Spring 2023

Characterizing Sequencing Artifacts

Kathy Thanh Lam
San Jose State University

Characterizing Sequencing Artifacts

A Project

Presented to

The Faculty of the Department of Computer Science

San José State University

In Partial Fulfillment

of the Requirements for the Degree

Master of Science

by

Kathy Thanh Lam

May 2023

The Designated Project Committee Approves the Project Titled

Characterizing Sequencing Artifacts

by

Kathy Thanh Lam

APPROVED FOR THE DEPARTMENT OF COMPUTER SCIENCE

SAN JOSÉ STATE UNIVERSITY

May 2023

ABSTRACT

Characterizing Sequencing Artifacts

by Kathy Thanh Lam

Next Generation Sequencing (NGS) introduces artifactual variants from library preparation methods and errors, which affects the accuracy of variant calling. Whole Exome Sequencing (WES) data from the National Center for Biotechnology Information (NCBI) Sequence Read Archive (SRA) database is processed. Comparison of single nucleotide polymorphism (SNP) calls to Genome In a Bottle (GIAB) provides labels that are used to build machine learning (ML) models. The left and right flanking region (LSEQ and RSEQ) of each SNP is extracted. Nucleotide frequency, kmers of size 4 and their counts, largest homopolymer size, largest palindrome size, and largest hairpin loop size were computed and used as features in model building. The Random Forest model had a precision of 98.8%, recall of 87.3%, and accuracy of 90.2%. High scores show the model's ability to correctly identify artifacts from non-artifacts and that the results are exceptionally accurate.

ACKNOWLEDGMENTS

I want to thank my advisor Dr. Wendy Lee for her tremendous support and guidance throughout my graduate journey and the development of this project. I would also like to thank Dr. Philip Heller and Dr. William Andreopoulos for being a part of my committee and providing invaluable advice and feedback. Finally, I would like to thank my mom, dad, and brother for their continued support and for always believing in me.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

Introduction

Cancer is the most common genetic disease. Each year millions of people are diagnosed with a type of this disease. This disease continues to evolve and progress through genetic alterations from germline cells and somatic cells [3]. Germline cell mutations are inherited from parents to offspring while somatic cell mutations are acquired throughout one's lifetime from exposure to carcinogens, chemicals, radiation, alcohol, food, or any other environmental factors [4]. The DNA sequence variations that arise from these mutations can lead to the development of cancer. Cancer can go undetected for years, making diagnosis and treatment difficult. Typically, once it is diagnosed, there is the option of surgery, chemotherapy, or radiation, all of which can be expensive, invasive, and have distressing side effects.

To detect and monitor cancer, a biopsy needs to be done. This allows collected tissue samples to be used for molecular testing. A typical biopsy takes tissue samples from cancer cells, which can be invasive and difficult to obtain. Some examples are brain or lung tissue samples. Alternatively, a liquid biopsy collects blood samples to detect cancer cells from pieces of a tumor that is circulating in the blood [5]. Liquid biopsies paired with Next-generation sequencing (NGS) are non-invasive approaches to cancer detection and monitoring since they can be repeated without any risks or side effects [5]. The limitations of NGS are the artifactual variants that are introduced by library preparation and errors from NGS platforms. These artifacts can be mistaken as true variants, thus affecting the accuracy of variant calling.

DNA consists of exons, introns, coding regions, and non-coding regions. In protein production and gene expression, introns are spliced out while the exons bond together to create mRNA that is then translated to protein. This project will focus on whole exome sequencing (WES), which allows for the sequencing of only the coding regions of a genome. WES is the main sequencing technique used in cancer genome sequencing [6].

The goal of this project is to use the DNA sequences flanking the single-nucleotide artifact to identify and classify true variants from artifactual variants. With the final variant call format (VCF) file, the left flanking regions (LSEQ) and the right flanking regions (RSEQ) of the variants will be extracted. The LSEQ is the 20 base pairs (bp) neighboring the variant at the 5' end, while the RSEQ is the 20 bp neighboring the variant at the 3' end [7]. Metrics will be computed from the LSEQ and RSEQ, including: the content of each base, each kmer of size 4 and their counts, if there are homopolymers of size 3 or more of each base, if it contains a palindrome and forms complementary with itself, and if it forms a hairpin loop. Using machine learning (ML) models on the LSEQ and RSEQ metrics, potential sequencing platform-specific and molecular technique-specific artifactual variants can be characterized.

1.1 Next Generation Sequencing

NGS allows for massive parallel sequencing of whole genomes. NGS can be used to diagnose different diseases and viruses. This technology has the ability to provide genetic information about an individual and determine what diseases that individual is more susceptible to. It is becoming increasingly popular to use NGS in cancer therapeutics because of its speed and ability to detect rare variants due to its sequencing depth.

The NGS workflow contains several parts: nucleic acid extraction, library preparation, sequencing, and data analysis (Fig. 1). The library is created by randomly fragmenting the DNA extraction, adding NGS platform-specific adapter sequences, and amplifying by Polymerase Chain Reaction (PCR) or some alternative technique [5, 6]. Some common NGS platforms include Illumina, Ion Torrent, PacBio single-molecule real-time (SMRT), and Oxford Nanopore [5, 6].

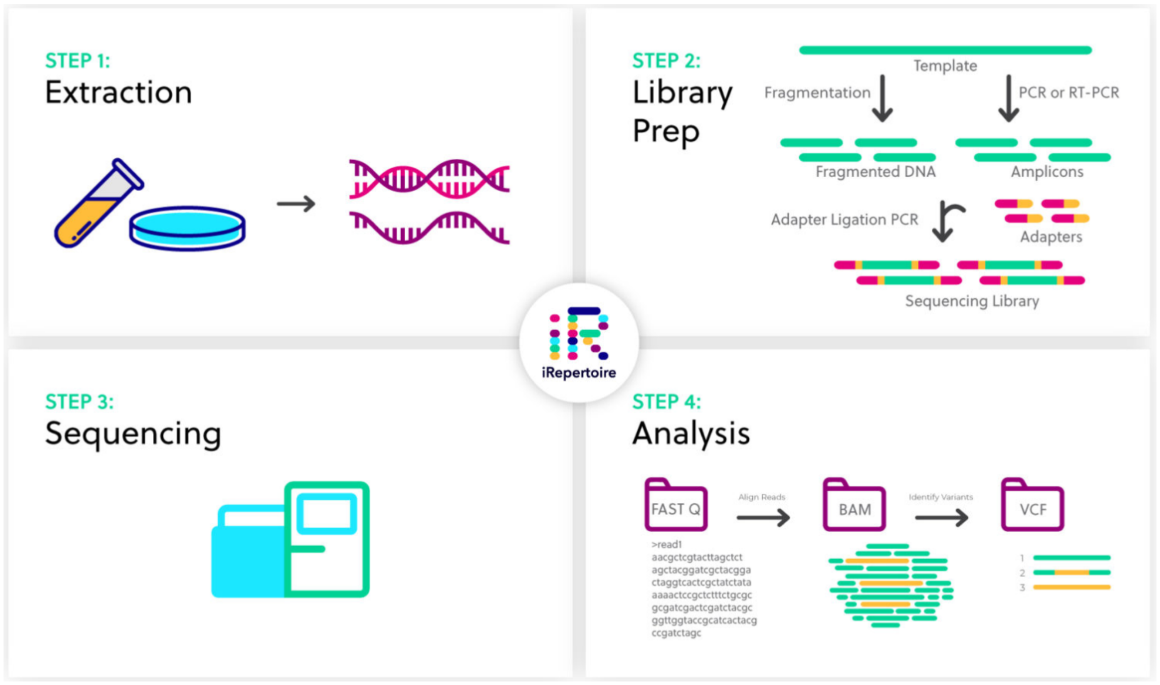


Fig. 1: NGS workflow [1]

1.2 Variants and Artifacts

Library preparation has physical steps such as DNA shearing, and chemical steps such as PCR, which can introduce artifactual variants in the DNA. The sequencing errors that arise from the NGS workflow make detecting true variants difficult.

Variants are naturally occurring mutations in the genome that can affect how an individual responds to certain drugs or pathogens. Artifacts are artificial mutations that arise from various steps in the NGS workflow, such as sample handling, library preparation, PCR enrichment, and sequencing. If artifacts are not properly identified, they will mislead downstream analysis and can hinder the applications of precision medicine. Sequencing errors are key confounding factors for detecting low-frequency genetic variants that are important for cancer diagnosis and treatment [8].

CHAPTER 2

Materials and Methods

The bioinformatics pipeline, flanking region metrics, machine learning pipeline, and resources are described. Data from the National Center for Biotechnology Information (NCBI) Sequence Read Archive (SRA) and Genome in a Bottle Consortium (GIAB) were used [9, 10]. The NCBI SRA was used because it includes sequencing read datasets from different sequencing platforms and library preparation methods. This project focuses on WES paired-end data. The GIAB will be the "Gold-standard" that we will compare our variant calls with. GIAB was chosen because of its extensive research and high confidence (HC) variants used in benchmarking, which reliably identify false-positive and false-negative variant calls [10]. HC calls were developed using 11 whole-genome and three exome data sets from five sequencing platforms and seven mappers [11].

2.1 Bioinformatics Pipeline

The bioinformatics pipeline was built using Snakemake, a workflow management tool. This allows for the pipeline to be reproducible and scalable [12]. This project was scaled on a high-performance compute cluster using SLURM. An overview of the bioinformatics pipeline is shown in Fig. 2.

2.1.1 Downloading Data from SRA Repository

Twenty-one WES paired-end datasets from eight different samples with SRA accession numbers ERR1831349, ERR1831346, ERR1831347, ERR1831348, ERR1831350, ERR1831351, ERR1831352, ERR1831353, ERR1905889, ERR1905890, SRR14724463, SRR14724473, SRR14724483, SRR14724493, SRR14724503, SRR14724513, SRR2106342, SRR2106344, DRR189730, DRR189731, and DRR189732 were downloaded from the NCBI SRA repository using the SRA Toolkit [13]. This is a command-line tool that extracts the data into FASTQ format. The data are from BGISEQ-500,

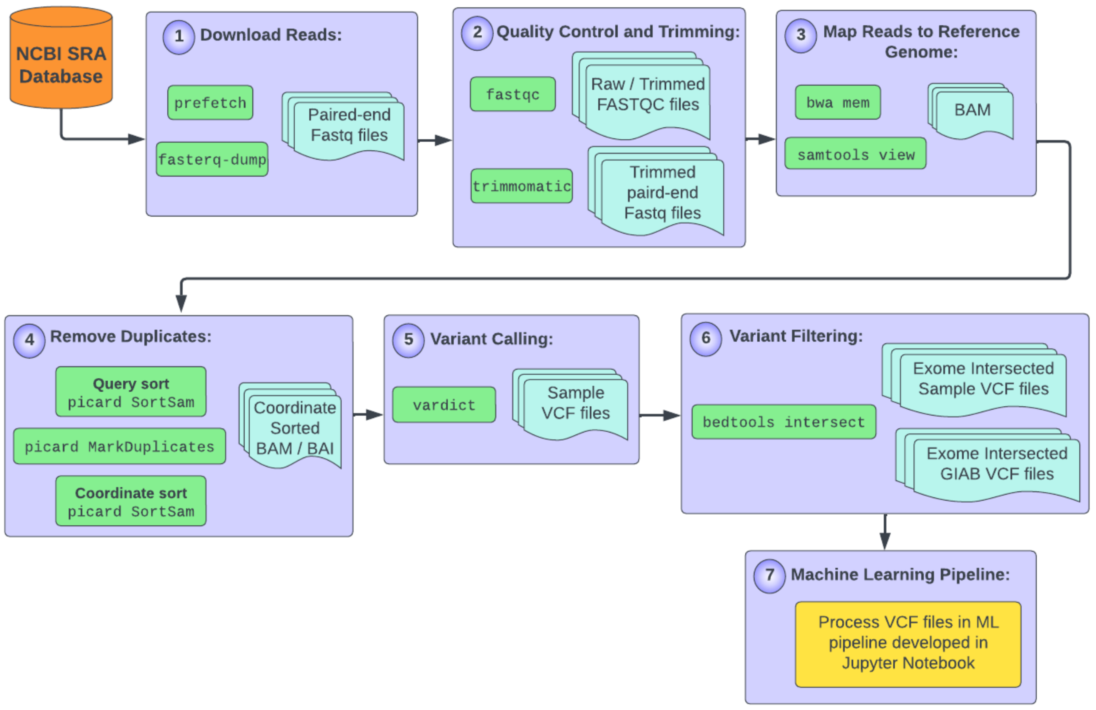

NCBI SRA Database
1 Download Reads:
prefetch
fasterq-dump
Paired-end Fastq files
2 Quality Control and Trimming:
fastqc
Raw / Trimmed FASTQC files
trimmomatic
Trimmed paird-end Fastq files
3 Map Reads to Reference Genome:
bwa mem
samtools view
BAM
4 Remove Duplicates:
Query sort picard SortSam
picard MarkDuplicates
Coordinate sort picard SortSam
Coordinate Sorted BAM / BAI
5 Variant Calling:
vardict
Sample VCF files
6 Variant Filtering:
bedtools intersect
Exome Intersected Sample VCF files
Exome Intersected GIAB VCF files
7 Machine Learning Pipeline:
Process VCF files in ML pipeline developed in Jupyter Notebook

Fig. 2: Overview of Bioinformatics Pipeline

NovaSeq 600, HiSeq 4000, and HiSeq 2500 sequencers, using Agilent V5 sure select, Agilent V7 sure select, Truseq, IDT, Illumina Nextera Rapid Capture, Agilent SureSelectv5+UTR, Agilent SureSelect XT HS, Agilent SureSelect XT, and Kapa HyperPrep exome capture kit of NA12878/HG001 cell line DNA. The NA12878/HG001 cell line was chosen because GIAB already characterized it as a pilot genome to be used in benchmarking and validating variant calling pipelines.

2.1.2 Quality Control

Quality control of the raw read data will ensure that the data is clean and of high quality, so it will not negatively impact downstream analysis. FastQC, a quality control tool for high throughput sequence data was used for visualization [14]. This allows the raw reads to be evaluated and generates a report that can be viewed in any web browser.

2.1.3 Adapter Trimming

From the FastQC results, we can determine if sequencing reads need to be trimmed due to the presence of low-quality bases and/or sequencing adapter sequences. Trimmomatic was the command-line tool used to trim FASTQ data and remove adapter sequences [15]. When the DNA library is prepared for NGS, adapter sequences are ligated to the DNA fragments to allow them to attach to the flow cell for sequencing [16]. Adapter sequences should be removed from reads because they are not part of the original DNA we are trying to sequence and interfere with downstream analysis, such as the alignment of reads to a reference genome [16].

2.1.4 Map Reads to Reference Genome

The trimmed reads were aligned to Genome Reference Consortium Human Build 38 (hg38), the reference genome, using the Burrows-Wheeler Alignment Tool. Specifically, the Burrows-Wheeler Alignment-Maximal Exact Matches (BWA-MEM) algorithm was used. The BWA software package is used for mapping low-divergent sequences against a large reference genome [17]. The tool outputs a sequence alignment map (SAM) file that was converted to its compressed binary counterpart, known as a BAM file, for easier handling and saving disk space. This was accomplished using SAMtools. SAMtools provide a set of utilities for manipulating alignments in SAM and BAM formats [18]. This SAM/BAM file stores alignment information for each sequencing read against the reference genome.

2.1.5 Remove Duplicates

The BAM file created from mapping reads to the reference genome was sorted by query name using Picard. Picard contains command-line tools for manipulating high-throughput sequencing (HTS) data and formats [19]. Reads must be query-sorted prior to removing duplicates so that unmapped mates of mapped records and supplementary alignments can be marked as duplicates; coordinate-sorted does not achieve the same results [19].

Picard was then used to mark and remove duplicate reads. Duplicate reads are reads that originate from a single fragment of DNA. Duplicates can arise during library preparation of the NGS workflow where PCR is involved, which are known as PCR duplicates [19]. Duplicate reads can also arise from a single amplification cluster, incorrectly detected as multiple clusters by the optical sensor of the sequencer, which are known as optical duplicates [19]. An index was created for the deduplicated BAM file to be used in variant calling.

2.1.6 Variant Calling

VarDictJava was used for variant calling because it provides the LSEQ and RSEQ of the called variants. The VarDictJava inputs were hg38 reference genome in FASTA format, deduplicated aligned reads in BAM format, and GIAB target regions in Browser Extensible Data (BED) format. Providing the GIAB BED file as the target region means that the resulting VCF file will filter out any variant calls that are not present in GIAB, and therefore not of HC. An allele frequency (AF) of 0.01 was used during variant calling. Normal AF is typically 0.5 or 1 since the mother has the allele, the father has the allele, or they both have the allele. It is easy to determine germline variants by looking at AF because it will either be 50% or 100% when looking at the depth. Cancer variants have a low AF, typically 0.01 or 1%, making them harder to identify and why 0.01 is used for this parameter.

2.1.7 Variant Filtering

The SRA accession numbers and their target exome capture kits are listed in Table 1. Each sequencing platform varies in its exome capture methods, which influences sequencing. It is important to do an intersection/filtering step where the

VarDictJava VCF output is intersected with the sequencing platform's target exome BED file. This ensures that the final VCF file used to generate metrics and build the machine learning model will contain only the variant calls that are present in the GIAB HC regions and the sequencing platform's target exome regions, as seen in Fig. 3. The GIAB HC VCF file was also intersected with the sequencing platform's target exome BED file for the same reason. This final GIAB VCF file will be the truth set in which we compare the final SRA VCF file. Both intersection steps were done using BEDtools, since the BEDTools suite provides a set of utilities for genomic analysis tasks [20].

Table 1: SRA accession numbers and target exome regions

SRA accession number	Exome capture kit type	Platform
ERR1831349	Agilent SureSelect v5	BGI
ERR1831346	Agilent SureSelect v5	BGI
ERR1831347	Agilent SureSelect v5	BGI
ERR1831348	Agilent SureSelect v5	BGI
ERR1831350	Agilent SureSelect v5	BGI
ERR1831351	Agilent SureSelect v5	BGI
ERR1831352	Agilent SureSelect v5	BGI
ERR1831353	Agilent SureSelect v5	BGI
SRR14724463	TruSeq	Illumina
SRR14724473	IDT	Illumina
SRR14724483	Agilent SureSelect v7	Illumina
SRR14724493	TruSeq	Illumina
SRR14724503	IDT	Illumina
SRR14724513	Agilent SureSelect v7	Illumina
ERR1905889	Agilent SureSelect v5	Illumina
ERR1905890	Agilent SureSelect v5	Illumina
SRR2106344	Illumina Nextera Rapid Capture	Illumina
SRR2106342	Agilent SureSelectv5+UTR	Illumina
DRR189730	Agilent SureSelect XT HS	Illumina
DRR189731	Agilent SureSelect XT	Illumina
DRR189732	Kapa HyperPrep	Illumina

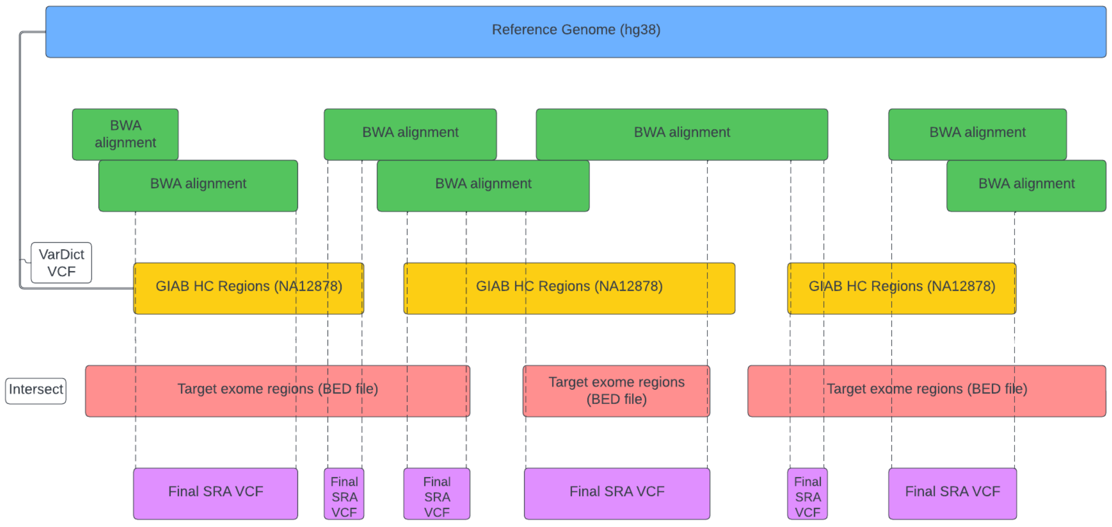


Fig. 3: SRA VCF file intersection with target exome region BED file

2.2 Determine Flanking Region Metrics

After completion of the bioinformatics pipeline, we are left with the final SRA VCF file and the final GIAB VCF file. A Jupyter notebook called `final.ipynb` was developed to import the `get_metrics.py` module built for this project. All of the SRA VCF files were converted into individual Pandas data frames with the LSEQs and RSEQs. The flanking region metrics were computed and populated into the data frames.

2.2.1 get_metrics.py

The python module `get_metrics.py` was created to store all the functions that will be imported and used in the Jupyter notebook `final.ipnyb`. It determines artifacts and computes flanking region metrics, using the functions described below.

2.2.1.1 vcf_to_df

This function takes any VCF file and extracts CHROM, POS, REF, and ALT columns. This information is converted into a Pandas data frame. There is also a default argument `extract_flank_seqs=False` that can be set to `True` if a VarDict

VCF file is used and the LSEQ and RSEQ information is desired. The allele frequency (AF), allele frequency using only high-quality bases (HIAF), high quality variant reads (HICNT), variant depth (VD), signal to noise (SN), adjusted AF for indels due to local realignment (ADJAF), and variant depth by strand (VARBIAS) are also extracted because these features have a high percent accuracy for model training [21]. This function parses the appropriate information for the final GIAB VCF file and the final SRA VCF file.

2.2.1.2 snp_freq_not1

This function takes a data frame and finds positions where the ALT has multiple variant calls (e.g. C, CAA). If any of the calls for that position are single nucleotides, then this is considered to be a SNP. This is used for the GIAB VCF files where this situation is present. When vardict encounters this situation, it automatically reports the variant with the highest frequency, so we do not see multiple calls in the VCF file.

2.2.1.3 filter_for_snvs

This function filters for single nucleotide variants (SNVs) by ensuring the string length of the REF column and ALT column is 1. SNVs are sequence variations that occur when a single nucleotide is altered.

2.2.1.4 join_datasets

This function is used to merge the SRA and GIAB Pandas data frame on the CHROM and POS columns. In `final.ipnyb` CHROM and POS were used as the indexes for each data frame to make each index unique.

2.2.1.5 extract_artifacts_column

This function extracts artifacts and differentiates between the 4 different variant calling cases, explained further in the Machine Learning Pipeline section, Chapter 2.4.

2.2.1.6 content

This function calculates the content of each base: A, T, C, G, in the sequence. With the content of each individual base, it is easy to calculate the AT and GC contents. A list of dictionaries storing the content of A, T, C, and G is returned, where each dictionary represents each sequence passed to this function.

2.2.1.7 find_kmer

This function finds all kmers of a given size using a sliding window along the sequence. It calls the helper function, `kmer_count`, to find the count of each unique kmer. A kmer size of 4 was used as the default value but can be changed by editing the `kmer_size` parameter. A list of dictionaries is returned where each dictionary contains all kmers and their counts for each sequence passed to this function.

2.2.1.8 homopolymer

This function calls on the `find_kmer` function to take advantage of the sliding window to determine if the kmers are homopolymers. A homopolymer is repeated copies of a single base (e.g. AAAA). This is done for a homopolymer size of 3 up to the length of the sequence. A list of dictionaries is returned showing the largest homopolymer size for each base. Each dictionary in the list represents each sequence passed to this function.

2.2.1.9 palindrome

A palindrome is a word that reads the same backward as forward. In DNA, a palindrome is a sequence that when read backward is the complement of the sequence read forward [2]. When a palindromic sequence is split in half, the two halves complement each other as seen in Fig. 4.

This function calls on the `find_kmer` function to take advantage of the sliding window to determine if the kmers are palindromes. This is done for a palindrome size of 3 up to the length of the sequence. If the reverse complement is identical to the original kmer sequence, then it is a palindrome. The helper function, `complement`, assists in computing the sequence's complement strand. A list is returned with the largest palindrome size for each sequence passed to this function.

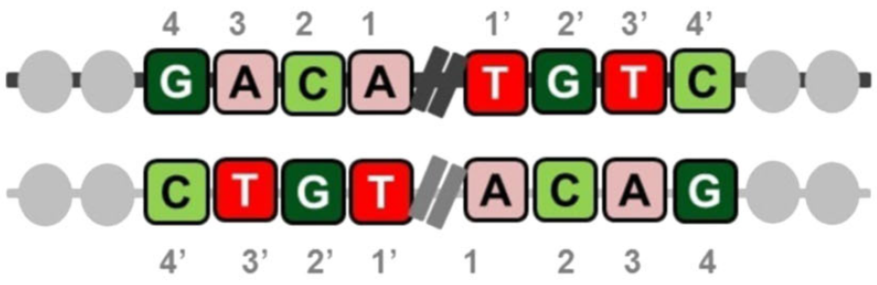


Fig. 4: Example of palindromic sequence [2]

2.2.1.10 hairpin

This function calls on the `find_kmer` function to take advantage of the sliding window to determine if the kmers form hairpin loops. A hairpin loop consists of the first part of the stem, the loop, and the second part of the stem that folds back on itself to pair with the first part of the stem. If the flanking regions of the loop are the reverse complement of each other, the sequence folds back on itself to form a hairpin, as seen in Fig. 5. The optimal loop length is between 4 and 8 bases [22]. This function is called by the `main` function to try all permutations of loop length of 4 to 8 bases and stem length of 4 to 8 bases. The largest loop size for each sequence is stored in a list.

2.2.1.11 main

This function calls all previous functions to return a complete data frame containing all metrics.

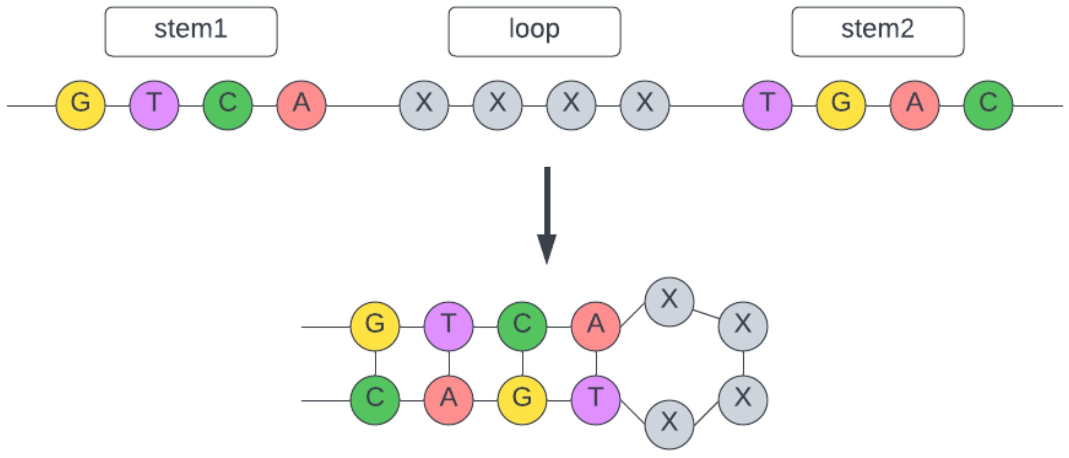


Fig. 5: Example of formation of hairpin loop

2.3 Artifact Identification

All of the GIAB VCF files were also converted into individual Pandas data frames. Artifacts are determined by comparing the GIAB VCF data frame to the SRA VCF data frame. This project focuses on SNVs, all other variants are dropped from their respective data frames before comparison.

There are 4 possible cases when looking at a variant call. In Case 1, GIAB and SRA contain the same variant at the same locus. These are considered true variants and assumed to not be artifacts. In Case 2, GIAB contains a variant and SRA contains a different variant at the same locus. These are assumed to be artifacts. In Case 3, GIAB contains a variant, but SRA does not contain a variant in the same locus. There are two possible scenarios for this case: Case 3.1 the SRA does not contain a call due to lack of read coverage and is not an artifact; Case 3.2 the SRA does not contain a call because the SRA read matches the reference genome and is an artifact. In Case 4, SRA contains a variant, but GIAB does not contain a variant in the same locus. These are also assumed to be artifacts. Fig. 6 shows the different cases and whether they are an artifact. This project focuses on Case 2 and Case 4 because the more difficult Case 3 has not been modeled yet.

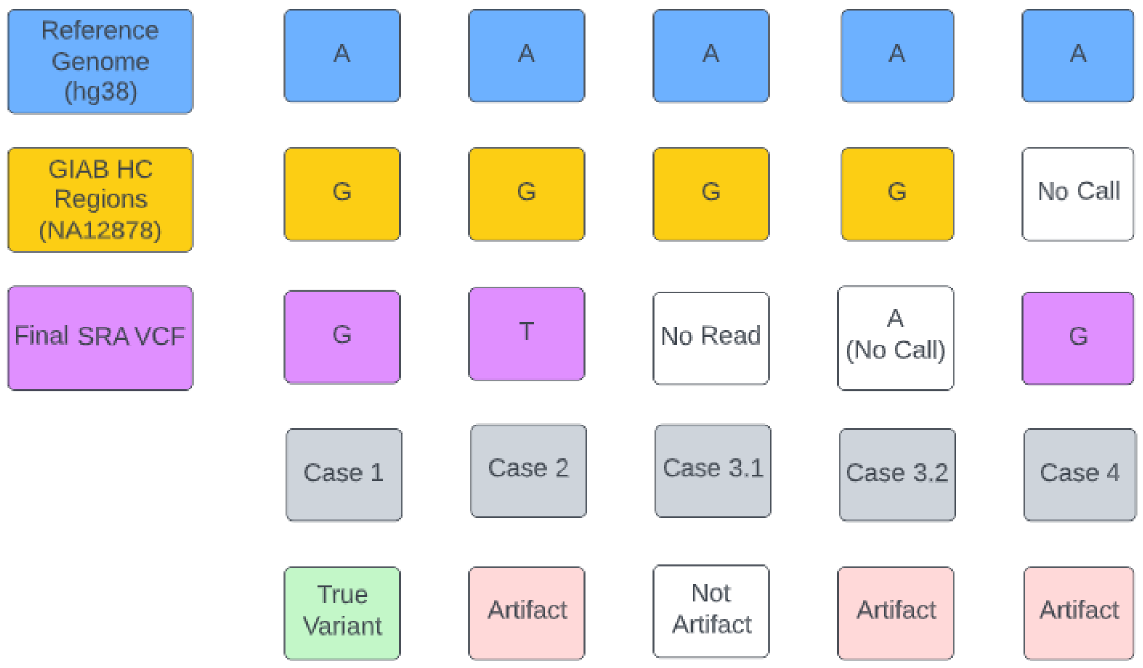


Fig. 6: Example of the 4 possible variant call cases

2.4 Machine Learning Pipeline

After artifacts were identified for each SRA, the data frames were combined into a single master data frame to be explored. A ML pipeline was developed and ML models were built to classify true variants from artifacts. An overview of the ML pipeline is shown in Fig. 7.

2.4.1 Data Exploration

A heat map was generated to determine which SNV was the most prevalent. The count of each nucleotide combination flanking each SNP was generated on a heat map to visualize which nucleotide combinations are most prevalent immediately on the left and right of SNPs. For example, if there is a CCG, where the middle C represents the SNP, the nucleotide combination flanking this SNP would be CG.

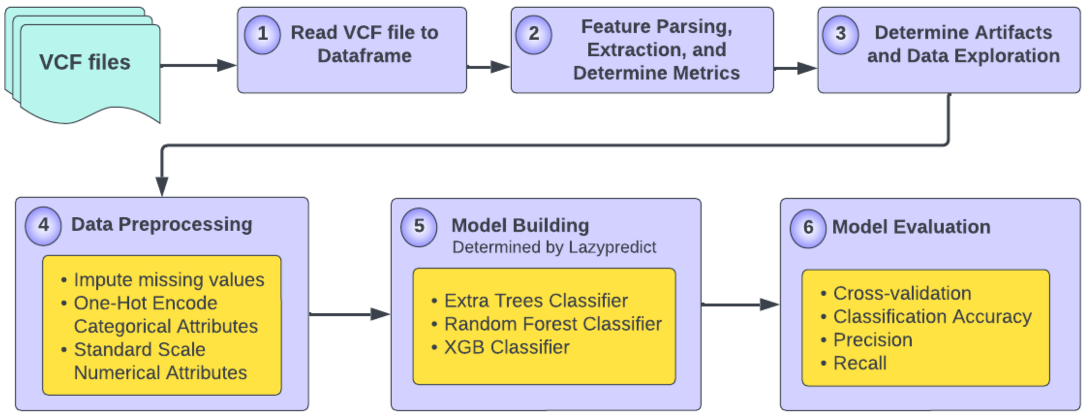

VCF files
1 Read VCF file to Dataframe
2 Feature Parsing, Extraction, and Determine Metrics
3 Determine Artifacts and Data Exploration
4 Data Preprocessing
• Impute missing values
• One-Hot Encode Categorical Attributes
• Standard Scale Numerical Attributes
5 Model Building
Determined by Lazypredict
• Extra Trees Classifier
• Random Forest Classifier
• XGB Classifier
6 Model Evaluation
• Cross-validation
• Classification Accuracy
• Precision
• Recall

Fig. 7: Overview of Machine Learning Pipeline

There are 12 possible SNPs and 16 possible flanking combinations. Heat maps were also generated to visualize the flanking region frequencies of each nucleotide, kmers of size 4, homopolymers, palindromes, and hairpin loops. For kmers, only the 20 most prevalent were visualized. Plots were created using Matplotlib, Seaborn, and Pandas.

2.4.2 Feature Selection

VCF files contain a lot of information in the INFO column that could be used as features. Selecting good features is vital when it comes to modeling data. AF, HIAF, HICNT, VD, SN, and ADJAF were chosen because they were the top 6 most important features for model accuracy [21]. All flanking region metrics, REF, and ALT were also used as features.

2.4.3 Data Pre-Processing

The LSEQ and RSEQ columns are dropped before modeling since these were only needed to determine flanking region metrics. VARBIAS was also dropped because it is a categorical feature that resulted in approximately 38,000 additional columns after one-hot encoding. There are 540 features in total. The categorical ones are REF and ALT, while the numerical ones are everything else, with IS_ARTIFACT being the label.

This data does not contain any missing values, however, it is common for missing values to be present in real-world datasets. To guarantee that future datasets won't have missing values, it is best to apply an imputer to all numerical attributes. The strategy used was to fill missing values with median values since the median is less affected by outliers. Feature scaling on numerical values was done since ML algorithms do not perform well when the input numerical attributes have very different scales. For this, standardization was used because it is much less affected by outliers than normalization (min-max scaling). In standardization, the mean is subtracted from each value and then divided by the standard deviation. The resulting distribution then has unit variance and a mean of 0 [23].

One-hot encoding was used for the categorical attributes in order to convert these text-based columns to numerical outputs to work with ML algorithms. This works by creating one binary attribute per category, where 1 attribute will be equal to 1 and the others will be 0. This type of encoding is needed for feeding categorical data into many Scikit-learn estimators [24].

The numerical and categorical transformation pipelines were then joined to create a full transformation pipeline that can be used on the dataset to create the features.

2.4.4 Model Building

This project utilizes supervised ML. The model is trained using a labeled dataset, IS_ARTIFACT, and based on the training, the model predicts the output, True or False. The type of supervised ML used will be binary classification. The lazypredict package was used to build a lot of basic models without any parameter tuning to identify which top models will be used for additional ML tasks [25]. The default list of models that lazypredict uses contains 2 very computationally heavy models that were removed. These models were Support Vector Classifier and Nu-Support Vector Classifier. These models created a bottleneck in the pipeline and would not run to completion after 48 hours.

The top 3 models are Extra Trees Classifier, Random Forest Classifier, and XGB Classifier. These algorithms will be used to model the data. For Extra Trees and Random Forest `class_weights="balance"`, while for XGB `scale_pos_weight=sum(negative instances)/sum(positive instances)` [26, 27, 28]. Since artifacts are much more present in the data than non-artifacts, it is critical to train models with balanced datasets to avoid bias when predicting one class over another [29].

2.4.5 Performance Evaluation of Model Building

K-fold cross-validation with a K of 3 was used to evaluate each model's performance. This technique randomly splits the training set into distinct K subsets, called folds. It trains and evaluates the model K times, picking a different fold for evaluation each time and training on the other K-1 folds. A higher number of folds was initially tested but the computation time was unmanageable and created memory issues. Cross-validation was used to detect overfitting, which is a result of the model failing to extrapolate a pattern.

Confusion matrices were built for the test set of each model, where metrics such as accuracy, precision, and recall can be calculated [30]. Precision measures the proportion of positive identifications that were actually correct, while recall measures correctly identifying true positives. The equations for precision and recall are:

$$\text{Precision} = \frac{\text{True Positive}}{\text{True Positive} + \text{False Positive}}$$

$$\text{Recall} = \frac{\text{True Positive}}{\text{True Positive} + \text{False Negatives}}$$

For this specific problem, precision tells us the measure of correctly identifying artifacts out of all positive instances, while recall tells us how many are correctly identified as being an artifact out of all artifacts. The Precision-Recall curve shows the trade-off between these two measurements. A high area under the curve represents both high precision and high recall, with 1 being a perfect score [31].

The Receiver Operating Characteristic (ROC) curve and the Area Under the Curve (AUC) is a plot that shows the performance of a classification model and its ability to distinguish between classes [32, 33]. It plots the True Positive Rate (TPR) against the False Positive Rate (FPR) and can be defined as:

$$\text{TPR} = \frac{\text{True Positive}}{\text{True Positive} + \text{False Negatives}}$$

$$\text{FPR} = \frac{\text{False Positives}}{\text{True Negatives} + \text{False Positives}}$$

The higher the AUC score, the better the model is at differentiation between artifacts and non-artifacts, with 1 being a perfect score. An AUC score of 0.5 means that the classifier will randomly predict a class and is unable to distinguish between artifacts and non-artifacts.

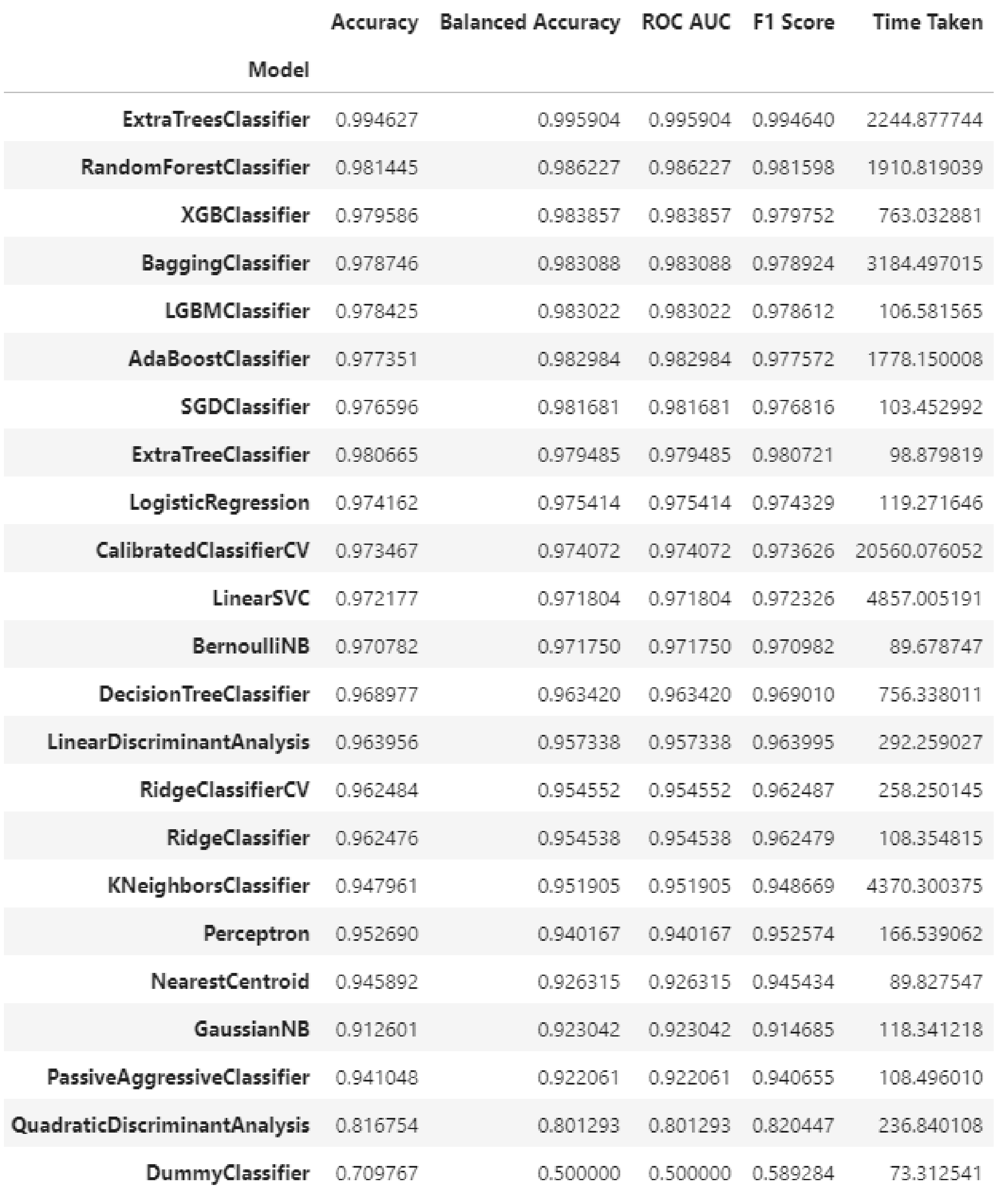
CHAPTER 3

Results

In the following sections, results from quality control, data exploration, feature importance, predictive power, and performance evaluation of model building are shown. Table 2 shows the case counts for each SRA accession number during Artifact Investigation, Chapter 2.3. The ranking of lazypredict models from Model Building, Chapter 2.4.4, can be seen in Fig. 8.

Table 2: Case Count

SRA accession number	Case 1	Case 2	Case 3	Case 4	Total No. of Artifacts
ERR1831349	31052	1	1764	9381	9382
ERR1831346	31218	0	1599	28157	28157
ERR1831347	31163	0	1654	20005	20005
ERR1831348	31183	0	1634	23751	23751
ERR1831350	31150	0	1667	18249	18249
ERR1831351	31159	2	1656	21227	21229
ERR1831352	31200	1	1616	27328	27329
ERR1831353	31167	1	1649	21628	21629
ERR1905889	31204	1	1612	67932	67933
ERR1905890	31227	3	1587	59447	59450
SRR14724463	24232	0	1641	93208	93208
SRR14724473	16130	0	1300	53557	53557
SRR14724483	18053	0	1022	64341	64341
SRR14724493	24105	1	1767	86657	86658
SRR14724503	16086	0	1344	55561	55561
SRR14724513	17928	0	1147	55983	55983
SRR2106342	49195	0	2535	93890	93890
SRR2106344	24309	1	1563	227400	227401
DRR189730	9621	14	8533	29687	29701
DRR189731	9568	45	8555	106551	106596
DRR189732	12233	46	12403	137042	137088



Model	Accuracy	Balanced Accuracy	ROC AUC	F1 Score	Time Taken
ExtraTreesClassifier	0.994627	0.995904	0.995904	0.994640	2244.877744
RandomForestClassifier	0.981445	0.986227	0.986227	0.981598	1910.819039
XGBClassifier	0.979586	0.983857	0.983857	0.979752	763.032881
BaggingClassifier	0.978746	0.983088	0.983088	0.978924	3184.497015
LGBMClassifier	0.978425	0.983022	0.983022	0.978612	106.581565
AdaBoostClassifier	0.977351	0.982984	0.982984	0.977572	1778.150008
SGDClassifier	0.976596	0.981681	0.981681	0.976816	103.452992
ExtraTreeClassifier	0.980665	0.979485	0.979485	0.980721	98.879819
LogisticRegression	0.974162	0.975414	0.975414	0.974329	119.271646
CalibratedClassifierCV	0.973467	0.974072	0.974072	0.973626	20560.076052
LinearSVC	0.972177	0.971804	0.971804	0.972326	4857.005191
BernoulliNB	0.970782	0.971750	0.971750	0.970982	89.678747
DecisionTreeClassifier	0.968977	0.963420	0.963420	0.969010	756.338011
LinearDiscriminantAnalysis	0.963956	0.957338	0.957338	0.963995	292.259027
RidgeClassifierCV	0.962484	0.954552	0.954552	0.962487	258.250145
RidgeClassifier	0.962476	0.954538	0.954538	0.962479	108.354815
KNeighborsClassifier	0.947961	0.951905	0.951905	0.948669	4370.300375
Perceptron	0.952690	0.940167	0.940167	0.952574	166.539062
NearestCentroid	0.945892	0.926315	0.926315	0.945434	89.827547
GaussianNB	0.912601	0.923042	0.923042	0.914685	118.341218
PassiveAggressiveClassifier	0.941048	0.922061	0.922061	0.940655	108.496010
QuadraticDiscriminantAnalysis	0.816754	0.801293	0.801293	0.820447	236.840108
DummyClassifier	0.709767	0.500000	0.500000	0.589284	73.312541

Fig. 8: Model Rankings from Lazypredict

3.1 Quality Control

Looking at SRR14724493 FastQC results show that 'Per base sequence content' and 'Adapter content' modules failed, as shown Fig. 9 and Fig. 10. This was the case for all other SRA accession numbers used, therefore the quality control and trimming were the same throughout this project.

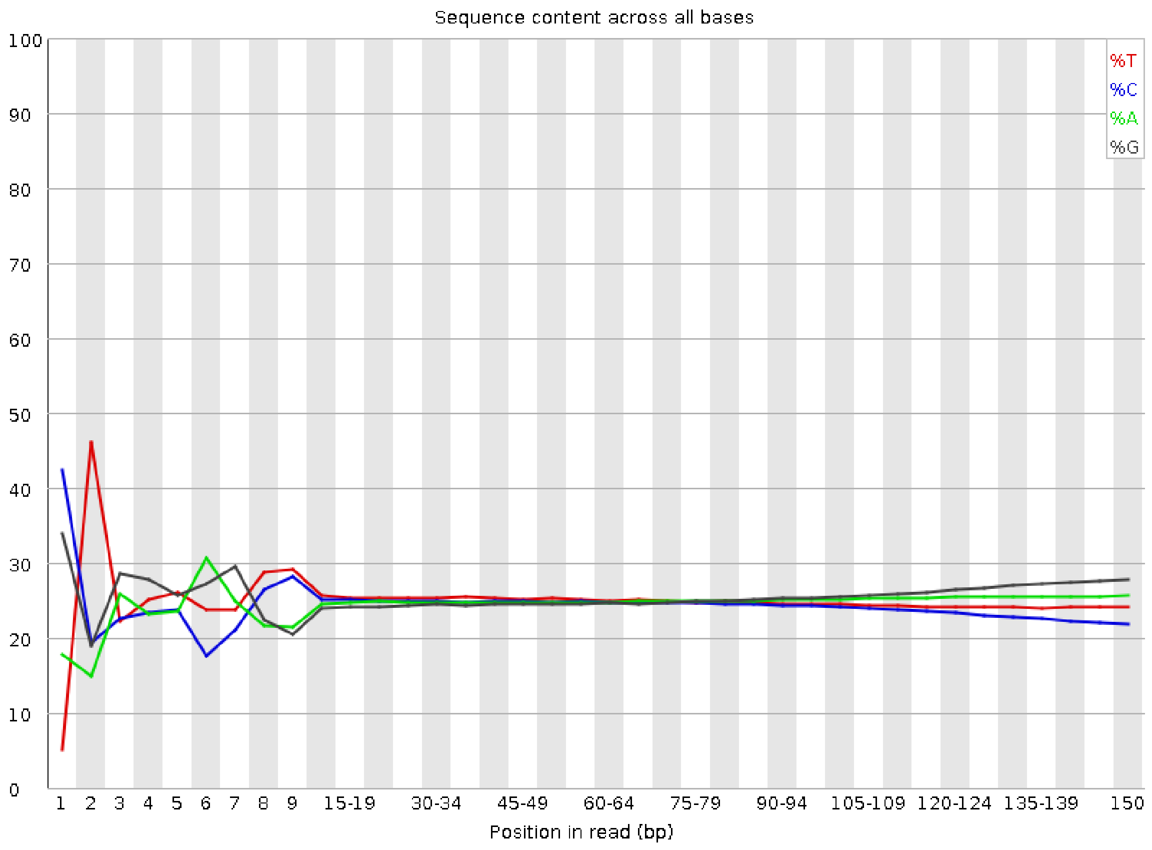


Fig. 9: Per base sequence content of raw SRR14724493 reverse read

After trimming the raw sequencing reads, the 'Per base sequence content' module still failed, while the 'Adapter content' module passed, as seen in Fig. 11 and Fig. 12.

The raw 'Per base sequence content' in Fig. 9 shows the first 9 bases have a lot of noise in the graph. There is also a horizontal line that diverges toward the right end. This graph plots the proportion of each nucleotide at each position across

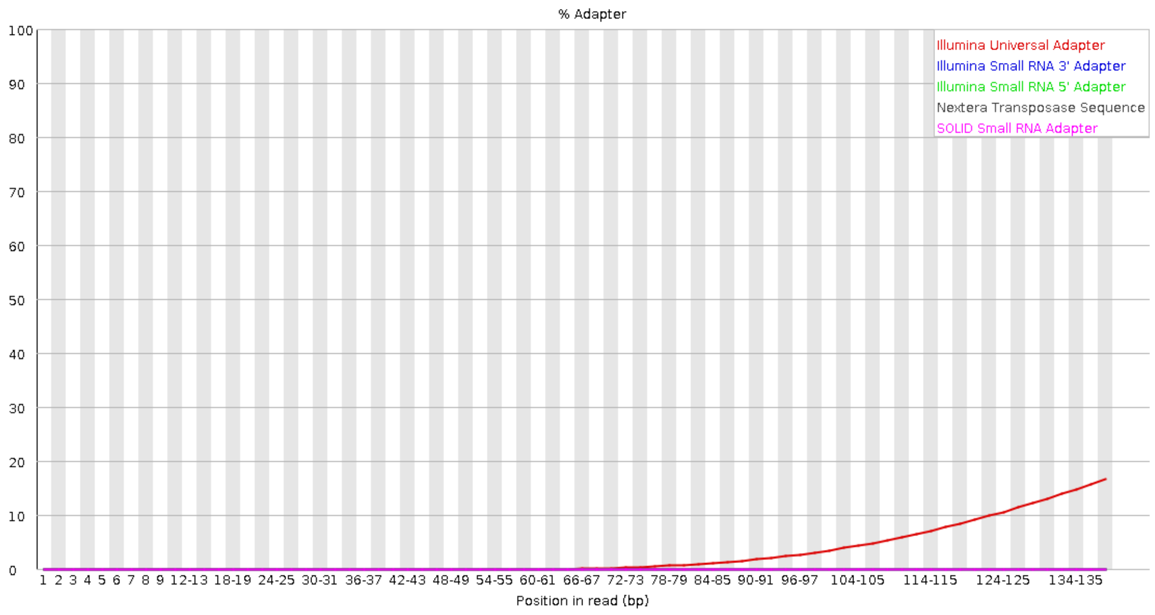


Fig. 10: Adapter content of raw SRR14724493 reverse read

all reads in the sequence file. In a random library, we expect little to no difference between the four bases. The proportion of each of the four bases should remain relatively constant over the length of the read, with A=T and G=C, and the lines in this plot should run parallel with each other [14]. It is common to produce biased sequence composition at the start of the read which typically does not negatively affect downstream analysis, therefore no trimming was done in this aspect. Libraries produced where fragmentation uses transposases will have an intrinsic bias in the positions at the start of the reads. Capture kits such as Nextera, Truseq rapid exome, and Kapa HyperPrep use transposase as their fragmentation method. This bias enriches the number of different kmers at the 5' end of the reads. It is not something that can be corrected by trimming, however, it will still produce a warning or failed module [14]. Since kmers are of interest in this project, trimming the first 5 bases may be helpful to see if this changes the kmer counts. The trimmed 'Per base sequence content' in Fig. 11 shows some improvement in the horizontal line, with essentially no

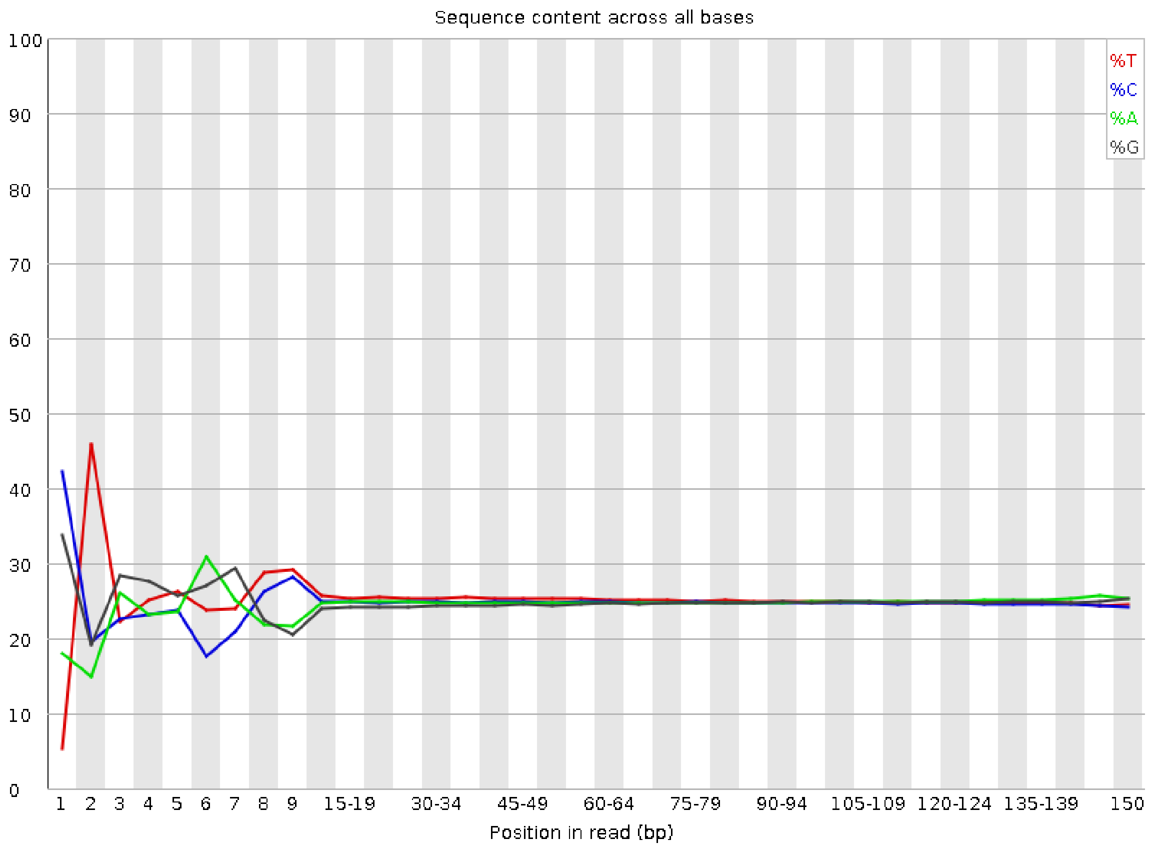


Fig. 11: Per base sequence content of trimmed SRR14724493 reverse read

divergence towards the right end. This slight improvement is a byproduct of adapter trimming.

The 'Adapter content' of the raw reads in Fig. 10 shows the presence of illumina universal adapter read-through building up on the end of the sequences. It is common for libraries that have insert sizes shorter than the read length to have adapter read-through [14]. The abundance of adapter read-through caused the module to fail. Adapter trimming needs to be done before proceeding with any downstream analysis [14]. Trimmomatic was used to remove these adapters from the reads. This was accomplished by providing Trimmomatic with the nucleotide sequence of the illumina universal adapter using the ILLUMINACLIP option.

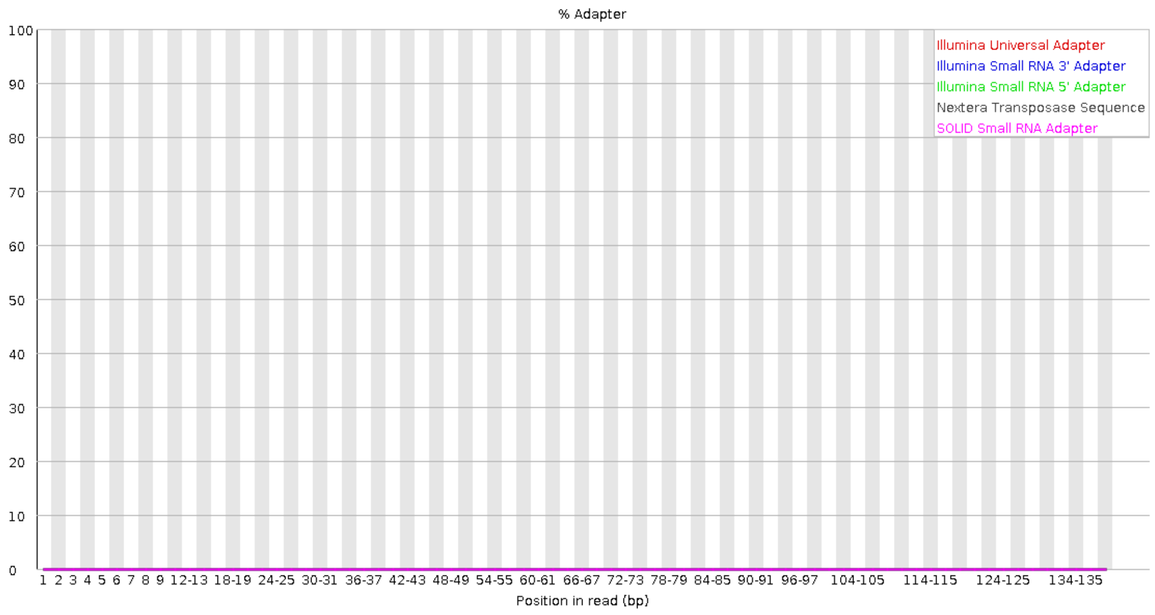


Fig. 12: Adapter content of trimmed SRR14724493 reverse read

3.2 Data Exploration

From a total of 1,834,281 variant calls 1,301,098 were artifacts, making up 70.94%. The heat map in Fig. 13 shows the number of each type of SNP. The base C being switched with the base A is the most prevalent, while a G being switched with an A is the second most prevalent. Purines are nucleotides A and G, which contain two-rings, while pyrimidines are nucleotides C and T, which contain one ring. The heat map shows transitions (purine to purine or pyrimidine to pyrimidine) being more common than transversions (purine to pyrimidine or pyrimidine to purine). Although there are twice as many possible transversions, transitions are more likely to occur in SNPs [34, 35]. Studies have shown that an excess of G to T transversion and C to A transversions are prevalent in sequencing errors due to DNA damage from the most commonly used DNA shearing technique, acoustic shearing, which can be visualized in the heat map. [8]

A heatmap of the percentage of each nucleotide in the flanking regions is shown

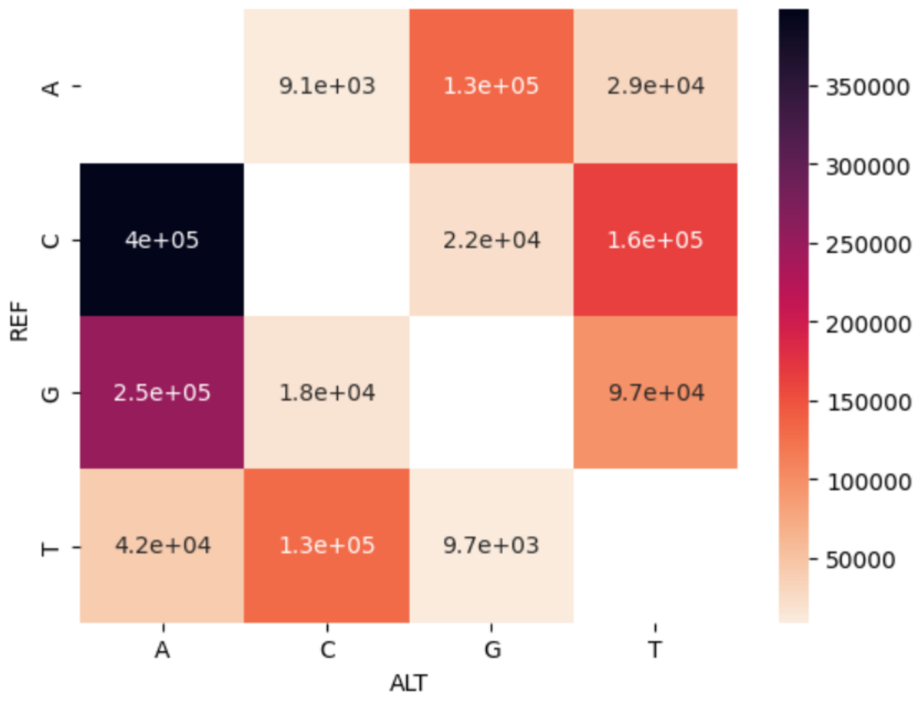


Fig. 13: Frequencies of Sequencing Artifact SNPs

in Fig. 14. There is a slightly higher percentage of C's and G's in these regions as opposed to A's and T's.

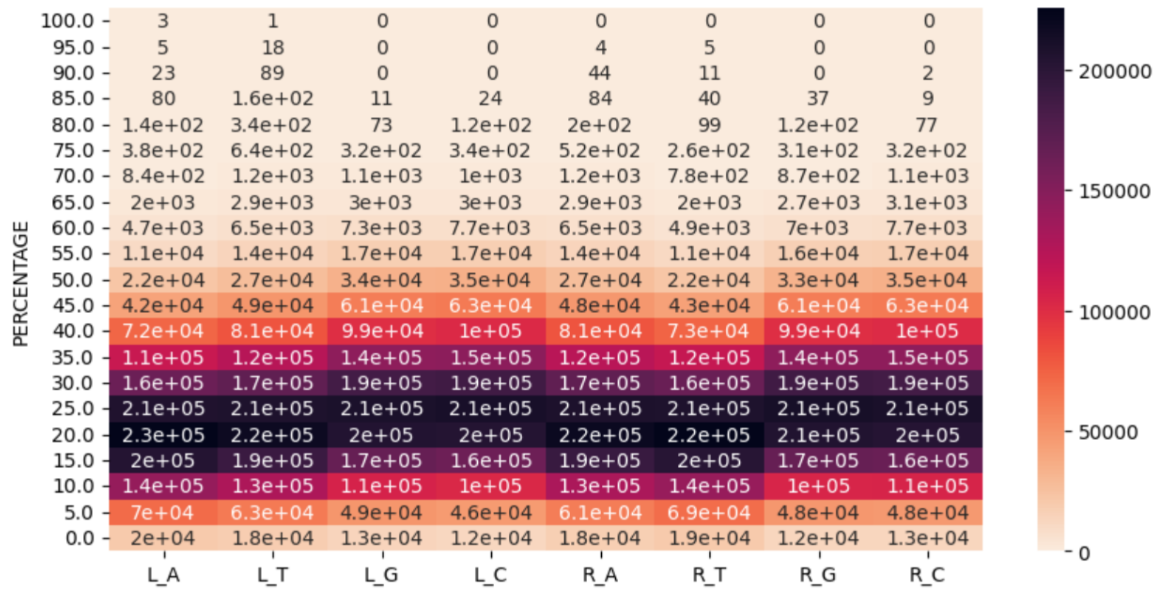


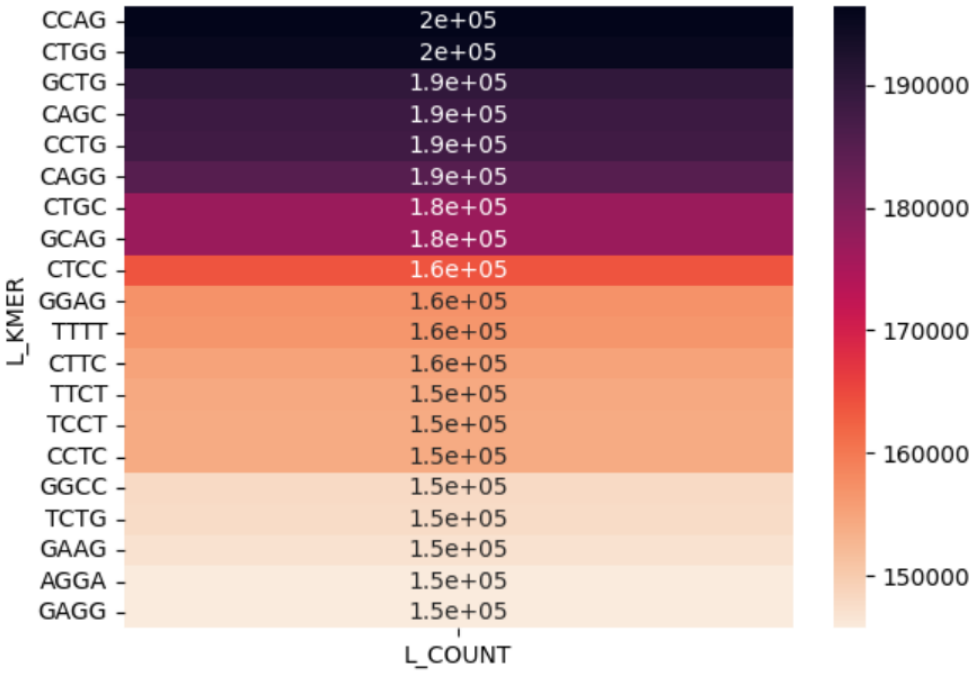
Fig. 14: Percent of Flanking Region Nucleotides

A heat map of the 20 most prevalent kmers of size 4 for LSEQ and RSEQ is shown in Fig. 15. Of these, 17 are shared among them, which include: CCAG, CTGG, GCTG, CAGC, CCTG, CAGG, CTGC, GCAG, CTCC, GGAG, CTTC, TCCT, CCTC, GGCC, GAAG, AGGA, and GAGG. Almost all of these kmers contain 3 C's and/or G's, with only 4 kmers containing 2 of these nucleotides and 1 kmer containing 4. There are some difficult regions to sequence DNA, with GC-rich regions being one of them [36]. There is no standard definition of what the lower bound of a GC-rich threshold is, but it cannot be lower than 50%. All of these kmers contain between 50-100% GC content. With difficult to sequence regions, DNA polymerase can make a mistake and slip a base, creating the sequencing errors present in these sequencing artifact flanking regions.

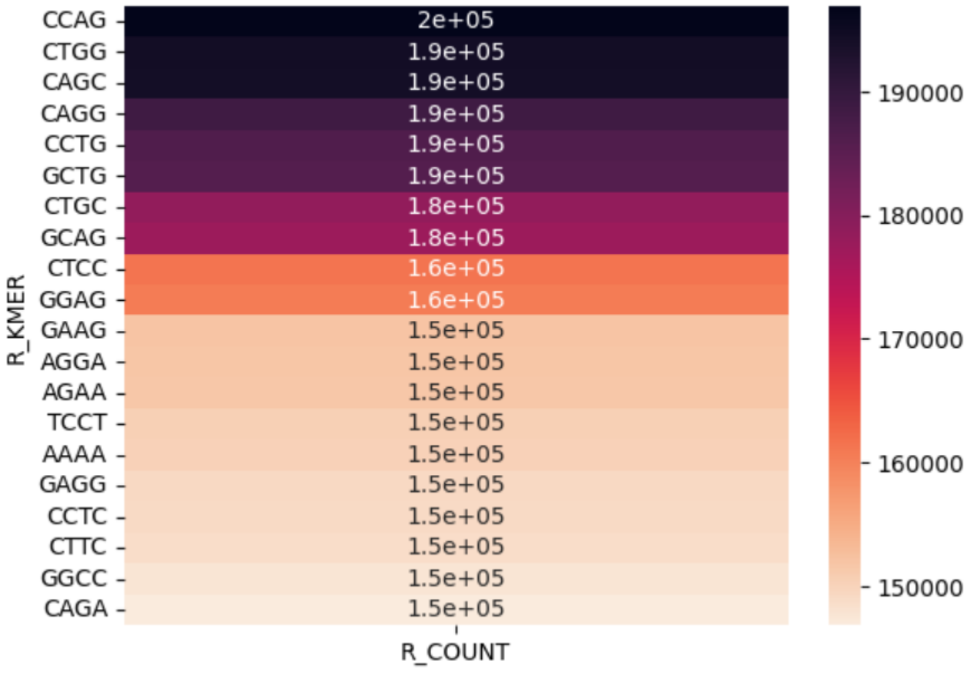
The count of each nucleotide combination flanking each SNP variation was generated on a heat map and shown in Fig. 16. Recall that for example, if there is a CCG, where the middle C represents the SNP, the nucleotide combination flanking this SNP would be CG. The image shows that CG and GC are the most prevalent. This is in line with GC-rich regions being hard to sequence areas, resulting in a slipped base.

A heat map of different homopolymer sizes and their counts are shown in Fig. 17. It is no surprise that smaller homopolymers are more prevalent, what is interesting though is that longer homopolymer sizes are particular to A's and T's. DNA sequences that contain long homopolymer stretches of A's or T's are difficult to sequence because they can hinder DNA polymerases during sequencing reactions and cause slippage and deterioration, thus creating sequencing errors [36, 37].

A heat map of different palindrome sizes and their counts are shown in Fig. 18. Smaller palindromic sequences are more prevalent than longer palindromic sequences. Palindromic sequences cause genomic instability and are susceptible to DNA breakage



(a) LSEQ 4mers



(b) RSEQ 4mers

Fig. 15: Frequencies of Flanking Region Kmers of size 4

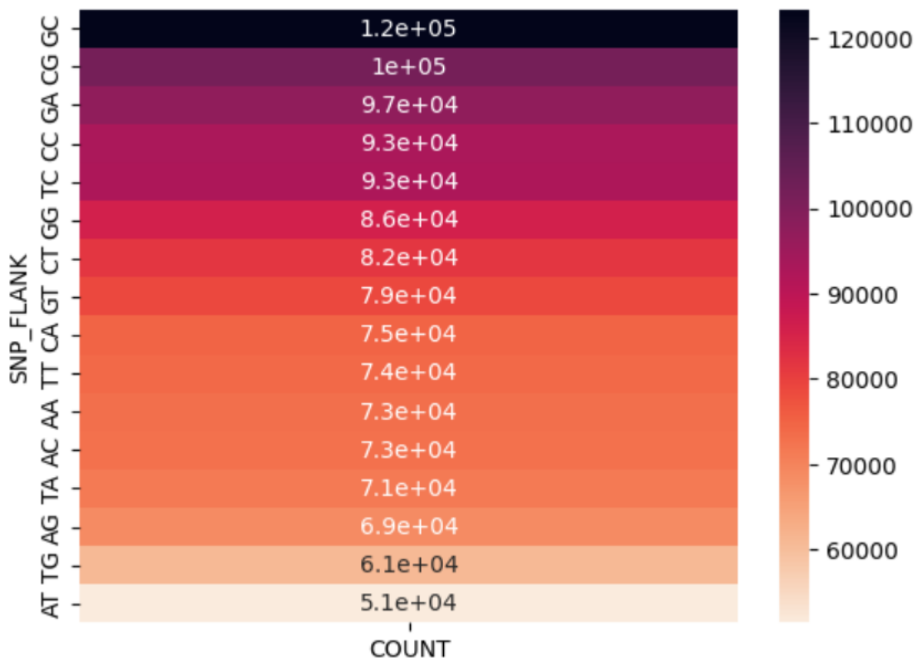


Fig. 16: Frequencies of Nucleotide Combination Flanking Each SNP

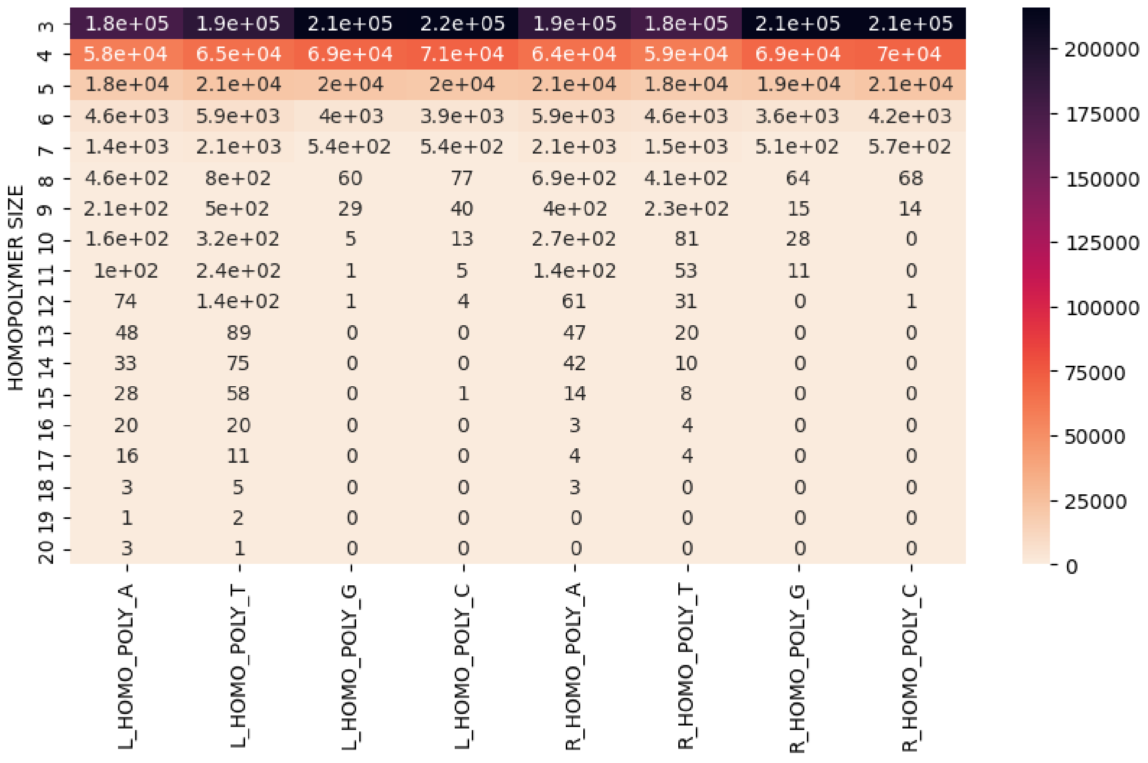


Fig. 17: Frequencies of Flanking Region Homopolymer Sizes

and mutations [2]. Because these regions are fragile and can break, the cell employs repair mechanisms that can result in genetic rearrangements, such as SNPs [38]. Palindromes are abundant in the human genome and are frequently found in cancers, which have low allele frequency. This can be an explanation as to why palindromes are present in the flanking regions and labeled as artifacts, they are low allelic mutations that do not qualify as variants.

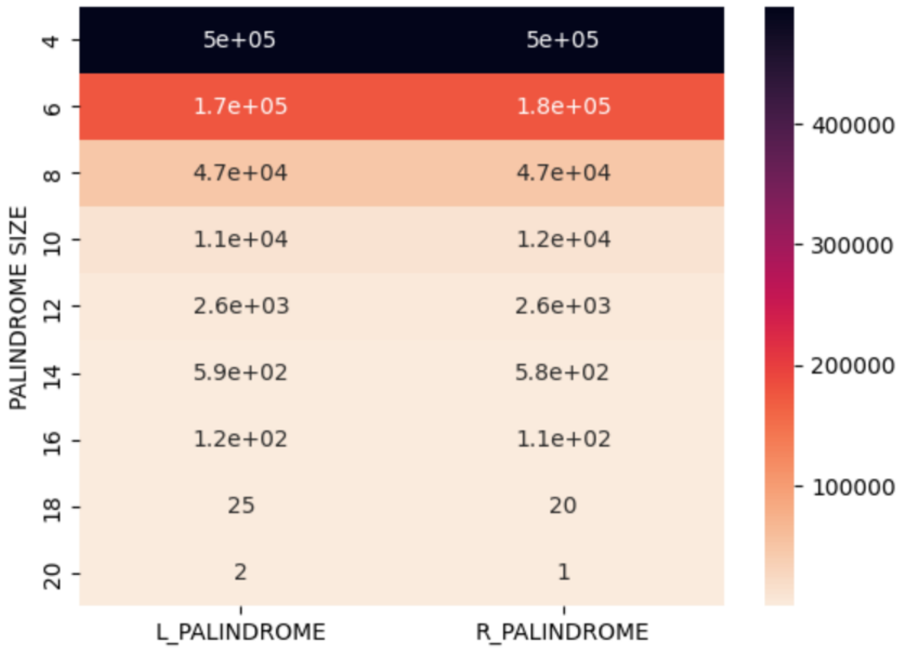


Fig. 18: Frequencies of Flanking Region Palindrome Sizes

A heat map of the different hairpin loop sizes and their counts are shown in Fig. 19. With palindromic regions present in the flanking regions, it is only natural to also see hairpin loops present since palindromes have the ability to fold back on themselves and form hairpin loops. A loop size of 4 or 5 nucleotides is the most stable, and a loop size of 7 is energetically the most favorable [22, 39]. Since these hairpin loop sizes are the most stable, they stick around and accumulate, which is precisely what the heat map shows.

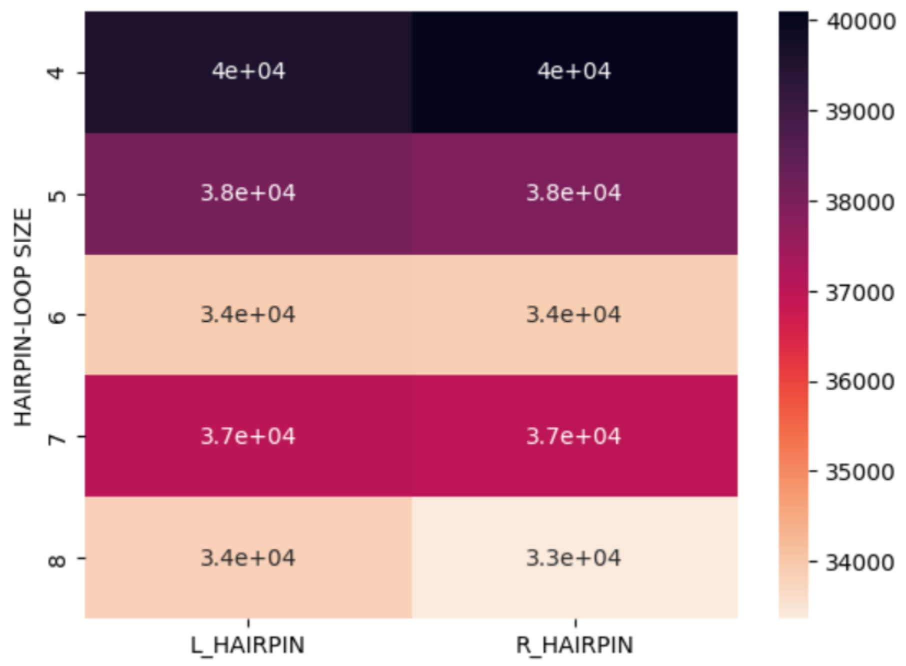


Fig. 19: Frequencies of Flanking Region Hairpin Loop Sizes

3.3 Feature Importance

After the top 3 models were determined through lazypredict, features were explored to determine feature importance. Random Forest was used for this experimental stage since it chooses the optimum split at each tree node as opposed to Extra Trees which chooses it randomly [40].

This was accomplished by randomly selecting 5 features and training 10000 models since the data frame has 540 features. This included the L features and R features representing the left and right flanking regions. Another experiment was conducted where the L and R features were combined in order to reduce the number of features. For nucleotide content and kmers, the L and R features were added together since these represented the counts or percentages of each. For homopolymer, palindrome, and hairpin, the max of the L and R features were taken since these represented the largest size of each. This data frame now has 274 features. Three features were randomly selected and 10000 models were trained to get the same coverage.

Feature importance is calculated by taking the average accuracy score for models trained on that feature. The combined L and R features performed slightly better so it was used for predictive power, training, and testing models. The Random Forest Classifier has the option to use bootstrap samples when building trees. Experiments were conducted with bootstrap samples and without, where the whole dataset is used to build each tree. Using bootstrap samples yielded better performance so bootstrap samples will be used for Random Forest on the training and testing of models. The top 33 important features are reported in Fig. 20.

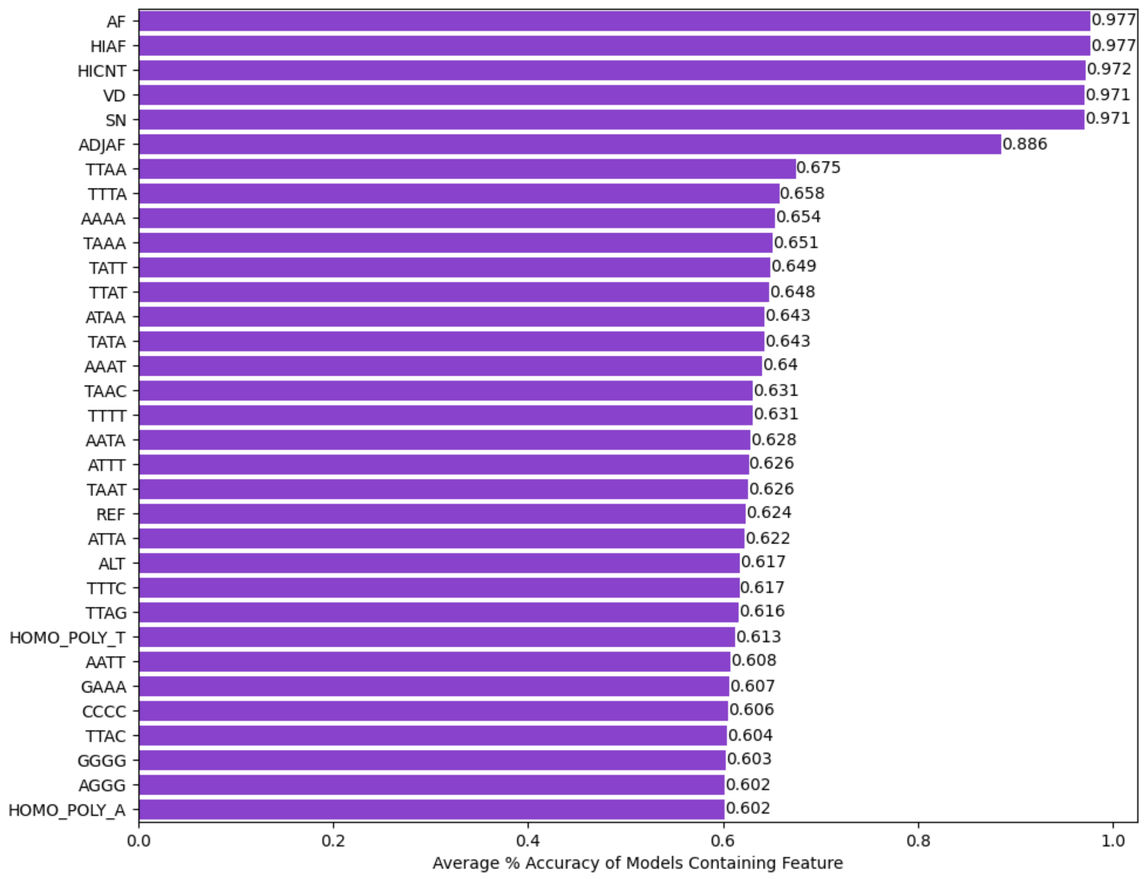


Fig. 20: Relative Importance of Model Features

3.4 Predictive Power

Each feature was removed sequentially in order of calculated feature importance. The top 33 feature removal statistics are shown in Table 3. The model drops in accuracy and recall substantially, scoring less than 70% when removing the top 6 features, which are features given in the VarDict VCF file. Precision, however, stays relatively high, scoring greater than 75%.

3.5 Performance Evaluation of Model Building

The overall accuracy, precision, and recall will be described in the next section for models built with all features and models built with only calculated LSEQ/RSEQ features. In the subsequent sections for each individual model and in the 'Experiments' section, confusion matrices and summaries will be provided using only models built with LSEQ/RSEQ features, as that is the focus of this project. With a robust enough model, Case 3 can be trained and tested. There is no variant call in the VarDict VCF for this tough case, but the LSEQ/RSEQ features can be extracted and calculated from the reference genome.

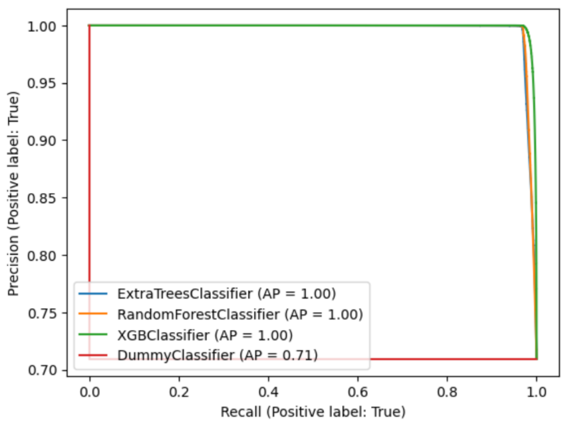
3.5.1 ROC Curves and Precision-Recall Curves

Though high accuracy, precision, and recall are important for a good model, precision is especially important in this case because we want to correctly identify all artifacts in the dataset. The Precision-Recall Curve for each model with all features and only the calculated LSEQ/RSEQ features are shown in Fig. 21. The average precision value for recall values from 0 to 1 (AP), is 1 for all features. This shows the results are highly accurate, while the AP value for only LSEQ/RSEQ features is between 0.87 and 0.93, which is still highly accurate. The Dummy Classifier is a no-skill classifier that predicts all cases as artifacts. Since the data is made up of 71% artifacts, the AP value is 0.71 here. The AP value being greater than the Dummy

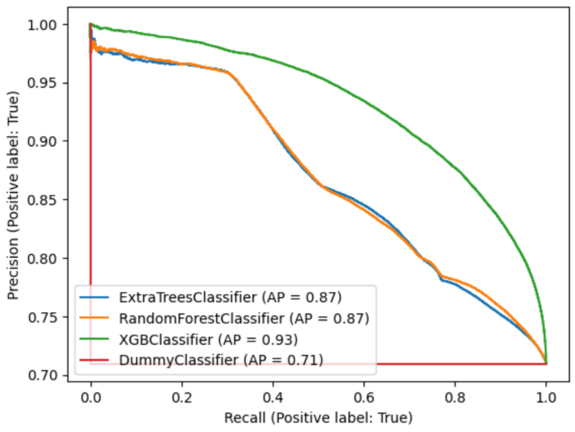
Table 3: Feature Removal Statistics

Number Feature Set	Accuracy	Precision	Recall
All Features	97.7%	99.9%	96.8%
Previous Row Minus ‘AF’	97.5%	99.7%	96.7%
Previous Row Minus ‘HIAF’	97.2%	99.6%	96.4%
Previous Row Minus ‘HICNT’	97.1%	99.5%	96.5%
Previous Row Minus ‘VD’	96.0%	98.8%	95.9%
Previous Row Minus ‘SN’	81.8%	92.2%	86.7%
Previous Row Minus ‘ADJAF’	66.6%	83.4%	64.7%
Previous Row Minus ‘TTAA’	66.9%	81.8%	69.4%
Previous Row Minus ‘TTTA’	65.8%	81.4%	69.4%
Previous Row Minus ‘AAAA’	65.7%	81.8%	68.4%
Previous Row Minus ‘TAAA’	65.5%	81.3%	69.6%
Previous Row Minus ‘TATT’	65.2%	81.8%	68.6%
Previous Row Minus ‘TTAT’	64.1%	82.5%	66.5%
Previous Row Minus ‘ATAA’	65.7%	81.5%	68.2%
Previous Row Minus ‘TATA’	66.1%	81.9%	68.6%
Previous Row Minus ‘AAAT’	66.4%	82.3%	67.8%
Previous Row Minus ‘TAAC’	65.2%	81.2%	69.8%
Previous Row Minus ‘TTTT’	65.1%	82.6%	66.5%
Previous Row Minus ‘AATA’	66.5%	81.0%	70.6%
Previous Row Minus ‘ATTT’	65.7%	81.5%	68.7%
Previous Row Minus ‘TAAT’	66.7%	81.4%	69.1%
Previous Row Minus ‘REF’	63.0%	82.5%	64.7%
Previous Row Minus ‘ATTA’	62.6%	83.6%	60.1%
Previous Row Minus ‘ALT’	59.2%	75.4%	63.3%
Previous Row Minus ‘TTTC’	59.1%	75.5%	63.3%
Previous Row Minus ‘TTAG’	58.7%	75.5%	63.1%
Previous Row Minus ‘HOMO_POLY_T’	59.0%	75.6%	62.9%
Previous Row Minus ‘AATT’	58.7%	75.5%	62.4%
Previous Row Minus ‘GAAA’	58.9%	75.6%	63.0%
Previous Row Minus ‘CCCC’	58.6%	75.7%	62.4%
Previous Row Minus ‘TTAC’	58.9%	75.5%	62.5%
Previous Row Minus ‘GGGG’	58.9%	75.9%	63.1%
Previous Row Minus ‘AGGG’	58.8%	75.7%	62.1%
Previous Row Minus ‘HOMO_POLY_A’	58.6%	75.7%	62.4%

Classifier means that those classifiers are able to distinguish between artifacts and non-artifacts.



(a) All features



(b) LSEQ/RSEQ features

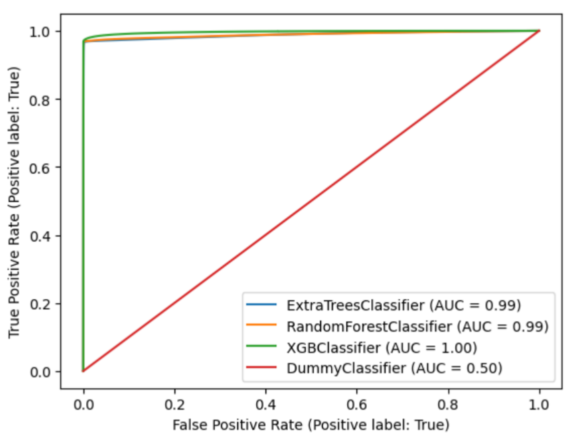
Fig. 21: Comparison of ML models Precision-Recall Curve

The ROC curve and AUC scores for each model with all features and only the calculated LSEQ/RSEQ features are shown in Fig. 22. The AUC scores for all features is between 0.99 and 1, indicating that all 3 models accurately predicted whether a call was an artifact or a non-artifact, while the AUC scores for only LSEQ/RSEQ features is between 0.72 and 0.85, which is still relatively accurate. Since the AUC scores are greater than 0.5, the classifiers can distinguish between the two classes and will detect more True Positives and True Negatives than False Positives and False Negatives [33].

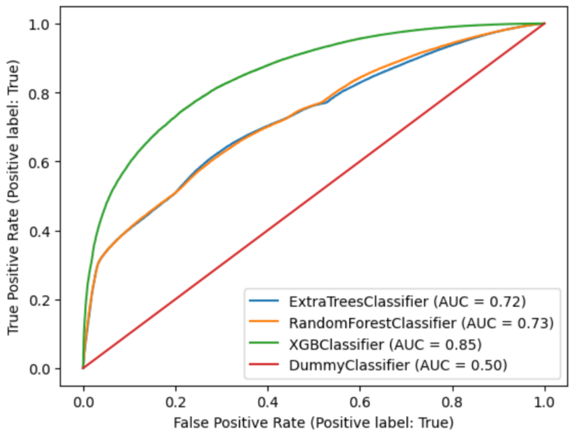
It is expected that the ROC and Precision-Recall curves for all features is better than the curves for only LSEQ/RSEQ features since all features include the top 6 features from the VarDict VCF file with the most predictive power.

3.5.2 Random Forest Classifier

The Random Forest Classifier is made up of many Decision Tree Classifiers on sub-samples of the dataset and takes the average to improve predictive accuracy in



(a) All features



(b) LSEQ/RSEQ features

Fig. 22: Comparison of ML models ROC Curve and AUC scores

classification [27]. The Random Forest model had a mean cross-validation score for accuracy of 66.9%. For the test set, it had an accuracy of 67.1%, precision of 81.2%, and recall of 69.8%. The confusion matrix and reported metrics are shown in Fig. 23 and Table 4.

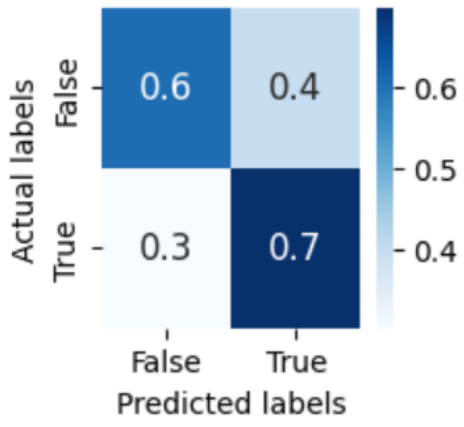


Fig. 23: Random Forest Classifier Confusion Matrix

3.5.3 Extra Trees Classifier

The Extra Trees Classifier is very similar to the Random Forest Classifier, except it fits randomized trees [26, 40]. This model is computationally faster since it randomly chooses the split points instead of calculating the optimal one, giving it

Table 4: Random Forest Classification Results

Metric	Score
Cross-validation Accuracy Scores	66.9%
	66.9%
	66.9%
Mean cross-validation score	66.9%
Test Accuracy	67.1%
Test Precision	81.2%
Test Recall	69.8%

the name ’Extremely Randomized Trees’ [40]. The Extra Trees model had a mean cross-validation score for accuracy of 67.6%. For the test set, it had an accuracy of 67%, precision of 80.6%, and recall of 71.5%. The confusion matrix and reported metrics are shown in Fig. 24 and Table 5.

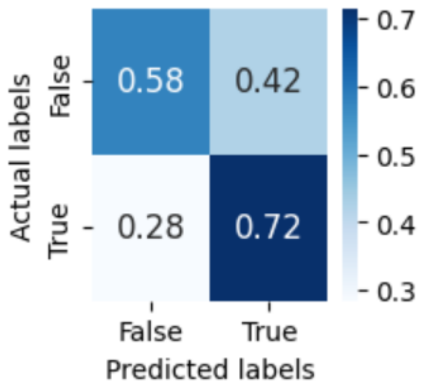


Fig. 24: Extra Trees Classifier Confusion Matrix

3.5.4 XGBoost Classifier

The XGBoost Classifier uses an optimized gradient boosting framework for parallel decision tree boosting [28]. Boosting is a method to reduce errors in predictive data analysis which improves the model’s predictive accuracy and performance; this is accomplished by converting several weak learners into a single strong learning model [41]. This model is known for improving on other methods like Random Forest,

Table 5: Extra Trees Classification Results

Metric	Score
Cross-validation Scores	66.7%
	67.2%
	67.2%
Mean cross-validation score	67.0%
Test Accuracy	67.6%
Test Precision	80.6%
Test Recall	71.5%

and works well with large datasets due to its optimization techniques [42]. The XGBoost model had a mean cross-validation score for accuracy of 73.1%. For the test set, it had an accuracy of 73%, precision of 72.6%, and recall of 99.6%. The confusion matrix and reported metrics are shown in Fig. 25 and Table 6.

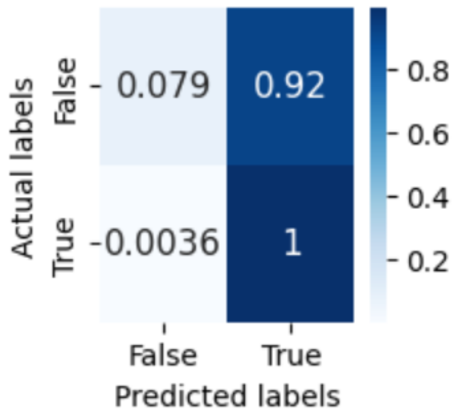


Fig. 25: XGBoost Classifier Confusion Matrix

3.5.5 Summary

A summary of each model's results is illustrated in Table 7.

3.5.6 Experiments

To try and improve the accuracy, precision, and recall of models containing only LSEQ/RSEQ features, several experiments were conducted.

Table 6: XGBoost Classification Results

Metric	Score
Cross-validation Scores	73.1%
	73.0%
	73.1%
Mean cross-validation score	73.1%
Test Accuracy	73.0%
Test Precision	72.6%
Test Recall	99.6%

Table 7: Classification Statistics Summary

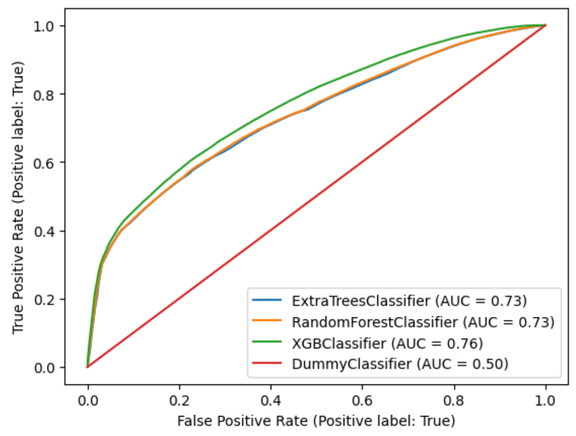
Model	Precision	Recall	Accuracy	Cross-validation Accuracy
Random Forest	81.2%	69.8%	67.1%	66.9%
Extra Trees	80.6%	71.5%	67.6%	67.0%
XGBoost	72.6%	99.6%	73.0%	73.1%

3.5.6.1 Remove features with less than 60% predictive power

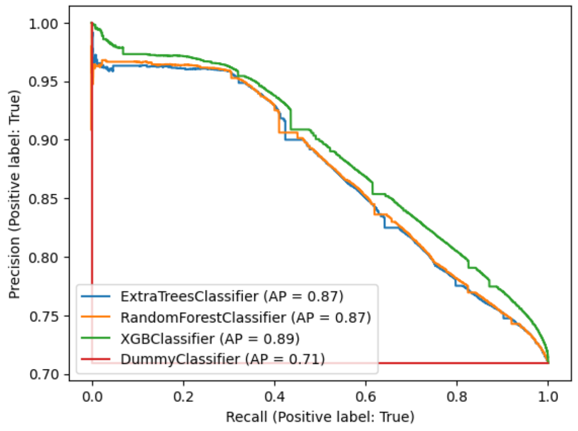
The same 3 models were trained on only features with greater than 60% predictive power. The ROC and Precision-Recall curves are illustrated in Fig. 26. The AUC score improved by 0.1 for Extra Trees and decreased by 0.9 for XGB, while the Precision-Recall decreased by 0.4 for XGB.

3.5.6.2 Clustering Features

Originally there were 540 features that made model building computationally expensive. ML is an iterative process that can take hours or even days to train a single model. This is dependent on factors such as hyperparameters, the number of features, and cross-validation folds. Previously the LSEQ and RSEQ features were combined, bringing the total features from 540 down to 274. The Python package VarClusHi performs variable clustering with a hierarchical structure, which allows similar features



(a) ROC Curve and AUC scores



(b) Precision-Recall Curve

Fig. 26: Comparison of ML models with LSEQ/RSEQ features greater than 60% predictive power

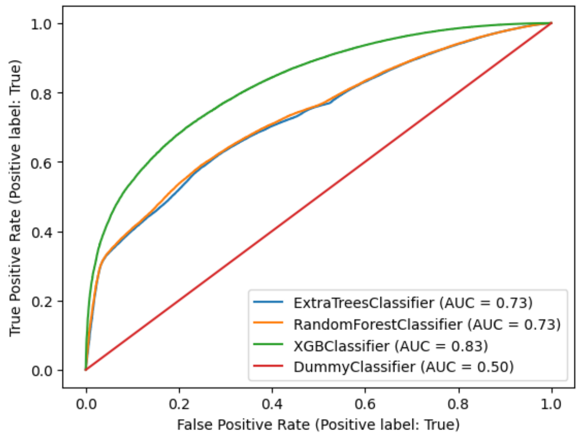
to be grouped [43]. Using this clustering information, the high dimension of features was further reduced to 163. This was accomplished by using 3/4 of the top predictive features from each cluster. Clustering features took roughly 2 hours, but with this shortened list of features, the downstream analysis will be sped up significantly. This allows tweaking of hyperparameters and experimenting with other models that previously were unavailable due to high dimensions.

The ROC and Precision-Recall curves are illustrated in Fig. 27. The AUC score improved by 0.1 for Extra Trees and decreased by 0.2 for XGB, while the Precision-Recall decreased by 0.1 for XGB.

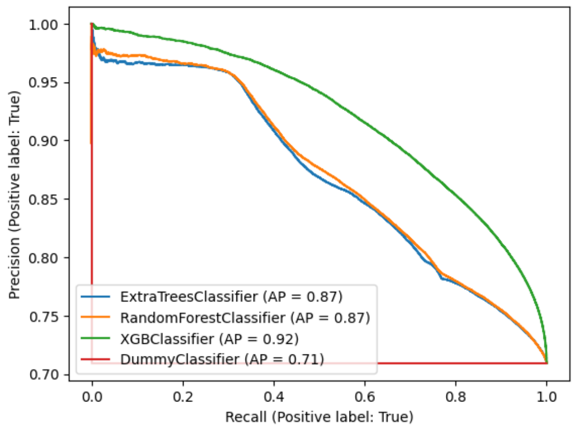
Since this shortened feature list did not greatly affect model performance, it will be used for downstream experiments.

3.5.6.3 Hyperparameter Tuning

With the shortened feature list, fine-tuning the Random Forest model was conducted with Randomized Search and Grid Search [44, 45]. Random Forest was chosen because it had the best precision score.



(a) ROC Curve and AUC scores



(b) Precision-Recall Curve

Fig. 27: Comparison of ML models after LSEQ/RSEQ features clustered with VarClusHi

With both of these hyperparameter tuning techniques, optimization is done by cross-validated search over parameter settings. The scoring metric used was precision and cross-validation with k=5. Two hyperparameters were experimented with, `max_features`, which is the number of features to consider when looking for the best split, and `n_estimators`, which is the number of trees in the forest.

Randomized Search was conducted first as it randomly chooses a combination of parameters from lists of given parameters. Initially, 10 iterations were specified with `max_features` being a random number chosen between 1 and 200; while `n_estimators` was a random number between 1 and 8. This resulted in the best parameters being `max_features=5` and `n_estimators=2`. To ensure these were actually the "best" parameters, Grid Search was conducted to include even more combinations of parameters, with these numbers included. With Grid Search, every combination of parameters is tested. There were 33 total combinations tested, with `max_features=[2,4,5,7,9,20,50,100,"sqrt",None]` and `n_estimators=[2,10,20]`. This resulted in the best parameters being `max_features=None` and `n_estimators=2`. The `max_features=None` means that `max_features=n_features`, in this case, 163 after feature clustering with VarClusHi. The Random Forest classification results before and after hyperparameter tuning are illustrated in Table 8. All metrics are improved significantly after fine-tuning the model. The mean cross-validation score for accuracy is 87.7%, a 20.8% improvement. For the test set, we see a precision of 98.8%, recall of 87.3%, and accuracy of 90.2%. These are 17.6%, 17.5%, and 23.1% improvements. The cross-validation accuracy is similar to the test set accuracy, showing that the model is not overfitting.

Table 8: Random Forest Classification Results Pre and Post Hyperparameter Tuning

Hyperparameter Tuning	Precision	Recall	Accuracy	Cross-validation Accuracy
Before	81.2%	69.8%	67.1%	66.9%
After	98.8%	87.3%	90.2%	87.7%

CHAPTER 4

Discussion

This section will cover the work done thus far and future work to be done. All files discussed in this paper are available on Github, as well as additional scripts created in the development of this project [46].

4.1 SNP Flanking Sequence Metrics

WES pair-end reads from SRA were mapped to a reference genome and variants were called. The variants were compared with GIAB NA12878 gold standard to determine if the calls were true variants or artifactual variants. This project focused on SNPs of Case 2 and Case 4. The LSEQ and RSEQ metrics such as base content, homopolymer sizes, kmers, palindrome sizes, and hairpin loop sizes were determined so that they can be used to train models.

The relative abundance of base content, homopolymer sizes, kmers, palindrome sizes, and hairpin loop sizes in the human exonic regions was calculated. The counts in the flanking regions are compared to the observed frequencies in the exonic regions. `Seqtk` is a command line tool used to process sequences in FASTA or FASTQ formats [47]. This tool was used to extract the nucleotide sequences of the exonic regions of the human genome. Since a majority of the exome capture kits from this project were from Agilent SureSelect v5, that was the BED file used.

The percent of nucleotides in the exonic regions is shown in Appendix Fig. A.28. The exonic regions show a slightly lower percentage of G and C but the flanking regions show a higher percentage of G and C, Fig. 14. This higher percentage of G and C in the flanking region nucleotides could be a characteristic of sequencing artifacts.

A heat map of the 20 most prevalent kmers of size 4 in the exonic region is shown in Appendix Fig. A.29. Previously, we saw in Fig. 15 that the flanking regions share 17 kmers, which were GC-rich. The exonic region shares 14 kmers with the flanking regions, which include: CTGG, CCAG, CCTG, CAGG, CAGC, GCTG, CTGC, GCAG, CTCC, GGAG, TCCT, AGGA, CTTC, and GAAG. All of these kmers are GC-rich as well, but since the flanking region and exonic region share so many of the same kmers, the distribution we see in the flanking regions could just be because it is a subsample of the exonic region.

A heat map of different homopolymer sizes and their counts in the exonic region is shown in Appendix Fig. A.30. Previously, we saw in Fig. 17 that longer homopolymer sizes in the flanking regions are particular to A's and T's. In the exonic regions there is a huge abundance of long homopolymer A's and T's, so the distribution we see in the flanking regions could just be because it is a subsample of the exonic region.

A heat map of different palindrome sizes and their counts in the exonic region is shown in Appendix Fig. A.31. Previously, we saw in Fig. 18 that shorter palindromic sequences are more prevalent than longer palindromic sequences. This is the same trend we see in the exonic regions, which could just be because the flanking region is a subsample of the exonic region.

A heat map of different hairpin loop sizes and their counts in the exonic region is shown in Appendix Fig. A.32. Previously, we saw in Fig. 19 that loop sizes of 4, 5, and 7 are the most stable or favorable, so they were the most prevalent. In the exonic regions, loop sizes 8 and 4 are the most prevalent. A loop size of 4 is the most common class, while a loop size of 8 is the second most common class [22]. We did not see this trend in the flanking region, but we do see it in the exonic regions. A hairpin loop size of 5 and 7 may be characteristics of sequencing artifacts.

The counts in the flanking sequences metrics were compared to the observed frequencies in the exonic regions of the human genome and inferences were made. However, a statistical test such as T-test or KS-test can be used to determine if the LSEQ metrics are significantly different from the RSEQ metrics, or if the flanking metrics as a whole are significantly different than the exonic region metrics.

Extra Trees, Random Forest, and XGBoost Classifiers were built since they were the top 3 models from lazypredict. For precision, Random Forest and Extra Trees performed similarly, with a score of 81.2% and 80.6%, while XGBoost had a score of 72.6%. Likewise, for recall, Random Forest had 69.8%, Extra Trees had 71.5%, while XGBoost had 99.6%. For accuracy, all 3 models performed within 67.1% and 73%.

To get the final list of features, LSEQ and RSEQ features were joined together to reduce the number of features from 540 to 274. VarClusHi was used to cluster similar features, which reduced the dimensions to 163. Hyperparameter tuning on the Random Forest model lead to improved model performance for all metrics. The mean cross-validation score for accuracy is 87.7%, a 20.8% improvement. The test set precision of 98.8%, recall of 87.3%, and accuracy of 90.2% showed 17.6%, 17.5%, and 23.1% improvements.

4.2 Case 3 can be considered in the future

The Support Vector Machine (SVM) Classifier is robust but took a long time to model because of the high dimensions. After feature reduction with VarClusHi, this model may be tested. Some other techniques to try are Principal Component Analysis (PCA) for dimensionality reduction prior to classification or K-means clustering prior to classification.

The Random Forest classifier performed very well. The next step would be to model Case 3 and calculate performance metrics. Steps were already taken to differentiate Case 3.1 (non-artifacts) from Case 3.2 (artifacts). The LSEQ and RSEQ sequences need to be extracted from the reference genome so that metrics can be calculated. Then training and testing the Random Forest model on these cases can be done.

LIST OF REFERENCES

[1] iRepertoire, ‘‘Ngs overview: from sample to sequencer to results,’’ https://irepertoire.com/ngs-overview-from-sample-to-sequencer-to-results/, (Accessed on 10/24/2022).

[2] M. K. Ganapathiraju, S. Subramanian, S. Chaparala, and K. B. Karunakaran, ‘‘A reference catalog of dna palindromes in the human genome and their variations in 1000 genomes,’’ *Human genome variation*, vol. 7, no. 1, p. 40, 2020.

[3] H. Nakagawa and M. Fujita, ‘‘Whole genome sequencing analysis for cancer genomics and precision medicine,’’ *Cancer science*, vol. 109, no. 3, pp. 513--522, 2018.

[4] S.-K. Low, H. Zembutsu, and Y. Nakamura, ‘‘Breast cancer: The translation of big genomic data to cancer precision medicine,’’ *Cancer science*, vol. 109, no. 3, pp. 497--506, 2018.

[5] S. Morganti, P. Tarantino, E. Ferraro, P. D’Amico, G. Viale, D. Trapani, B. A. Duso, and G. Curigliano, ‘‘Complexity of genome sequencing and reporting: next generation sequencing (ngs) technologies and implementation of precision medicine in real life,’’ *Critical reviews in oncology/hematology*, vol. 133, pp. 171--182, 2019.

[6] S. Yohe and B. Thyagarajan, ‘‘Review of clinical next-generation sequencing,’’ *Archives of pathology & laboratory medicine*, vol. 141, no. 11, pp. 1544--1557, 2017.

[7] Z. Lai, A. Markovets, M. Ahdesmaki, B. Chapman, O. Hofmann, R. McEwen, J. Johnson, B. Dougherty, J. C. Barrett, and J. R. Dry, ‘‘Vardict: a novel and versatile variant caller for next-generation sequencing in cancer research,’’ *Nucleic acids research*, vol. 44, no. 11, pp. e108--e108, 2016.

[8] L. Chen, P. Liu, T. C. Evans Jr, and L. M. Ettwiller, ‘‘Dna damage is a pervasive cause of sequencing errors, directly confounding variant identification,’’ *Science*, vol. 355, no. 6326, pp. 752--756, 2017.

[9] N. C. f. B. I. National Library of Medicine (US), ‘‘Home - sra - ncbi,’’ https://www.ncbi.nlm.nih.gov/sra/, (Accessed on 05/15/2022).

[10] J. M. Zook, J. McDaniel, N. D. Olson, J. Wagner, H. Parikh, H. Heaton, S. A. Irvine, L. Trigg, R. Truty, C. Y. McLean, *et al.*, ‘‘An open resource for accurately benchmarking small variant and reference calls,’’ *Nature biotechnology*, vol. 37, no. 5, pp. 561--566, 2019.

[11] J. M. Zook, B. Chapman, J. Wang, D. Mittelman, O. Hofmann, W. Hide, and M. Salit, ‘‘Integrating human sequence data sets provides a resource of benchmark snp and indel genotype calls,’’ *Nature biotechnology*, vol. 32, no. 3, pp. 246--251, 2014.

[12] F. Mölder, K. P. Jablonski, B. Letcher, M. B. Hall, C. H. Tomkins-Tinch, V. Sochat, J. Forster, S. Lee, S. O. Twardziok, A. Kanitz, *et al.*, ‘‘Sustainable data analysis with snakemake,’’ *F1000Research*, vol. 10, 2021.

[13] B. H. P. C. at the NIH, ‘‘Sra-toolkit,’’ https://hpc.nih.gov/apps/sratoolkit.html, (Accessed on 05/15/2022).

[14] B. Bioinformatics, ‘‘Fastqc,’’ https://www.bioinformatics.babraham.ac.uk/projects/fastqc/, (Accessed on 05/15/2022).

[15] A. M. Bolger, M. Lohse, and B. Usadel, ‘‘Trimmomatic: a flexible trimmer for illumina sequence data,’’ *Bioinformatics*, vol. 30, no. 15, pp. 2114--2120, 2014.

[16] Illumina, ‘‘Adapter trimming: Why are adapter sequences trimmed from only the 3’ ends of reads?’’ https://support.illumina.com/bulletins/2016/04/adapter-trimming-why-are-adapter-sequences-trimmed-from-only-the--ends-of-reads.html, (Accessed on 10/24/2022).

[17] H. Li and R. Durbin, ‘‘Fast and accurate long-read alignment with burrows--wheeler transform,’’ *Bioinformatics*, vol. 26, no. 5, pp. 589--595, 2010.

[18] H. Li, B. Handsaker, A. Wysoker, T. Fennell, J. Ruan, N. Homer, G. Marth, G. Abecasis, and R. Durbin, ‘‘The sequence alignment/map format and samtools,’’ *bioinformatics*, vol. 25, no. 16, pp. 2078--2079, 2009.

[19] B. Institute, ‘‘Picard toolkit,’’ https://broadinstitute.github.io/picard/, (Accessed on 02/28/2023).

[20] A. R. Quinlan and N. Kindlon, ‘‘bedtools: a powerful toolset for genome arithmetic,’’ https://bedtools.readthedocs.io/en/latest/index.html, (Accessed on 10/24/2022).

[21] Y. Leong, ‘‘Modeling sequencing artifacts for next generation sequencing,’’ 2022.

[22] D. R. Groebe and O. C. Uhlenbeck, ‘‘Characterization of rna hairpin loop stability,’’ *Nucleic Acids Research*, vol. 16, no. 24, pp. 11 725--11 735, 1988.

[23] Scikit-learn, ‘‘sklearn.preprocessing.standardscaler,’’ https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html?highlight=standardscaler#sklearn.preprocessing.StandardScaler, (Accessed on 03/27/2023).

[24] Scikit-learn, "sklearn.preprocessing.onehotencoder," https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.OneHotEncoder.html, (Accessed on 03/27/2023).

[25] S. Pandala, "lazypredict," https://github.com/shankarpandala/lazypredict, (Accessed on 04/13/2023).

[26] Scikit-learn, "sklearn.ensemble.extratreesclassifier," https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.ExtraTreesClassifier.html, (Accessed on 04/13/2023).

[27] Scikit-learn, "sklearn.ensemble.randomforestclassifier," https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html#sklearn.ensemble.RandomForestClassifier, (Accessed on 04/13/2023).

[28] dmlc XGBoost, "Xgboost parameters," https://xgboost.readthedocs.io/en/stable/parameter.html, (Accessed on 04/13/2023).

[29] T. D. Science, "Why weight? the importance of training on balanced datasets," https://towardsdatascience.com/why-weight-the-importance-of-training-on-balanced-datasets-f1e54688e7df, (Accessed on 03/27/2023).

[30] Scikit-learn, "sklearn.metrics.confusion_matrix," https://scikit-learn.org/stable/modules/generated/sklearn.metrics.confusion_matrix.html, (Accessed on 04/13/2023).

[31] Scikit-learn, "sklearn.metrics.precisionrecalldisplay," https://scikit-learn.org/stable/modules/generated/sklearn.metrics.PrecisionRecallDisplay.html, (Accessed on 04/13/2023).

[32] Scikit-learn, "sklearn.metrics.roc_auc_score," https://scikit-learn.org/stable/modules/generated/sklearn.metrics.roc_auc_score.html, (Accessed on 04/13/2023).

[33] A. Vidhya, "Guide to auc roc curve in machine learning: What is specificity?" https://www.analyticsvidhya.com/blog/2020/06/auc-roc-curve-machine-learning/, (Accessed on 04/13/2023).

[34] D. M. Lyons and A. S. Lauring, "Evidence for the selective basis of transition-to-transversion substitution bias in two rna viruses," *Molecular biology and evolution*, vol. 34, no. 12, pp. 3205--3215, 2017.

[35] S. M. Carr, "Transition versus transversion mutations," https://www.mun.ca/biology/scarr/Transitions_vs_Transversions.html, (Accessed on 03/27/2023).

[36] J. Kieleczawa, "Fundamentals of sequencing of difficult templates—an overview," *Journal of biomolecular techniques: JBT*, vol. 17, no. 3, p. 207, 2006.

[37] J. Langan, L. Rowbottom, T. Liloglou, J. Field, and J. Risk, ‘‘Sequencing of difficult templates containing poly (a/t) tracts: Closure of sequence gaps,’’ *BioTechniques*, vol. 33, no. 2, pp. 276--280, 2002.

[38] M. Svetec Miklenić and I. K. Svetec, ‘‘Palindromes in dna—a risk for genome stability and implications in cancer,’’ *International Journal of Molecular Sciences*, vol. 22, no. 6, p. 2840, 2021.

[39] C. Hilbers, M. Blommers, C. Haasnoot, G. Van der Marel, and J. Van Boom, ‘‘Struktur und faltung von dna und rna hairpins,’’ *Fresenius' Zeitschrift für analytische Chemie*, vol. 327, pp. 70--70, 1987.

[40] P. Aznar, ‘‘What is the difference between extra trees and random forest?’’ https://quantdare.com/what-is-the-difference-between-extra-trees-and-random-forest/, (Accessed on 04/13/2023).

[41] aws, ‘‘What is boosting?’’ https://aws.amazon.com/what-is/boosting/, (Accessed on 04/13/2023).

[42] esri, ‘‘How xgboost algorithm works,’’ https://pro.arcgis.com/en/pro-app/latest/tool-reference/geoai/how-xgboost-works.htm, (Accessed on 04/13/2023).

[43] X. Jing, ‘‘Varclushi,’’ https://github.com/jingtt/varclushi, (Accessed on 04/13/2023).

[44] Scikit-learn, ‘‘sklearn.model_selection.randomizedsearchvc,’’ https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.RandomizedSearchCV.html, (Accessed on 04/13/2023).

[45] Scikit-learn, ‘‘sklearn.model_selection.gridsearchcv,’’ https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html, (Accessed on 04/13/2023).

[46] K. Lam, ‘‘sequencing_artifacts,’’ https://github.com/kathylambchops/sequencing_artifacts, (Accessed on 05/23/2023).

[47] H. Li, ‘‘seqtk,’’ https://github.com/lh3/seqtk, (Accessed on 04/13/2023).

APPENDIX

Exonic Region Statistics

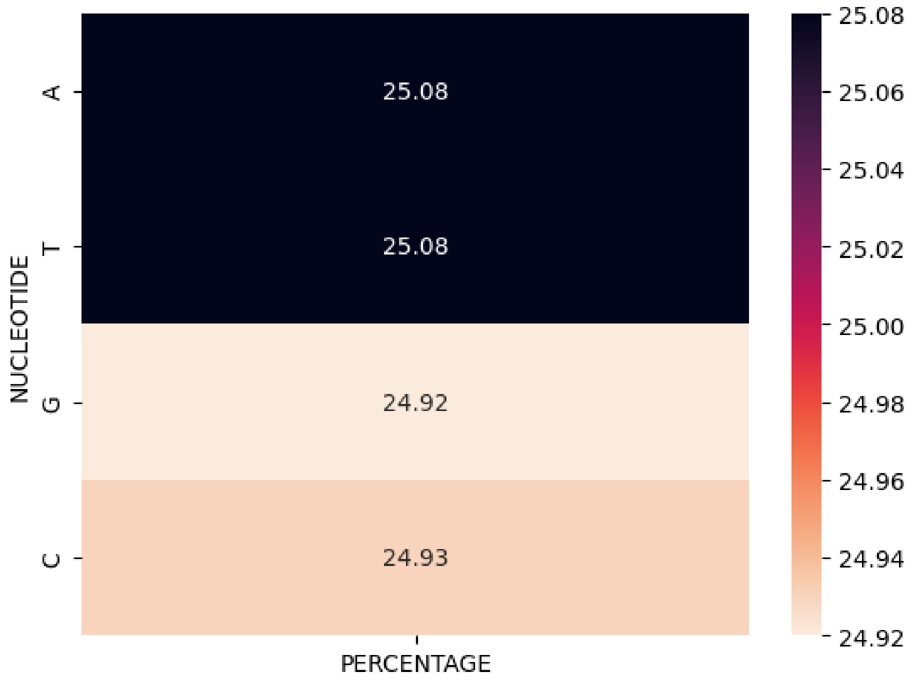


Fig. A.28: Percent of Exonic Region Nucleotides

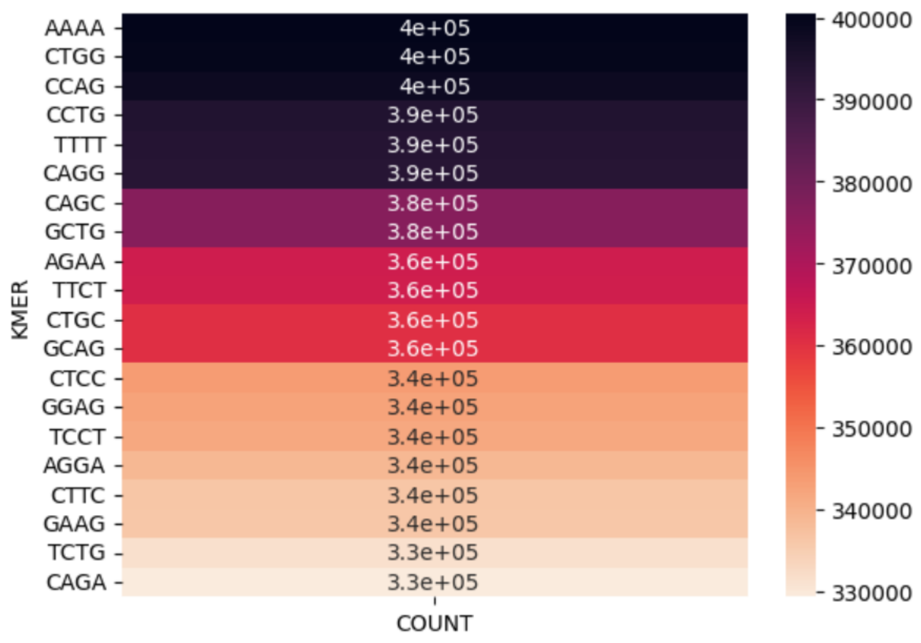


Fig. A.29: Frequencies of Exonic Region Kmers of size 4

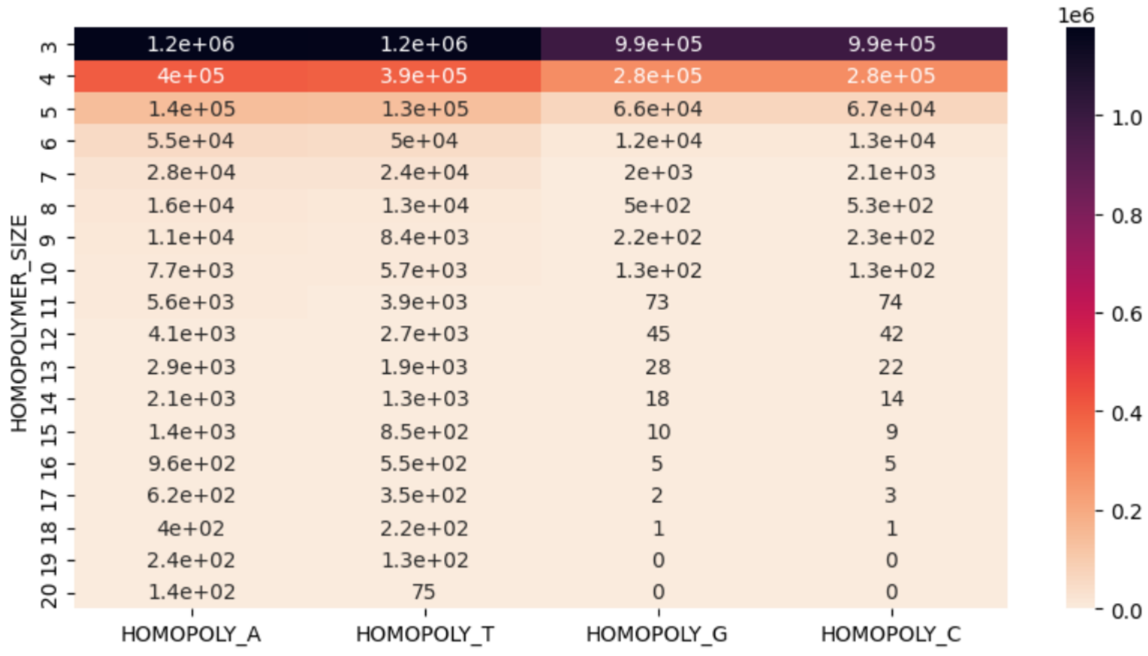


Fig. A.30: Frequencies of Exonic Region Homopolymer Sizes

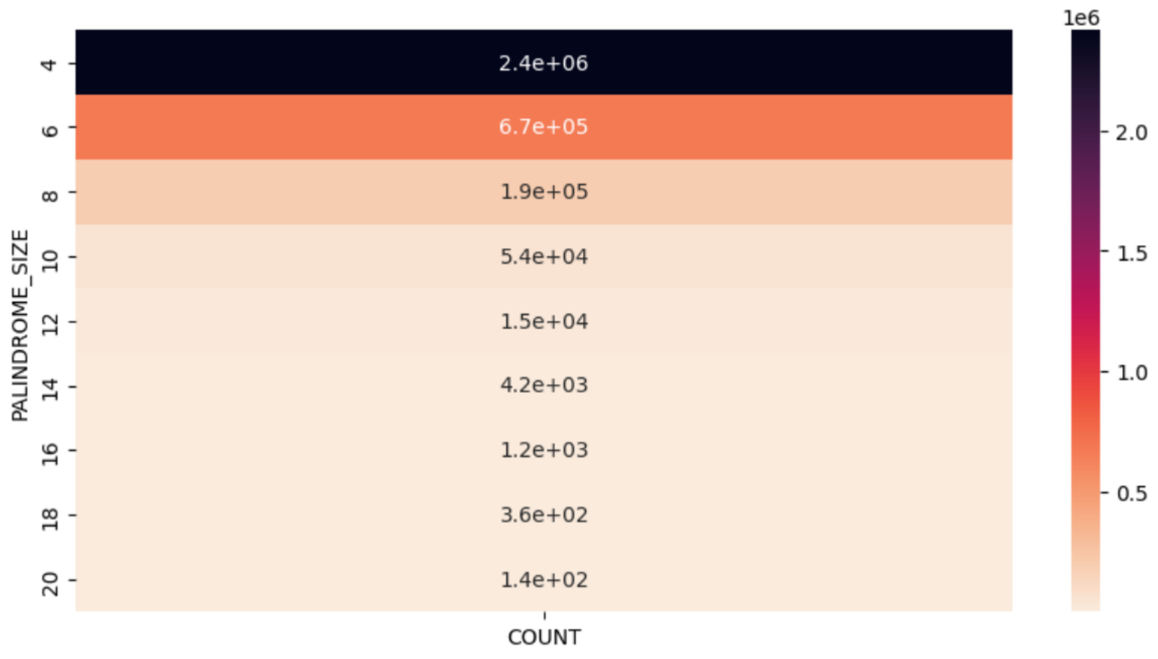


Fig. A.31: Frequencies of Exonic Region Palindrome Sizes

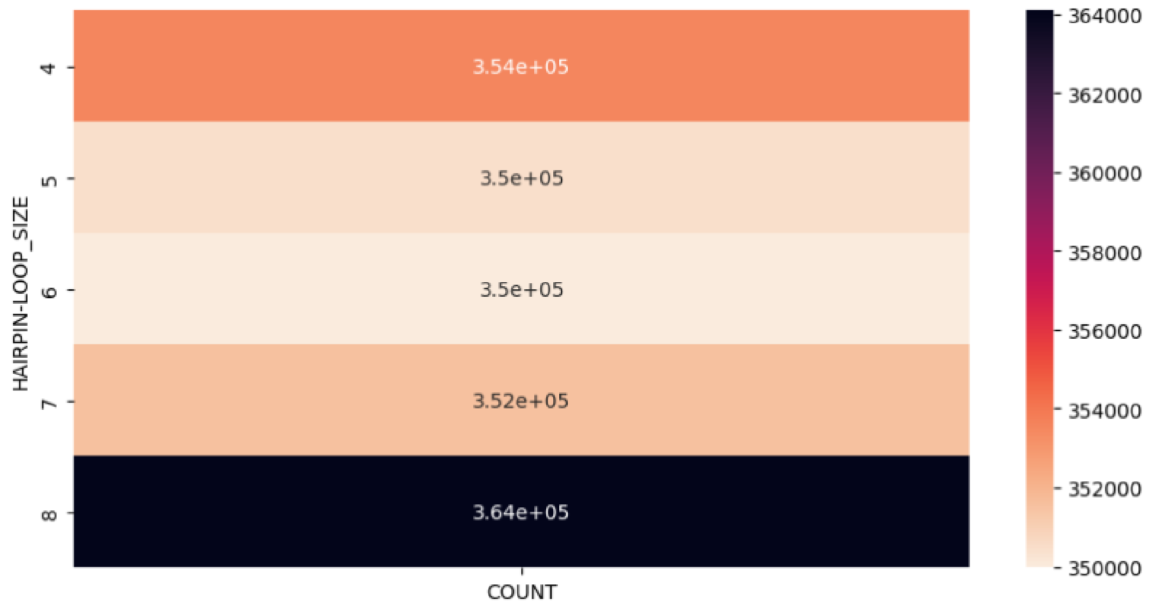


Fig. A.32: Frequencies of Exonic Region Hairpin Loop Sizes