGTTCCTGTA
DMD 139	CTTACAGGAACTCCAGGA	AACAGTTTGCCGCTGCCCAATGCCATCCTTGGAGTTCCTGTAA
DMD 140	CTTACAGGAACTCCAGGA	ACAGTTTGCCGCTGCCCAATGCCATCCTTGGAGTTCCTGTAA
DMD 141	CTTACAGGAACTCCAGGA	CAGTTTGCCGCTGCCCAATGCCATCCTTGGAGTTCCTGTAA
DMD 142	CTTACAGGAACTCCAGGA	AGTTTGCCGCTGCCCAATGCCATCCTTGGAGTTCCTGTAA
DMD 143	CTTACAGGAACTCCAGGA	GTTTGCCGCTGCCCAATGCCATCCTTGGAGTTCCTGTAA
DMD 144	CTTACAGGAACTCCAGGA	TTTGCCGCTGCCCAATGCCATCCTTGGAGTTCCTGTAA
DMD 145	CTTACAGGAACTCCAGGA	TTGCCGCTGCCCAATGCCATCCTTGGAGTTCCTGTAA
DMD 146	CTTACAGGAACTCCAGGA	TGCCGCTGCCCAATGCCATCCTTGGAGTTCCTGTAA
DMD 147	CTTACAGGAACTCCAGGA	GCCGCTGCCCAATGCCATCCTTGGAGTTCCTGTAA

DMD 148	CTTACAGGAACTCCAGGA	CCGCTGCCCAATGCCATCCTTGGAGTTCCTGTAA
DMD 149	CTTACAGGAACTCCAGGA	CGCTGCCCAATGCCATCCTTGGAGTTCCTGTAA
DMD 150	CTTACAGGAACTCCAGGA	GCTGCCCAATGCCATCCTTGGAGTTCCTGTAA
DMD 151	CTTACAGGAACTCCAGGA	CTGCCCAATGCCATCCTTGGAGTTCCTGTAA
DMD 152	CTTACAGGAACTCCAGGA	TGCCCAATGCCATCCTTGGAGTTCCTGTAA
DMD 153	CTTACAGGAACTCCAGGA	GCCCAATGCCATCCTTGGAGTTCCTGTAA
DMD 154	CTTACAGGAACTCCAGGA	CCCAATGCCATCCTTGGAGTTCCTGTAA
DMD 155	CTTACAGGAACTCCAGGA	CCAATGCCATCCTTGGAGTTCCTGTAA
DMD 156	CTTACAGGAACTCCAGGA	CAATGCCATCCTTGGAGTTCCTGTAA
DMD 157	CTTACAGGAACTCCAGGA	AATGCCATCCTTGGAGTTCCTGTAA
DMD 158	CTTACAGGAACTCCAGGA	ATGCCATCCTTGGAGTTCCTGTAA
DMD 159	CTTACAGGAACTCCAGGA	TGCCATCCTTGGAGTTCCTGTAA
DMD 160	CTTACAGGAACTCCAGGA	GCCATCCTTGGAGTTCCTGTAA
DMD 161	CTTACAGGAACTCCAGGA	CCATCCTTGGAGTTCCTGTAA
DMD 162	CTTACAGGAACTCCAGGA	AACAGTTTGCCGCTGCCCAATGCCATCCTTGGAGTTCCTGTAAG
DMD 163	CTTACAGGAACTCCAGGA	ACAGTTTGCCGCTGCCCAATGCCATCCTTGGAGTTCCTGTAAG
DMD 164	CTTACAGGAACTCCAGGA	CAGTTTGCCGCTGCCCAATGCCATCCTTGGAGTTCCTGTAAG
DMD 165	CTTACAGGAACTCCAGGA	AGTTTGCCGCTGCCCAATGCCATCCTTGGAGTTCCTGTAAG
DMD 166	CTTACAGGAACTCCAGGA	GTTTGCCGCTGCCCAATGCCATCCTTGGAGTTCCTGTAAG
DMD 167	CTTACAGGAACTCCAGGA	TTTGCCGCTGCCCAATGCCATCCTTGGAGTTCCTGTAAG
DMD 168	CTTACAGGAACTCCAGGA	TTGCCGCTGCCCAATGCCATCCTTGGAGTTCCTGTAAG
DMD 169	CTTACAGGAACTCCAGGA	TGCCGCTGCCCAATGCCATCCTTGGAGTTCCTGTAAG
DMD 170	CTTACAGGAACTCCAGGA	GCCGCTGCCCAATGCCATCCTTGGAGTTCCTGTAAG
DMD 171	CTTACAGGAACTCCAGGA	CCGCTGCCCAATGCCATCCTTGGAGTTCCTGTAAG
DMD 172	CTTACAGGAACTCCAGGA	CGCTGCCCAATGCCATCCTTGGAGTTCCTGTAAG
DMD 173	CTTACAGGAACTCCAGGA	GCTGCCCAATGCCATCCTTGGAGTTCCTGTAAG
DMD 174	CTTACAGGAACTCCAGGA	CTGCCCAATGCCATCCTTGGAGTTCCTGTAAG
DMD 175	CTTACAGGAACTCCAGGA	TGCCCAATGCCATCCTTGGAGTTCCTGTAAG
DMD 176	CTTACAGGAACTCCAGGA	GCCCAATGCCATCCTTGGAGTTCCTGTAAG
DMD 177	CTTACAGGAACTCCAGGA	CCCAATGCCATCCTTGGAGTTCCTGTAAG
DMD 178	CTTACAGGAACTCCAGGA	CCAATGCCATCCTTGGAGTTCCTGTAAG
DMD 179	CTTACAGGAACTCCAGGA	CAATGCCATCCTTGGAGTTCCTGTAAG
DMD 180	CTTACAGGAACTCCAGGA	AATGCCATCCTTGGAGTTCCTGTAAG
DMD 181	CTTACAGGAACTCCAGGA	ATGCCATCCTTGGAGTTCCTGTAAG
DMD 182	CTTACAGGAACTCCAGGA	TGCCATCCTTGGAGTTCCTGTAAG
DMD 183	CTTACAGGAACTCCAGGA	GCCATCCTTGGAGTTCCTGTAAG
DMD 184	CTTACAGGAACTCCAGGA	CCATCCTTGGAGTTCCTGTAAG

Table 9: Complete list of produced pegRNAs for reframing exon 45, deletion.

pegRNA	**spacer**	**3’ extension (5' to 3')**
DMD 1	CTTACAGGAACTCCAGGA	ACAACAGTTTGCCGCTGCCCAATGCCATTGGAGTTC
DMD 2	CTTACAGGAACTCCAGGA	CAACAGTTTGCCGCTGCCCAATGCCATTGGAGTTC
DMD 3	CTTACAGGAACTCCAGGA	AACAGTTTGCCGCTGCCCAATGCCATTGGAGTTC
DMD 4	CTTACAGGAACTCCAGGA	ACAGTTTGCCGCTGCCCAATGCCATTGGAGTTC
DMD 5	CTTACAGGAACTCCAGGA	CAGTTTGCCGCTGCCCAATGCCATTGGAGTTC
DMD 6	CTTACAGGAACTCCAGGA	AGTTTGCCGCTGCCCAATGCCATTGGAGTTC
DMD 7	CTTACAGGAACTCCAGGA	GTTTGCCGCTGCCCAATGCCATTGGAGTTC
DMD 8	CTTACAGGAACTCCAGGA	TTTGCCGCTGCCCAATGCCATTGGAGTTC
DMD 9	CTTACAGGAACTCCAGGA	TTGCCGCTGCCCAATGCCATTGGAGTTC
DMD 10	CTTACAGGAACTCCAGGA	TGCCGCTGCCCAATGCCATTGGAGTTC

DMD 11	CTTACAGGAACTCCAGGA	GCCGCTGCCCAATGCCATTGGAGTTC
DMD 12	CTTACAGGAACTCCAGGA	CCGCTGCCCAATGCCATTGGAGTTC
DMD 13	CTTACAGGAACTCCAGGA	CGCTGCCCAATGCCATTGGAGTTC
DMD 14	CTTACAGGAACTCCAGGA	GCTGCCCAATGCCATTGGAGTTC
DMD 15	CTTACAGGAACTCCAGGA	CTGCCCAATGCCATTGGAGTTC
DMD 16	CTTACAGGAACTCCAGGA	TGCCCAATGCCATTGGAGTTC
DMD 17	CTTACAGGAACTCCAGGA	GCCCAATGCCATTGGAGTTC
DMD 18	CTTACAGGAACTCCAGGA	CCCAATGCCATTGGAGTTC
DMD 19	CTTACAGGAACTCCAGGA	CCAATGCCATTGGAGTTC
DMD 20	CTTACAGGAACTCCAGGA	CAATGCCATTGGAGTTC
DMD 21	CTTACAGGAACTCCAGGA	AATGCCATTGGAGTTC
DMD 22	CTTACAGGAACTCCAGGA	ATGCCATTGGAGTTC
DMD 23	CTTACAGGAACTCCAGGA	TGCCATTGGAGTTC
DMD 24	CTTACAGGAACTCCAGGA	ACAACAGTTTGCCGCTGCCCAATGCCATTGGAGTTCC
DMD 25	CTTACAGGAACTCCAGGA	CAACAGTTTGCCGCTGCCCAATGCCATTGGAGTTCC
DMD 26	CTTACAGGAACTCCAGGA	AACAGTTTGCCGCTGCCCAATGCCATTGGAGTTCC
DMD 27	CTTACAGGAACTCCAGGA	ACAGTTTGCCGCTGCCCAATGCCATTGGAGTTCC
DMD 28	CTTACAGGAACTCCAGGA	CAGTTTGCCGCTGCCCAATGCCATTGGAGTTCC
DMD 29	CTTACAGGAACTCCAGGA	AGTTTGCCGCTGCCCAATGCCATTGGAGTTCC
DMD 30	CTTACAGGAACTCCAGGA	GTTTGCCGCTGCCCAATGCCATTGGAGTTCC
DMD 31	CTTACAGGAACTCCAGGA	TTTGCCGCTGCCCAATGCCATTGGAGTTCC
DMD 32	CTTACAGGAACTCCAGGA	TTGCCGCTGCCCAATGCCATTGGAGTTCC
DMD 33	CTTACAGGAACTCCAGGA	TGCCGCTGCCCAATGCCATTGGAGTTCC
DMD 34	CTTACAGGAACTCCAGGA	GCCGCTGCCCAATGCCATTGGAGTTCC
DMD 35	CTTACAGGAACTCCAGGA	CCGCTGCCCAATGCCATTGGAGTTCC
DMD 36	CTTACAGGAACTCCAGGA	CGCTGCCCAATGCCATTGGAGTTCC
DMD 37	CTTACAGGAACTCCAGGA	GCTGCCCAATGCCATTGGAGTTCC
DMD 38	CTTACAGGAACTCCAGGA	CTGCCCAATGCCATTGGAGTTCC
DMD 39	CTTACAGGAACTCCAGGA	TGCCCAATGCCATTGGAGTTCC
DMD 40	CTTACAGGAACTCCAGGA	GCCCAATGCCATTGGAGTTCC
DMD 41	CTTACAGGAACTCCAGGA	CCCAATGCCATTGGAGTTCC
DMD 42	CTTACAGGAACTCCAGGA	CCAATGCCATTGGAGTTCC
DMD 43	CTTACAGGAACTCCAGGA	CAATGCCATTGGAGTTCC
DMD 44	CTTACAGGAACTCCAGGA	AATGCCATTGGAGTTCC
DMD 45	CTTACAGGAACTCCAGGA	ATGCCATTGGAGTTCC
DMD 46	CTTACAGGAACTCCAGGA	TGCCATTGGAGTTCC
DMD 47	CTTACAGGAACTCCAGGA	ACAACAGTTTGCCGCTGCCCAATGCCATTGGAGTTCCT
DMD 48	CTTACAGGAACTCCAGGA	CAACAGTTTGCCGCTGCCCAATGCCATTGGAGTTCCT
DMD 49	CTTACAGGAACTCCAGGA	AACAGTTTGCCGCTGCCCAATGCCATTGGAGTTCCT
DMD 50	CTTACAGGAACTCCAGGA	ACAGTTTGCCGCTGCCCAATGCCATTGGAGTTCCT
DMD 51	CTTACAGGAACTCCAGGA	CAGTTTGCCGCTGCCCAATGCCATTGGAGTTCCT
DMD 52	CTTACAGGAACTCCAGGA	AGTTTGCCGCTGCCCAATGCCATTGGAGTTCCT
DMD 53	CTTACAGGAACTCCAGGA	GTTTGCCGCTGCCCAATGCCATTGGAGTTCCT
DMD 54	CTTACAGGAACTCCAGGA	TTTGCCGCTGCCCAATGCCATTGGAGTTCCT
DMD 55	CTTACAGGAACTCCAGGA	TTGCCGCTGCCCAATGCCATTGGAGTTCCT
DMD 56	CTTACAGGAACTCCAGGA	TGCCGCTGCCCAATGCCATTGGAGTTCCT
DMD 57	CTTACAGGAACTCCAGGA	GCCGCTGCCCAATGCCATTGGAGTTCCT
DMD 58	CTTACAGGAACTCCAGGA	CCGCTGCCCAATGCCATTGGAGTTCCT
DMD 59	CTTACAGGAACTCCAGGA	CGCTGCCCAATGCCATTGGAGTTCCT
DMD 60	CTTACAGGAACTCCAGGA	GCTGCCCAATGCCATTGGAGTTCCT
DMD 61	CTTACAGGAACTCCAGGA	CTGCCCAATGCCATTGGAGTTCCT

DMD 62	CTTACAGGAACTCCAGGA	TGCCCAATGCCATTGGAGTTCCT
DMD 63	CTTACAGGAACTCCAGGA	GCCCAATGCCATTGGAGTTCCT
DMD 64	CTTACAGGAACTCCAGGA	CCCAATGCCATTGGAGTTCCT
DMD 65	CTTACAGGAACTCCAGGA	CCAATGCCATTGGAGTTCCT
DMD 66	CTTACAGGAACTCCAGGA	CAATGCCATTGGAGTTCCT
DMD 67	CTTACAGGAACTCCAGGA	AATGCCATTGGAGTTCCT
DMD 68	CTTACAGGAACTCCAGGA	ATGCCATTGGAGTTCCT
DMD 69	CTTACAGGAACTCCAGGA	TGCCATTGGAGTTCCT
DMD 70	CTTACAGGAACTCCAGGA	ACAACAGTTTGCCGCTGCCCAATGCCATTGGAGTTCCTG
DMD 71	CTTACAGGAACTCCAGGA	CAACAGTTTGCCGCTGCCCAATGCCATTGGAGTTCCTG
DMD 72	CTTACAGGAACTCCAGGA	AACAGTTTGCCGCTGCCCAATGCCATTGGAGTTCCTG
DMD 73	CTTACAGGAACTCCAGGA	ACAGTTTGCCGCTGCCCAATGCCATTGGAGTTCCTG
DMD 74	CTTACAGGAACTCCAGGA	CAGTTTGCCGCTGCCCAATGCCATTGGAGTTCCTG
DMD 75	CTTACAGGAACTCCAGGA	AGTTTGCCGCTGCCCAATGCCATTGGAGTTCCTG
DMD 76	CTTACAGGAACTCCAGGA	GTTTGCCGCTGCCCAATGCCATTGGAGTTCCTG
DMD 77	CTTACAGGAACTCCAGGA	TTTGCCGCTGCCCAATGCCATTGGAGTTCCTG
DMD 78	CTTACAGGAACTCCAGGA	TTGCCGCTGCCCAATGCCATTGGAGTTCCTG
DMD 79	CTTACAGGAACTCCAGGA	TGCCGCTGCCCAATGCCATTGGAGTTCCTG
DMD 80	CTTACAGGAACTCCAGGA	GCCGCTGCCCAATGCCATTGGAGTTCCTG
DMD 81	CTTACAGGAACTCCAGGA	CCGCTGCCCAATGCCATTGGAGTTCCTG
DMD 82	CTTACAGGAACTCCAGGA	CGCTGCCCAATGCCATTGGAGTTCCTG
DMD 83	CTTACAGGAACTCCAGGA	GCTGCCCAATGCCATTGGAGTTCCTG
DMD 84	CTTACAGGAACTCCAGGA	CTGCCCAATGCCATTGGAGTTCCTG
DMD 85	CTTACAGGAACTCCAGGA	TGCCCAATGCCATTGGAGTTCCTG
DMD 86	CTTACAGGAACTCCAGGA	GCCCAATGCCATTGGAGTTCCTG
DMD 87	CTTACAGGAACTCCAGGA	CCCAATGCCATTGGAGTTCCTG
DMD 88	CTTACAGGAACTCCAGGA	CCAATGCCATTGGAGTTCCTG
DMD 89	CTTACAGGAACTCCAGGA	CAATGCCATTGGAGTTCCTG
DMD 90	CTTACAGGAACTCCAGGA	AATGCCATTGGAGTTCCTG
DMD 91	CTTACAGGAACTCCAGGA	ATGCCATTGGAGTTCCTG
DMD 92	CTTACAGGAACTCCAGGA	TGCCATTGGAGTTCCTG
DMD 93	CTTACAGGAACTCCAGGA	ACAACAGTTTGCCGCTGCCCAATGCCATTGGAGTTCCTGT
DMD 94	CTTACAGGAACTCCAGGA	CAACAGTTTGCCGCTGCCCAATGCCATTGGAGTTCCTGT
DMD 95	CTTACAGGAACTCCAGGA	AACAGTTTGCCGCTGCCCAATGCCATTGGAGTTCCTGT
DMD 96	CTTACAGGAACTCCAGGA	ACAGTTTGCCGCTGCCCAATGCCATTGGAGTTCCTGT
DMD 97	CTTACAGGAACTCCAGGA	CAGTTTGCCGCTGCCCAATGCCATTGGAGTTCCTGT
DMD 98	CTTACAGGAACTCCAGGA	AGTTTGCCGCTGCCCAATGCCATTGGAGTTCCTGT
DMD 99	CTTACAGGAACTCCAGGA	GTTTGCCGCTGCCCAATGCCATTGGAGTTCCTGT
DMD 100	CTTACAGGAACTCCAGGA	TTTGCCGCTGCCCAATGCCATTGGAGTTCCTGT
DMD 101	CTTACAGGAACTCCAGGA	TTGCCGCTGCCCAATGCCATTGGAGTTCCTGT
DMD 102	CTTACAGGAACTCCAGGA	TGCCGCTGCCCAATGCCATTGGAGTTCCTGT
DMD 103	CTTACAGGAACTCCAGGA	GCCGCTGCCCAATGCCATTGGAGTTCCTGT
DMD 104	CTTACAGGAACTCCAGGA	CCGCTGCCCAATGCCATTGGAGTTCCTGT
DMD 105	CTTACAGGAACTCCAGGA	CGCTGCCCAATGCCATTGGAGTTCCTGT
DMD 106	CTTACAGGAACTCCAGGA	GCTGCCCAATGCCATTGGAGTTCCTGT
DMD 107	CTTACAGGAACTCCAGGA	CTGCCCAATGCCATTGGAGTTCCTGT
DMD 108	CTTACAGGAACTCCAGGA	TGCCCAATGCCATTGGAGTTCCTGT
DMD 109	CTTACAGGAACTCCAGGA	GCCCAATGCCATTGGAGTTCCTGT
DMD 110	CTTACAGGAACTCCAGGA	CCCAATGCCATTGGAGTTCCTGT
DMD 111	CTTACAGGAACTCCAGGA	CCAATGCCATTGGAGTTCCTGT
DMD 112	CTTACAGGAACTCCAGGA	CAATGCCATTGGAGTTCCTGT

DMD 113	CTTACAGGAACTCCAGGA	AATGCCATTGGAGTTCCTGT
DMD 114	CTTACAGGAACTCCAGGA	ATGCCATTGGAGTTCCTGT
DMD 115	CTTACAGGAACTCCAGGA	TGCCATTGGAGTTCCTGT
DMD 116	CTTACAGGAACTCCAGGA	ACAACAGTTTGCCGCTGCCCAATGCCATTGGAGTTCCTGTA
DMD 117	CTTACAGGAACTCCAGGA	CAACAGTTTGCCGCTGCCCAATGCCATTGGAGTTCCTGTA
DMD 118	CTTACAGGAACTCCAGGA	AACAGTTTGCCGCTGCCCAATGCCATTGGAGTTCCTGTA
DMD 119	CTTACAGGAACTCCAGGA	ACAGTTTGCCGCTGCCCAATGCCATTGGAGTTCCTGTA
DMD 120	CTTACAGGAACTCCAGGA	CAGTTTGCCGCTGCCCAATGCCATTGGAGTTCCTGTA
DMD 121	CTTACAGGAACTCCAGGA	AGTTTGCCGCTGCCCAATGCCATTGGAGTTCCTGTA
DMD 122	CTTACAGGAACTCCAGGA	GTTTGCCGCTGCCCAATGCCATTGGAGTTCCTGTA
DMD 123	CTTACAGGAACTCCAGGA	TTTGCCGCTGCCCAATGCCATTGGAGTTCCTGTA
DMD 124	CTTACAGGAACTCCAGGA	TTGCCGCTGCCCAATGCCATTGGAGTTCCTGTA
DMD 125	CTTACAGGAACTCCAGGA	TGCCGCTGCCCAATGCCATTGGAGTTCCTGTA
DMD 126	CTTACAGGAACTCCAGGA	GCCGCTGCCCAATGCCATTGGAGTTCCTGTA
DMD 127	CTTACAGGAACTCCAGGA	CCGCTGCCCAATGCCATTGGAGTTCCTGTA
DMD 128	CTTACAGGAACTCCAGGA	CGCTGCCCAATGCCATTGGAGTTCCTGTA
DMD 129	CTTACAGGAACTCCAGGA	GCTGCCCAATGCCATTGGAGTTCCTGTA
DMD 130	CTTACAGGAACTCCAGGA	CTGCCCAATGCCATTGGAGTTCCTGTA
DMD 131	CTTACAGGAACTCCAGGA	TGCCCAATGCCATTGGAGTTCCTGTA
DMD 132	CTTACAGGAACTCCAGGA	GCCCAATGCCATTGGAGTTCCTGTA
DMD 133	CTTACAGGAACTCCAGGA	CCCAATGCCATTGGAGTTCCTGTA
DMD 134	CTTACAGGAACTCCAGGA	CCAATGCCATTGGAGTTCCTGTA
DMD 135	CTTACAGGAACTCCAGGA	CAATGCCATTGGAGTTCCTGTA
DMD 136	CTTACAGGAACTCCAGGA	AATGCCATTGGAGTTCCTGTA
DMD 137	CTTACAGGAACTCCAGGA	ATGCCATTGGAGTTCCTGTA
DMD 138	CTTACAGGAACTCCAGGA	TGCCATTGGAGTTCCTGTA
DMD 139	CTTACAGGAACTCCAGGA	ACAACAGTTTGCCGCTGCCCAATGCCATTGGAGTTCCTGTAA
DMD 140	CTTACAGGAACTCCAGGA	CAACAGTTTGCCGCTGCCCAATGCCATTGGAGTTCCTGTAA
DMD 141	CTTACAGGAACTCCAGGA	AACAGTTTGCCGCTGCCCAATGCCATTGGAGTTCCTGTAA
DMD 142	CTTACAGGAACTCCAGGA	ACAGTTTGCCGCTGCCCAATGCCATTGGAGTTCCTGTAA
DMD 143	CTTACAGGAACTCCAGGA	CAGTTTGCCGCTGCCCAATGCCATTGGAGTTCCTGTAA
DMD 144	CTTACAGGAACTCCAGGA	AGTTTGCCGCTGCCCAATGCCATTGGAGTTCCTGTAA
DMD 145	CTTACAGGAACTCCAGGA	GTTTGCCGCTGCCCAATGCCATTGGAGTTCCTGTAA
DMD 146	CTTACAGGAACTCCAGGA	TTTGCCGCTGCCCAATGCCATTGGAGTTCCTGTAA
DMD 147	CTTACAGGAACTCCAGGA	TTGCCGCTGCCCAATGCCATTGGAGTTCCTGTAA
DMD 148	CTTACAGGAACTCCAGGA	TGCCGCTGCCCAATGCCATTGGAGTTCCTGTAA
DMD 149	CTTACAGGAACTCCAGGA	GCCGCTGCCCAATGCCATTGGAGTTCCTGTAA
DMD 150	CTTACAGGAACTCCAGGA	CCGCTGCCCAATGCCATTGGAGTTCCTGTAA
DMD 151	CTTACAGGAACTCCAGGA	CGCTGCCCAATGCCATTGGAGTTCCTGTAA
DMD 152	CTTACAGGAACTCCAGGA	GCTGCCCAATGCCATTGGAGTTCCTGTAA
DMD 153	CTTACAGGAACTCCAGGA	CTGCCCAATGCCATTGGAGTTCCTGTAA
DMD 154	CTTACAGGAACTCCAGGA	TGCCCAATGCCATTGGAGTTCCTGTAA
DMD 155	CTTACAGGAACTCCAGGA	GCCCAATGCCATTGGAGTTCCTGTAA
DMD 156	CTTACAGGAACTCCAGGA	CCCAATGCCATTGGAGTTCCTGTAA
DMD 157	CTTACAGGAACTCCAGGA	CCAATGCCATTGGAGTTCCTGTAA
DMD 158	CTTACAGGAACTCCAGGA	CAATGCCATTGGAGTTCCTGTAA
DMD 159	CTTACAGGAACTCCAGGA	AATGCCATTGGAGTTCCTGTAA
DMD 160	CTTACAGGAACTCCAGGA	ATGCCATTGGAGTTCCTGTAA
DMD 161	CTTACAGGAACTCCAGGA	TGCCATTGGAGTTCCTGTAA
DMD 162	CTTACAGGAACTCCAGGA	ACAACAGTTTGCCGCTGCCCAATGCCATTGGAGTTCCTGTAAG
DMD 163	CTTACAGGAACTCCAGGA	CAACAGTTTGCCGCTGCCCAATGCCATTGGAGTTCCTGTAAG

DMD 164	CTTACAGGAACTCCAGGA	AACAGTTTGCCGCTGCCCAATGCCATTGGAGTTCCTGTAAG
DMD 165	CTTACAGGAACTCCAGGA	ACAGTTTGCCGCTGCCCAATGCCATTGGAGTTCCTGTAAG
DMD 166	CTTACAGGAACTCCAGGA	CAGTTTGCCGCTGCCCAATGCCATTGGAGTTCCTGTAAG
DMD 167	CTTACAGGAACTCCAGGA	AGTTTGCCGCTGCCCAATGCCATTGGAGTTCCTGTAAG
DMD 168	CTTACAGGAACTCCAGGA	GTTTGCCGCTGCCCAATGCCATTGGAGTTCCTGTAAG
DMD 169	CTTACAGGAACTCCAGGA	TTTGCCGCTGCCCAATGCCATTGGAGTTCCTGTAAG
DMD 170	CTTACAGGAACTCCAGGA	TTGCCGCTGCCCAATGCCATTGGAGTTCCTGTAAG
DMD 171	CTTACAGGAACTCCAGGA	TGCCGCTGCCCAATGCCATTGGAGTTCCTGTAAG
DMD 172	CTTACAGGAACTCCAGGA	GCCGCTGCCCAATGCCATTGGAGTTCCTGTAAG
DMD 173	CTTACAGGAACTCCAGGA	CCGCTGCCCAATGCCATTGGAGTTCCTGTAAG
DMD 174	CTTACAGGAACTCCAGGA	CGCTGCCCAATGCCATTGGAGTTCCTGTAAG
DMD 175	CTTACAGGAACTCCAGGA	GCTGCCCAATGCCATTGGAGTTCCTGTAAG
DMD 176	CTTACAGGAACTCCAGGA	CTGCCCAATGCCATTGGAGTTCCTGTAAG
DMD 177	CTTACAGGAACTCCAGGA	TGCCCAATGCCATTGGAGTTCCTGTAAG
DMD 178	CTTACAGGAACTCCAGGA	GCCCAATGCCATTGGAGTTCCTGTAAG
DMD 179	CTTACAGGAACTCCAGGA	CCCAATGCCATTGGAGTTCCTGTAAG
DMD 180	CTTACAGGAACTCCAGGA	CCAATGCCATTGGAGTTCCTGTAAG
DMD 181	CTTACAGGAACTCCAGGA	CAATGCCATTGGAGTTCCTGTAAG
DMD 182	CTTACAGGAACTCCAGGA	AATGCCATTGGAGTTCCTGTAAG
DMD 183	CTTACAGGAACTCCAGGA	ATGCCATTGGAGTTCCTGTAAG
DMD 184	CTTACAGGAACTCCAGGA	TGCCATTGGAGTTCCTGTAAG

Bibliography:

[1] "Genetic Engineering - an overview | ScienceDirect Topics." [Online]. Available: https://www.sciencedirect.com/topics/neuroscience/genetic-engineering. [Accessed: 22-Nov-2020].

[2] A. V Anzalone *et al.*, "Search-and-replace genome editing without double-strand breaks or donor DNA," *Nature*, Oct. 2019.

[3] G. A. R. Gonçalves and R. de M. A. Paiva, "Gene therapy: advances, challenges and perspectives," *Einstein (Sao Paulo, Brazil)*, vol. 15, no. 3. Instituto de Ensino e Pesquisa Albert Einstein, pp. 369–375, 01-Jul-2017.

[4] F. Camastra, M. D. Di Taranto, and A. Staiano, "Statistical and Computational Methods for Genetic Diseases: An Overview," *Computational and Mathematical Methods in Medicine*, vol. 2015. Hindawi Publishing Corporation, 2015.

[5] V. A. Padilha, O. S. Alkhnbashi, S. A. Shah, A. C. P. L. F. de Carvalho, and R. Backofen, "CRISPRcasIdentifier: Machine learning for accurate identification and classification of CRISPR-Cas systems," *Gigascience*, vol. 9, no. 6, pp. 1–12, Jun. 2020.

[6] Y. L. Min *et al.*, "CRISPR-Cas9 corrects Duchenne muscular dystrophy exon 44 deletion mutations in mice and human cells," *Sci. Adv.*, vol. 5, no. 3, 2019.

[7] J. R. Costa, B. E. Bejcek, J. E. McGee, A. I. Fogel, K. R. Brimacombe, and R. Ketteler, *Genome Editing Using Engineered Nucleases and Their Use in Genomic Screening*. 2004.

[8] Addgene, "Addgene CRISPR Guide," 2018. [Online]. Available: https://www.addgene.org/guides/crispr/. [Accessed: 16-Dec-2019].

[9] J. G. Doench *et al.*, "Optimized sgRNA design to maximize activity and minimize off-target effects of CRISPR-Cas9," *Nat. Biotechnol.*, vol. 34, no. 2, pp. 184–191, 2016.

[10] T. W. Y. Wong and R. D. Cohn, "Therapeutic Applications of CRISPR/Cas for Duchenne Muscular Dystrophy," *Curr. Gene Ther.*, vol. 17, no. 4, Nov. 2017.

[11] "Duchenne muscular dystrophy | Genetic and Rare Diseases Information Center (GARD) – an NCATS Program." [Online]. Available: https://rarediseases.info.nih.gov/diseases/6291/duchenne-muscular-dystrophy. [Accessed: 08-Jan-2020].

[12] "The Role of Dystrophin in Duchenne | Duchenne.com." [Online]. Available: https://www.duchenne.com/importance-of-dystrophin. [Accessed: 09-Nov-2019].

[13] "Duchenne Muscular Dystrophy (DMD) | Muscular Dystrophy Association." [Online]. Available: https://www.mda.org/disease/duchenne-muscular-dystrophy. [Accessed: 08-Nov-2019].

[14] "Duchenne Muscular Dystrophy - NORD (National Organization for Rare Disorders)." [Online]. Available: https://rarediseases.org/rare-diseases/duchenne-muscular-dystrophy/. [Accessed: 08-Jan-2020].

[15] G. L. Walmsley *et al.*, "A duchenne muscular dystrophy gene hot spot mutation in dystrophin-deficient Cavalier King Charles Spaniels is amenable to exon 51 skipping," *PLoS One*, vol. 5, no. 1, Jan. 2010.

[16] "Taken from www.dmd.nl/gt. Mutation specific therapies."

[17] "Types of Mutations - Parent Project Muscular Dystrophy." [Online]. Available: https://www.parentprojectmd.org/about-duchenne/what-is-duchenne/types-of-mutations/. [Accessed: 18-Jan-2020].

[18] M. De Los Angeles Beytía, J. Vry, and J. Kirschner, "Drug treatment of Duchenne muscular dystrophy: Available evidence and perspectives," *Acta Myologica*, vol. 31, no. MAY. pp. 4–8, May-2012.

[19] L. Echevarría, P. Aupy, and A. Goyenvalle, “Exon-skipping advances for Duchenne muscular dystrophy,” *Human molecular genetics*, vol. 27, no. R2. NLM (Medline), pp. R163–R172, 01-Aug-2018.
[20] C. S. Young, E. Mokhonova, M. Quinonez, A. D. Pyle, and M. J. Spencer, “Creation of a Novel Humanized Dystrophic Mouse Model of Duchenne Muscular Dystrophy and Application of a CRISPR/Cas9 Gene Editing Therapy,” *J. Neuromuscul. Dis.*, vol. 4, no. 2, pp. 139–145, 2017.
[21] T. Koo *et al.*, “Functional Rescue of Dystrophin Deficiency in Mice Caused by Frameshift Mutations Using Campylobacter jejuni Cas9,” *Mol. Ther.*, vol. 26, no. 6, pp. 1529–1538, Jun. 2018.
[22] Y. Zhang *et al.*, “Enhanced CRISPR-Cas9 correction of Duchenne muscular dystrophy in mice by a self-complementary AAV delivery system,” *Sci. Adv.*, vol. 6, no. 8, 2020.
[23] “Genome Editing Heads to Primetime.” [Online]. Available: https://www.genengnews.com/insights/genome-editing-heads-to-primetime/. [Accessed: 06-Nov-2019].
[24] H. Ledford, “Super-precise new CRISPR tool could tackle a plethora of genetic diseases,” *Nature*, vol. 574, no. 7779. NLM (Medline), pp. 464–465, 01-Oct-2019.
[25] “Everything You Need to Know About Superstar CRISPR Prime Editing.” [Online]. Available: https://singularityhub.com/2019/11/05/everything-you-need-to-know-about-superstar-crispr-prime-editing/. [Accessed: 06-Nov-2019].
[26] S. H. Park *et al.*, “Highly efficient editing of the β-globin gene in patient-derived hematopoietic stem and progenitor cells to treat sickle cell disease,” *Nucleic Acids Res.*, vol. 47, no. 15, pp. 7955–7972, 2019.
[27] D. Kim, D. eun Kim, G. Lee, S. I. Cho, and J. S. Kim, “Genome-wide target specificity of CRISPR RNA-guided adenine base editors,” *Nature Biotechnology*, vol. 37, no. 4. Nature Publishing Group, pp. 430–435, 01-Apr-2019.
[28] H. Peng, Y. Zheng, Z. Zhao, T. Liu, and J. Li, “Recognition of CRISPR/Cas9 off-target sites through ensemble learning of uneven mismatch distributions,” *Bioinformatics*, vol. 34, no. 17, pp. i757–i765, Sep. 2018.
[29] B. P. Kleinstiver *et al.*, “High-fidelity CRISPR-Cas9 nucleases with no detectable genome-wide off-target effects,” *Nature*, vol. 529, no. 7587, pp. 490–495, Jan. 2016.
[30] I. M. Slaymaker, L. Gao, B. Zetsche, D. A. Scott, W. X. Yan, and F. Zhang, “Rationally engineered Cas9 nucleases with improved specificity,” *Science (80-.).*, vol. 351, no. 6268, pp. 84–88, 2016.
[31] H. Nishimasu *et al.*, “Engineered CRISPR-Cas9 nuclease with expanded targeting space,” *Science (80-.).*, vol. 361, no. 6408, pp. 1259–1262, 2018.
[32] “Benchling [Biology Software].” .
[33] “RRID | Resource Report (RRID:SCR_013955).” [Online]. Available: https://scicrunch.org/resources/Any/record/nlx_144509-1/SCR_013955/resolver?q=*&l=. [Accessed: 20-Jan-2020].
[34] K. Will, “Multiple Linear Regression – MLR Definition,” *Mult. Linear Regres. – MLR Defin.*, 2019.
[35] “Support Vector Machine Regression,” 2015. [Online]. Available: http://kernelsvm.tripod.com/. [Accessed: 15-Apr-2020].
[36] “8.6.2. sklearn.ensemble.RandomForestRegressor — scikit-learn 0.11-git documentation.” [Online]. Available: https://ogrisel.github.io/scikit-learn.org/sklearn-tutorial/modules/generated/sklearn.ensemble.RandomForestRegressor.html. [Accessed: 17-Mar-2020].
[37] “Gradient Boosting from scratch - ML Review - Medium.” [Online]. Available: https://medium.com/mlreview/gradient-boosting-from-scratch-1e317ae4587d.

[Accessed: 04-Apr-2020].
[38] "3.2.4.3.6. sklearn.ensemble.GradientBoostingRegressor — scikit-learn 0.22.2 documentation." [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.GradientBoostingRegressor.html. [Accessed: 17-Mar-2020].
[39] Will Koehrsen, "Hyperparameter Tuning the Random Forest in Python – Towards Data Science," *Towards Data Science*, 2018. [Online]. Available: https://towardsdatascience.com/hyperparameter-tuning-the-random-forest-in-python-using-scikit-learn-28d2aa77dd74. [Accessed: 16-May-2020].
[40] A. C. H. Choong and N. K. Lee, "Evaluation of Convolutionary Neural Networks Modeling of DNA Sequences using Ordinal versus one-hot Encoding Method," *bioRxiv*, p. 186965, Sep. 2017.
[41] "Working with DNA sequence data for ML | Kaggle." [Online]. Available: https://www.kaggle.com/thomasnelson/working-with-dna-sequence-data-for-ml. [Accessed: 06-Feb-2020].