Faculty of Computer Informatics & Engineer
Department of Software & Information System Engineer

كلية هندسة الحاسوب والمعلوماتية
قسم هندسة البرمجيات ونظم المعلومات




# تصنيف الامراض الجينية باستخدام بيانات حيوية

*Gene-Disease Classification Using Genomic Data*

**إعداد الطلاب**

**أحمد أسامة ريشي (4180172)**

**آية محمد وائل الأسود (4190209)**

**بإشراف**

**د. م. مهيب النقري**

**م. أنس عبد العزيز**

*2023/2022*

SUPERVISOR CERTIFICATION

I certify that the preparation of this project entitled
……....., prepare by
……
……, was under my supervision at Faculty of Computer, Informatics Engineering in partial fulfillment of the Requirements for the Degree of Bachelors of Software and Information System Engineering

Name: …….

Signature: ……

Date: ……..

# Abstract

This research focuses on utilizing various Machine Learning and Deep Learning techniques to classify disease-causing genes based on the specific type of each disease. Early detection of diseases plays a crucial role in facilitating prompt treatment. A large genetic dataset was employed, and the results obtained from different techniques were compared to identify the most suitable approach for each disease.

The primary objective of this research is to accurately classify genes while employing appropriate techniques to analyze and curate genomic data. This facilitates the identification of genes responsible for diseases, enabling early detection and preventive measures. Additionally, the research aims to anticipate future gene mutations that may give rise to diseases not previously evident in the genetic sequence.

Two models are developed as part of this research. The first model focuses on classifying each gene according to its associated disease, while the second model aims to predict the specific variant of the gene. It is important to consider that genes can have multiple variants, and predicting the specific variant adds further depth to the analysis and understanding of genetic factors contributing to diseases.

In conclusion, our models have demonstrated high accuracy, reaching up to 97% in classifying diseases such as lung cancer. Additionally, we have identified the optimal classifier, which is the Random Forest Classifier, for our NLP model, achieving the lowest loss and highest accuracy.

# الملخص

يركز هذا البحث على استخدام تقنيات التعلم الآلي والتعلم العميق المختلفة لتصنيف الجينات المسببة للأمراض بناءً على النوع المحدد لكل مرض. يلعب الاكتشاف المبكر للأمراض دورًا مهمًا في تسهيل العلاج السريع. تم استخدام مجموعة كبيرة من البيانات الجينية ، وتمت مقارنة النتائج التي تم الحصول عليها من تقنيات مختلفة لتحديد النهج الأنسب لكل مرض.

الهدف الأساسي من هذا البحث هو تصنيف الجينات بدقة مع استخدام التقنيات المناسبة لتحليل البيانات الجينومية وتنظيمها. هذا يسهل تحديد الجينات المسؤولة عن الأمراض ، مما يتيح الكشف المبكر والتدابير الوقائية. بالإضافة إلى ذلك ، يهدف البحث إلى توقع الطفرات الجينية المستقبلية التي قد تؤدي إلى ظهور أمراض لم تكن واضحة من قبل في التسلسل الجيني.

تم تطوير نموذجين كجزء من هذا البحث. يركز النموذج الأول على تصنيف كل جين حسب المرض المرتبط به ، بينما يهدف النموذج الثاني إلى التنبؤ بالتنوع المحدد للجين. من المهم اعتبار أن الجينات يمكن أن يكون لها متغيرات متعددة ، والتنبؤ بالمتغير المحدد يضيف مزيدًا من العمق لتحليل وفهم العوامل الوراثية التي تساهم في الأمراض.

في الختام ، أظهرت نماذجنا دقة عالية وصلت إلى 97٪ في تصنيف الأمراض مثل سرطان الرئة. بالإضافة إلى ذلك ، حددنا المصنف الأمثل ، وهو Random Forest Classifier ، لنموذج البرمجة اللغوية العصبية الخاص بنا ، مما يحقق أقل خسارة وأعلى دقة.

# DEDICATION/إهداء

# ACKNOWLEDGEMENTS

# Table of Contents


Introduction ..... 12

1.1 Problem Statement ..... 12

1.2 Goal of the Project ..... 12

1.3 Proposed Solution ..... 13

1.4 Challenges ..... 14

1.5 Project importance ..... 16

1.6 Proposed Solution Structure ..... 17

Theoretical Study ..... 21

2.1 Introduction ..... 21

2.2 Main concepts ..... 22

1. Bioinformatics ..... 22
2. Data analysis in Bioinformatics ..... 22
3. DNA and RNA ..... 23
4. Gene Expression ..... 24
5. Data Mining in Bioinformatics ..... 25
   - Clustering: ..... 25
   - Classification: ..... 25
   - Feature Selection: ..... 26
   - Pattern Discovery: ..... 26
   - Text Mining: ..... 26
6. DNA Sequence encoding techniques ..... 27
   - k-MERS ..... 27
   - One-Hot Encoding: ..... 28
   - Sequence Embedding: ..... 28
   - FASTA Format ..... 29
7. Machine Learning ..... 30
8. Machine Learning Classifiers ..... 31
   - Random Forest ..... 31
   - Stochastic Gradient Descent (SGD) ..... 31

- Calibrated Classifier.....32
- KNN.....32

9. Deep learning.....32
10. Neural Network.....33

- Epoch.....34
- Batch Size.....34
- Model Checkpoint.....34
- Early Stopping.....34
- Optimizer.....35
- Loss Function.....35
- Activation Function.....35

11. CNN.....36

2.3 References Studies.....37

1. BLAST.....37
2. Fast Alignment and Search Tool.....38

Design Solution.....40

3.1 Define Project Requirements.....40

3.2 Designing the Classification model.....41

1. NCBI Databases Overview:.....41
2. Data Gathering.....42
3. Explore the Data:.....43
4. Methodology.....45
5. Architecture.....45

3.3 Designing the Text mining model.....47

1. Data Gathering.....47
2. Methodology.....47
3. Model.....48

Implementation.....49

4.1 Genetic Classification in Deep Learning.....49

1. Dataset Characteristics.....49
2. Data Preprocessing.....50

3. Model Training .....52
4. Evaluation Results.....54
1. Alzhiemer.....54
2. Lung Cancer.....56
3. Head Cancer.....60
4. Breast Cancer.....61
4.2 Detect Gene Variation.....66
1. KNN Classifier.....70
2. SGD Classifier .....71
3. Random Forest Classifier.....72
Conclusion .....74
References.....75

# Table of Figures

Figure 1: System Architecture .....19
Figure 2: DNA and RNA Structure .....23
Figure 3: DNA Transcription Process .....24
Figure 4: DNA Sequence Encoding .....27
Figure 5: k-MER Encoding Techniques .....27
Figure 6: Neural Network Structure .....33
Figure 7: Convolutional Neural Networks Architecture .....36
Figure 8: Distribution of Lung Cancer Data .....43
Figure 9: Distribution of Alzheimer Data .....44
Figure 10: Gene Expression Data Sample .....44
Figure 11: Gene Sequence Data Sample .....44
Figure 12: CNN Architecture .....46
Figure 13: Fasta Reading Code .....50
Figure 14: Sequence Encoding Code .....51
Figure 15: Initial CNN Code .....52
Figure 16: Model Compiling Code .....52
Figure 17: Model Configuration Code .....52
Figure 18: Model Fitting Code .....53
Figure 19: Train Evaluate Code .....53
Figure 20: Model Performance Evaluation Code .....53
Figure 21: Alzhiemer Model Accuracy Diagram .....54
Figure 22: Alzhiemer Model Performance .....55
Figure 23: Lung Cancer Model Accuracy (Sequence) .....56
Figure 24: Lung Cancer Model Performance (Sequence) .....57
Figure 25: Lung Cancer Model Accuracy (Expression) .....58
Figure 26: Lung Cancer Model Performance (Expression) .....59
Figure 27: Head Cancer Model Accuracy .....60
Figure 28: Head Cancer Performance .....61
Figure 29: Breast Cancer Model Accuracy (Sequence) .....62
Figure 30: Breast Cancer Model Accuracy (Sequence) .....63
Figure 31: Breast Cancer Model Accuracy (Expression) .....64
Figure 32: Breast Cancer Model Performance (Expression) .....65
Figure 33: Import Data Code .....66
Figure 34: Text Data Preprocess Code .....67
Figure 35: Class Distribution Code .....68
Figure 36: Distribution of Gene Variant Class .....68
Figure 37: One-hot encoding code .....69

Figure 38: KNN Performance.....70
Figure 39: SGD Performance .....71
Figure 40: Random Forest Performance.....72

# Introduction

Within this chapter, we will present an overview of the problem statement, the objectives of the project, and outline the proposed solution.

## 1.1 Problem Statement

The identification and classification of genes associated with various diseases is a complex and challenging task. Traditional methods for analyzing genetic data often require significant manual effort and lack the ability to effectively process the vast amount of available genomic information. This time-consuming manual effort, limits the speed, accuracy, and scalability of disease classification, hindering advancements in personalized medicine and targeted treatments. To overcome these limitations, there is a need for an intelligent computational approach that can automate the process of classifying genes into disease categories, allowing for more efficient and accurate analysis.

## 1.2 Goal of the Project

Our project focuses on gene-disease classification, aiming to enable researchers and healthcare professionals to efficiently and accurately identify genes and their variations associated with specific diseases. The primary objective is to develop a robust and scalable model capable of analyzing large-scale genomic data, extracting pertinent features, and predicting disease categories based on genetic information. By achieving this, we aim to enhance our comprehension of the genetic mechanisms underlying diseases and facilitate the development of personalized treatment strategies.

To accomplish these goals, we will explore and evaluate various machine learning and deep learning techniques for gene classification. The project involves applying and comparing the performance of different techniques to identify the most effective approach for accurately predicting gene-

disease associations and their specific variations. Subsequently, we will select the most appropriate technique and evaluate its model.

Ultimately, our project aims to advance our understanding of genetic mechanisms in diseases and provide an efficient means of identifying disease-associated genes and their specific variations.

## 1.3 Proposed Solution

Our proposed solution focuses on automating the gene classification process by harnessing the computational power of data science techniques, thereby replacing manual efforts. By utilizing various aspects of data science, such as feature extraction and selection, our objective is to improve the accuracy and efficiency of gene classification models.

To achieve this, we will build and evaluate a CNN model that classifies and predicts gene associations with different diseases. Additionally, another model will support our project by identifying variations of each gene using Text Mining and Recognition techniques with NLP. Leveraging the computational capabilities of these algorithms, we will train and optimize the models using labeled genetic data. Through an iterative process, we will fine-tune the models by adjusting their parameters and hyperparameters to maximize their performance in gene classification. Techniques like cross-validation, grid search, and optimization algorithms will be employed to identify the optimal configuration for our models.

Our goal is to develop a robust gene classification system capable of efficiently analyzing large-scale genomic data. This system will provide valuable insights into gene-disease associations, contributing to advancements in personalized medicine, targeted treatments, and enhancing our understanding of the genetic mechanisms underlying diseases.

## 1.4 Challenges

1. **Lack of medical knowledge:** Without a proper medical background, we had a hard time understanding the biological concepts and terminologies related to genetics can. It was difficult to interpret the significance of various genes and their functions in relation to specific diseases or traits.

2. **Data Understanding and Preprocessing:** It was challenging to understand the structure of the data, preprocess it, and deal with missing values or outliers due to the high complexity of biological data and lack of knowledge in genetics.

3. **Feature Selection and Dimensionality:** Genomic data often contains a high number of features or variables, which can lead to the curse of dimensionality. Selecting relevant features and reducing dimensionality to improve classification accuracy requires a good understanding of genetics and data science techniques.

4. **Data Accessibility:** Acquiring suitable and comprehensive genomic data can be a challenging task. Genomic data is often subject to restrictions due to privacy concerns, ethical considerations, or data protection regulations. Access to large and diverse datasets may require collaborations with research institutions, access to public databases, or adherence to specific data-sharing agreements.

5. **Data Quality and Standardization:** Genomic data can vary in quality, consistency, and format. Different sources may have varying data collection methods, preprocessing techniques, and annotation standards, leading to potential inconsistencies and biases. Ensuring data quality and standardization becomes crucial for accurate and reliable analysis.

6. **Class Imbalance:** In gene classification, class imbalance can occur when certain disease-gene associations are more prevalent or better studied than others. Unequal representation of classes in the dataset can bias the model's learning process, leading to lower performance for underrepresented classes. Techniques such as oversampling, under-sampling, or class weighting can be employed to mitigate this issue.

7. **Noisy and Incomplete Data:** Genomic data can be noisy, containing errors or missing values. Missing values in genomic data can pose challenges during training and inference. Strategies like imputation or exclusion of missing data need to be carefully considered to avoid introducing bias or misleading the models.

8. **Computational Resources:** Deep learning models, especially when working with large-scale genomic data, can be computationally intensive and require substantial computational resources, including high-performance GPUs or cloud computing platforms. Availability and access to these resources may pose limitations for individuals or research groups with limited computational infrastructure.

Addressing these challenges requires careful planning, collaboration with domain experts, and utilization of appropriate techniques and methodologies. It is important to acknowledge and manage these challenges to ensure the validity, reliability, and generalizability of the project's outcomes.

## 1.5 Project importance

By integrating machine learning, deep learning, and text mining techniques, the project addresses critical challenges in genomics and personalized medicine. It has the potential to contribute to scientific knowledge, improve patient care, and drive advancements in the understanding and treatment of diseases. This project would provide valuable contributions and support to researchers and scientists in several ways:

- Our project will provide Efficiency Automation by developing computational models and leveraging machine learning, deep learning, and text mining techniques, this would automate and streamline the gene classification and gene variation recognition processes. This would significantly reduce manual effort and accelerate the analysis of large-scale genomic and textual data. Researchers and scientists would benefit from the increased efficiency and speed, allowing them to process and analyze a larger volume of data in a shorter timeframe.

- Improved Accuracy and Precision: The utilization of advanced techniques like machine learning and deep learning in gene classification and gene variation recognition would enhance the accuracy and precision of the results. Researchers and scientists would have access to more reliable and robust information, enabling them to make informed decisions and draw meaningful conclusions from their analyses. This improved accuracy would contribute to the overall quality of research findings and facilitate more accurate interpretations of genetic data.

- Discovery of Novel Gene-Disease Associations: By applying different machine learning and deep learning techniques in gene classification, this project would enable the discovery of novel gene-disease associations. This could uncover previously unknown relationships between genes and diseases, leading to new insights and discoveries in the field of genomics. Researchers and scientists would benefit from these discoveries, as they could open up new

avenues for further investigation, treatment development, and targeted therapies.

In summary, this project would empower researchers and scientists by automating and enhancing the gene classification and gene variation recognition processes. The increased efficiency, improved accuracy, and potential discoveries would support their research endeavors, enable the exploration of new gene-disease associations, and facilitate collaboration and knowledge sharing within the scientific community.

## 1.6 Proposed Solution Structure

Our proposed solution for this project which consists of two parts, involves the following key components:

Part 1: Gene Classification using Machine Learning and Deep Learning Techniques

In this project, we aim to explore and compare various machine learning and deep learning techniques for gene classification in order to determine the most effective approach. By leveraging different algorithms such as k-Nearest Neighbors (kNN), Support Vector Machines (SVM), Convolutional Neural Networks (CNN), and Long Short-Term Memory Networks (LSTM), we will evaluate their performance in accurately categorizing genes into disease categories.

- **Data Collection:** Gather relevant genomic data, including gene sequences, clinical data, disease information, and any other associated datasets.

- **Data Preprocessing:** Clean and preprocess the collected genomic data, ensuring its quality and compatibility with the chosen techniques. This step may involve data cleaning, handling missing values, normalizing features, and partitioning it into training, validation, and test sets

- **Feature Selection/Extraction:** Choose appropriate features from the preprocessed genomic data that effectively represent the genes and their characteristics.

- **Model Selection and Training:** Select and implement different machine learning and deep learning models, kNN, SVM, CNN, and LSTM, tailored to gene classification. These techniques have distinct approaches to learning patterns from the genetic data and making predictions about disease associations.

- **Model Evaluation:** This involves Model Training and Optimization. The models will be trained using the training data. Techniques such as cross-validation, grid search, and optimization algorithms will be employed to find the best configuration for each technique. Evaluate the performance of each model using appropriate evaluation metrics, such as accuracy, precision, recall, F1-score. Compare the results to determine the best-performing technique in terms of gene classification accuracy. This will allow us to assess their effectiveness in correctly classifying genes into disease categories.

- **Results Presentation:** Present the results of each technique, highlighting the accuracy achieved in gene classification. This can be done through tables, charts, or visualizations to provide a clear comparison and facilitate understanding.

- **Comparative Analysis:** Based on the evaluation results, we will compare and analyze the performance of kNN, SVM, CNN, and LSTM techniques. We will consider factors such as accuracy, computational efficiency, and interpretability to identify the technique that demonstrates the most promising results.

- **Further Optimization and Ensemble:** it's also called Iterative Refinement we will perform additional optimization on the chosen technique to fine-tune its parameters and enhance its performance.

Iterate and refine the implemented methods based on feedback, additional data, or emerging techniques in the field, To Continuously improve the performance of the gene classification and gene variation recognition approaches.

- **Reporting and Conclusion:** The findings of our analysis will be documented in a comprehensive report. We will summarize the performance results, compare the strengths and weaknesses of each technique, and provide insights into the most effective approach for gene classification. The report will also highlight potential avenues for future research and improvement.

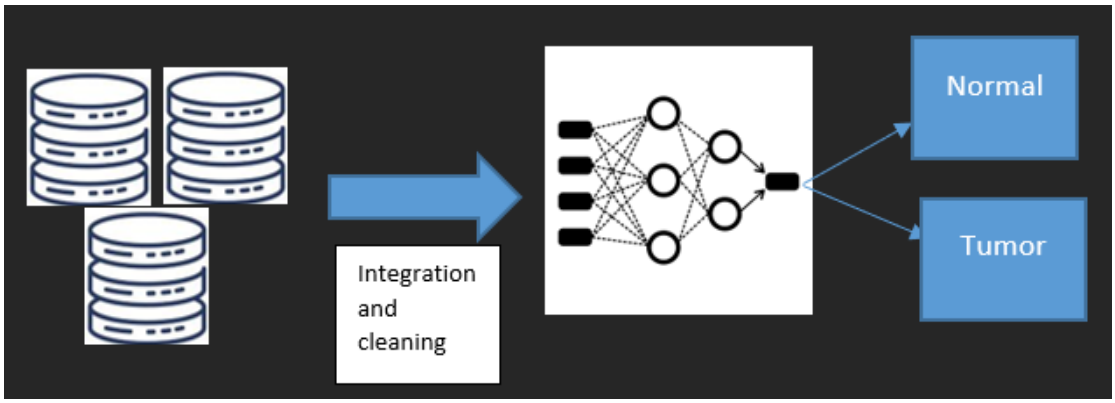



***Figure 1: System Architecture***

By implementing and evaluating kNN, SVM, CNN, and LSTM techniques, this proposed solution aims to identify the most suitable approach for gene classification for each gene based on their performance on the given dataset. The project's outcomes will contribute to advancing the field of gene-disease association prediction and enable more accurate diagnostics and personalized treatment strategies in genomics.

Part 2: Gene Variation Recognition through Text Mining

- **Text Corpus Construction:** Compile and preprocess textual data from scientific literature, clinical reports, and other relevant sources containing information on gene variations.

- **Relation Extraction:** Extract relationships between gene variations and associated diseases or phenotypes using techniques such as pattern matching, rule-based methods, or advanced natural language processing (NLP) approaches. Determine the co-occurrence or semantic associations between gene variations and relevant entities.

- **Evaluation:** Evaluate the performance of the text mining techniques by comparing the recognized gene variations with curated databases or existing knowledge resources. Assess metrics such as precision, recall, and F1-score to measure the accuracy of gene variation recognition.

- **Final Presentation:** Summarize the findings, methodologies, and conclusions of the project. Present the results of gene variation recognition alongside the gene classification results, demonstrating the effectiveness of the text mining approach in recognizing gene variations.

# Theoretical Study

## 2.1 Introduction

In this section, our objective is to establish a thorough comprehension of the theoretical underpinnings that form the basis of our proposed solution. We initiate the study by delving into the fundamental principles governing gene-disease classification and their profound significance in healthcare and research domains. Our exploration encompasses the intricate relationship between genetic variations and the development of diverse diseases, with a specific emphasis on the crucial role played by precise gene classification in enabling early detection and facilitating targeted treatment strategies.

Within the context of gene classification, we thoroughly examine the theoretical frameworks and methodologies rooted in data science, including machine learning and deep learning. This comprehensive analysis entails an extensive review of feature extraction and selection techniques, model architectures, and evaluation metrics that are commonly employed in gene-disease classification studies.

By embarking on this theoretical journey, we aim to establish a solid foundation for our proposed solution. We aspire to gain valuable insights from established theories, concepts, and best practices in gene classification. By harnessing the power of data science and building upon existing knowledge, we endeavor to develop a robust gene classification models. Theses models will not only contribute to advancements in personalized medicine and targeted treatments but also enhance our understanding of the intricate genetic mechanisms that underlie various diseases.

## 2.2 Main concepts

1. **Bioinformatics**

   Bioinformatics is an interdisciplinary field that combines Biology, Computer Science, and Statistics to analyze and interpret biological data. It involves the development and application of computational methods, algorithms, and tools to store, retrieve, organize, and analyze biological information. Bioinformatics plays a crucial role in understanding complex biological systems, unraveling the underlying genetic mechanisms, and facilitating advancements in areas such as genomics, proteomics, and drug discovery.

   In bioinformatics, large-scale biological data is generated from various sources, including DNA sequencing, gene expression analysis, protein structure determination, and biomedical literature. Bioinformatics tools and techniques enable researchers to process, integrate, and analyze these data to extract meaningful insights and knowledge.

2. **Data analysis in Bioinformatics**

   Data analysis in bioinformatics involves the use of computational algorithms, statistical modeling, machine learning techniques, and visualization tools to process and interpret the data. It involves various techniques and algorithms to extract meaningful information, identify patterns, and gain insights into biological processes. The goal is to uncover meaningful insights, discover new knowledge, and support biological discoveries and advancements in fields such as genomics, proteomics, drug discovery, and personalized medicine. All the insights are derived from large-scale biological datasets.

## 3. DNA and RNA

DNA sequence refers to the specific arrangement of nucleotides along a DNA molecule. DNA (deoxyribonucleic acid) is a molecule found in the cells of living organisms, carrying genetic information that determines an organism's traits and functions. The sequence of nucleotides, which include adenine (A), thymine (T), cytosine (C), and guanine (G), forms the genetic code that serves as instructions for the synthesis of proteins and regulates various biological processes. The unique sequence of nucleotides in DNA is essential for encoding the genetic information that governs an organism's growth, development, and functioning. DNA sequencing techniques enable scientists to read and decipher the order of nucleotides within a DNA molecule, facilitating the study of genetic variations, gene expression, and understanding the genetic basis of diseases and inherited traits.

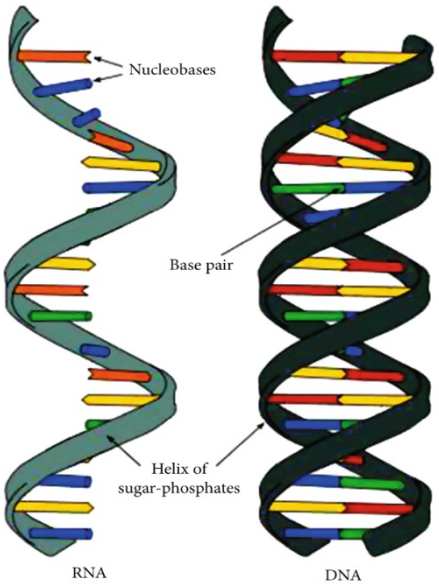



***Figure 2: DNA and RNA Structure***

## 4. Gene Expression

Gene expression refers to the process by which genetic information encoded in a gene is utilized to synthesize functional gene products, such as proteins or non-coding RNA molecules. It involves the conversion of genetic instructions stored in the DNA sequence into a functional product that performs specific biological functions within a cell or organism.

Gene expression is a tightly regulated process that allows cells to respond to various internal and external signals, ensuring that the right genes are activated or repressed at the appropriate times and in the relevant cell types. It plays a crucial role in determining cell identity, development, and response to environmental stimuli.

Studying gene expression is essential for understanding biological processes, identifying disease mechanisms, and developing therapies. Techniques such as gene expression profiling, microarray analysis, and RNA sequencing are used to measure and analyze gene expression levels across the genome, enabling researchers to gain insights into the complex regulatory networks that govern cellular functions.

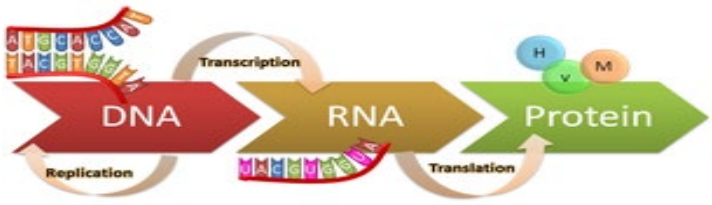



***Figure 3: DNA Transcription Process***

## 5. Data Mining in Bioinformatics

It involves the application of various computational techniques, statistical algorithms, and machine learning methods to extract meaningful information from data. The objective of data mining is to uncover hidden patterns, identify trends, and gain insights that can be used for decision-making and knowledge discovery. Data mining encompasses tasks such as classification, clustering, association rule mining, and anomaly detection. It is commonly used in fields such as business intelligence, marketing, and scientific research, including bioinformatics.

It involves the application of various computational techniques, statistical algorithms, and machine learning methods to extract meaningful information from data.

In bioinformatics Data mining plays a vital role in analyzing biological data and extracting meaningful knowledge. It involves the application of computational techniques and algorithms to discover patterns, associations, and relationships within biological datasets. It plays a crucial role in extracting meaningful insights from large-scale biological data and has numerous applications in genomics, proteomics and other fields of biology.

Here are some major aspects of data mining in bioinformatics:

- **Clustering:**
  techniques are used to group similar biological entities based on their characteristics. For example, gene expression data can be clustered to identify co-expressed genes or to classify samples into different subtypes. Clustering helps in understanding the underlying structure and organization of biological data.

- **Classification:**
  Methods are employed to assign biological entities to predefined categories or classes. For instance, machine learning algorithms can be trained on labeled datasets to classify DNA sequences, protein structures, or disease types. Classification models can assist in predicting unknown classes and making informed decisions based on the available data.

- **Feature Selection:**
  Techniques are utilized to identify the most relevant and informative features (variables) within a dataset. In bioinformatics, this can involve selecting genes or genetic variants that are most influential in determining a specific phenotype or disease outcome. Feature selection helps reduce dimensionality, improve model performance, and identify key factors driving biological processes.

- **Pattern Discovery:**
  Data mining methods enable the discovery of meaningful patterns and motifs within biological sequences. This includes identifying conserved regions, transcription factor binding sites, or protein domains. Pattern discovery contributes to understanding the functional elements and regulatory mechanisms within DNA, RNA, and protein sequences.

- **Text Mining:**
  Techniques are used to extract information and knowledge from scientific literature, such as research articles or biological databases. It involves natural language processing, information retrieval, and text classification to extract relevant information, identify relationships, and create knowledge bases for further analysis.

Data mining techniques in bioinformatics enable researchers to uncover valuable insights, discover novel associations, and generate hypotheses for further investigation. They assist in understanding complex biological systems, identifying biomarkers, predicting drug responses, and advancing personalized medicine.

## 6. DNA Sequence encoding techniques

The DNA sequence is very long, having a length of thousands nucleotides, and it is challenging to understand and interpret, and as the genomic sequence in the DNA dataset is categorical There are many techniques available to convert the categorical data to numerical. The encoding technique is the process of converting the categorical data of nucleotide into numerical form.

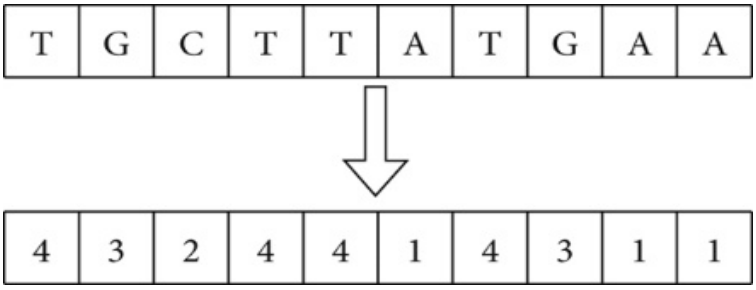



***Figure 4: DNA Sequence Encoding***

- **k-MERS**
  K-mers are subsequences of length k that are extracted from longer sequences, such as DNA or protein sequences. In the context of genomics, k-mers are commonly used as a fundamental unit of analysis. For example, a 6-mer (k = 6) would consist of all possible combinations of six adjacent nucleotides in a DNA sequence. By considering different k-mer lengths, we can capture varying levels of sequence information and structural patterns as shown in the following figure.

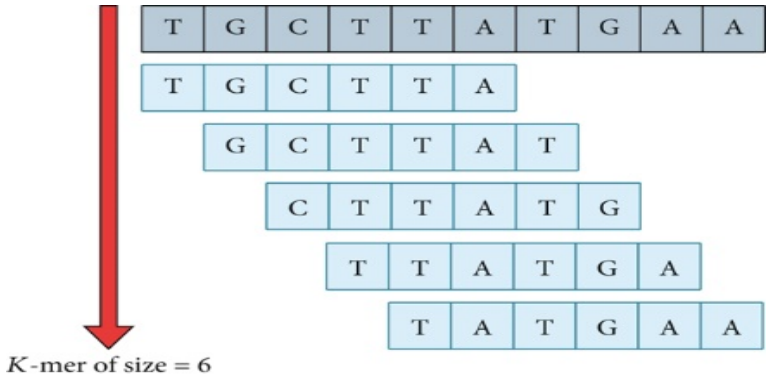



***Figure 5: k-MER Encoding Techniques***

k-MERs are advantageous in genomics research due to their simplicity and computational efficiency. They provide a compact representation of sequences and enable the analysis of large-scale genomic datasets. k-MERs can be used for tasks such as sequence alignment, motif discovery, and sequence classification.

- **One-Hot Encoding:**
  One-hot encoding is a technique used to represent categorical data, such as DNA or protein sequences, as binary vectors. Each unique category is assigned a binary value, where only one bit is "on" (1) and the rest are "off" (0). For example, in DNA sequences, the four nucleotides (A, C, G, T) can be one-hot encoded as [1, 0, 0, 0], [0, 1, 0, 0], [0, 0, 1, 0], and [0, 0, 0, 1], respectively.

  One-hot encoding allows us to transform categorical data into a numerical representation that can be effectively used as input for machine learning algorithms. It ensures that each category is represented independently and avoids any inherent ordinarily or numerical bias.

- **Sequence Embedding:**
  Sequence embedding refers to the process of representing sequential data, such as DNA or protein sequences, as continuous numerical vectors in a lower-dimensional space. It aims to capture the underlying patterns and relationships within the sequences, enabling machine learning algorithms to effectively process and learn from the data. Sequence embedding techniques, such as word embedding's in natural language processing, learn dense vector representations that encode the semantic or structural information of the sequences.

  By using sequence embedding, we can capture the nuanced relationships between genes and their variations. It allows us to effectively model the hierarchical and contextual information within sequences, enabling more sophisticated analysis and classification. In our project, we can employ sequence embedding techniques, such as word2vec or embedding layers

in neural networks, to learn informative representations of gene variations and improve the performance of our classification models.

The importance of these concepts lies in their ability to extract meaningful information and patterns from genomic data. k-MERs, one-hot encoding, and sequence embedding provide us with powerful tools to represent and analyze genes and their variations. By leveraging these techniques, we can effectively capture sequence patterns, explore gene-disease associations, and develop accurate classification models for personalized medicine and disease prediction.

- **FASTA Format**
  FASTA is a commonly used file format in bioinformatics for storing nucleotide or protein sequence data. It is named after the FASTA software package, which introduced this format. The FASTA format is simple and human-readable, making it widely adopted for representing sequence information.

  A FASTA file typically consists of one or more entries, each representing a sequence. Each entry is composed of two parts:

  **Header:** The header line begins with a ">" symbol, followed by a unique identifier or description of the sequence. It can also include additional information such as the organism, gene name, or other annotations.

  **Sequence:** The sequence part is represented as a series of characters, typically using the standard symbols for nucleotides (A, C, G, T) or amino acids (A, R, N, D, etc.). The sequence can be presented on a single line or split into multiple lines for better readability, but there are usually no line length restrictions.

Here is an example of a FASTA entry:

>sequence1

ATCGATCGATCGATCGATCGATCGATCGATCG

In this example, "sequence1" is the header, and "ATCGATCGATCGATCGATCGATCGATCGATCG" is the nucleotide sequence.

FASTA files are widely used for storing and exchanging sequence data in bioinformatics. They can be easily parsed and processed by various software tools and algorithms for tasks such as sequence alignment, sequence database searching, and sequence analysis.

## 7. Machine Learning

Machine learning, in general, is a branch of artificial intelligence that focuses on the development of algorithms and models that enable computers to learn from data and make predictions or take actions without being explicitly programmed. It involves the study of statistical and computational technique. The primary goal of machine learning is to build models that can learn from data, identify patterns, and make accurate predictions or decisions on new, unseen data. Machine learning algorithms are designed to automatically improve their performance with experience or training data. It involves tasks such as classification, regression, clustering, and dimensionality reduction. Machine learning techniques can be applied to a wide range of domains, including bioinformatics.

In the field of bioinformatics, machine learning plays a significant role in analyzing and interpreting biological data. It allows researchers to leverage large-scale biological datasets, such as genomic sequences, gene expression profiles, and protein structures, to gain insights into biological systems, predict outcomes, discover biomarkers, and facilitate personalized

medicine. It aids in data-driven decision-making, hypothesis generation, and accelerating discoveries in the field of biology.

## 8. Machine Learning Classifiers

In the field of machine learning, classifiers play a crucial role in solving various prediction and classification tasks. They are algorithms that learn from labeled data to make predictions or assign class labels to unseen instances. In this project, we will explore and discuss several popular machine learning classifiers, namely SVM, Random Forest, SGD, and Calibrated classifier. These classifiers have proven to be effective in a wide range of applications and offer different strengths and capabilities.

- **Random Forest**
  Random Forest is an ensemble learning method that combines multiple decision trees to make predictions. It is known for its ability to handle high-dimensional data and deal with noisy and missing data. Random Forest builds a collection of decision trees and aggregates their predictions to produce the final output. It provides robustness against overfitting and offers feature importance analysis, making it useful for understanding the underlying patterns in the data.

- **Stochastic Gradient Descent (SGD)**
  SGD is an optimization algorithm commonly used in training machine learning models, particularly in large-scale and online learning scenarios. It updates the model parameters iteratively by considering only a small subset of training examples, which makes it computationally efficient and suitable for handling large datasets. SGD is widely used in linear classifiers and neural networks, offering flexibility in model customization and parameter tuning.

- **Calibrated Classifier**
  It is a technique used to obtain calibrated probability estimates from a trained classifier. It ensures that the predicted probabilities reflect the true likelihood of an instance belonging to a particular class. Calibrated classifiers are particularly useful when probabilistic predictions are required, such as in risk assessment or decision-making scenarios. They can enhance the reliability and interpretability of the classifier's output.

- **KNN**
  k-Nearest Neighbors (kNN) is a simple yet powerful machine learning algorithm used for both classification and regression tasks. It operates based on the principle that similar data points tend to belong to the same class. In kNN, the "k" refers to the number of nearest neighbors considered for classification. Given a new data point, kNN identifies its k nearest neighbors based on a chosen distance metric (e.g., Euclidean distance) and assigns the class label based on the majority vote of the neighbors. kNN is a non-parametric algorithm, meaning it does not make any assumptions about the underlying data distribution. One advantage of kNN is its simplicity and ease of implementation. However, it can be computationally expensive for large datasets and may suffer from the curse of dimensionality.

## 9. Deep learning

Deep learning is a subfield of machine learning that focuses on training artificial neural networks with multiple layers to learn complex representations of data. It can be considered as an advanced and specialized form of machine learning, particularly suitable for handling large-scale, unstructured, and high-dimensional data.

## 10.Neural Network

A neural network consists of interconnected nodes, called artificial neurons or units, organized into layers. The most basic type of neural network is a feedforward neural network, where information flows in one direction, from the input layer through one or more hidden layers to the output layer. (as shown in figure) Each neuron in a layer receives input from neurons in the previous layer and produces an output, which serves as input to neurons in the subsequent layer. The outputs of the final layer represent the network's predictions or classifications. The strength of a neural network lies in its ability to learn complex patterns and relationships in data through a process called training. During training, the network adjusts the weights and biases associated with each neuron to minimize the difference between its predicted output and the desired output, typically using an optimization algorithm like gradient descent. This learning process involves forward propagation of inputs through the network, followed by backward propagation of errors to update the weights.

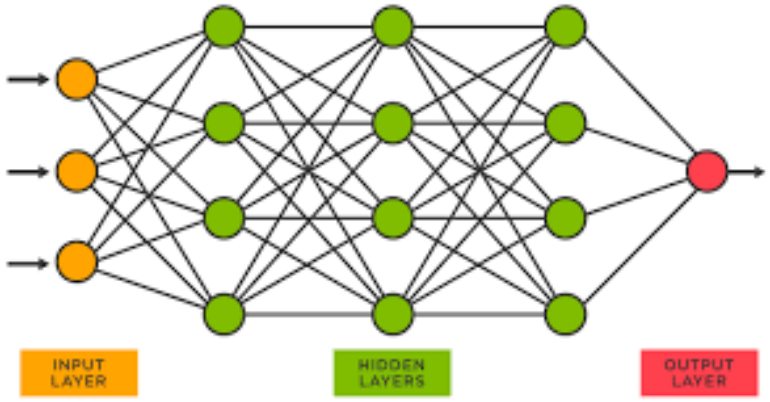



***Figure 6: Neural Network Structure***

**Important terminologies in neural networks:**

- **Epoch**
  An epoch refers to a single pass of the entire training dataset through the neural network. During an epoch, the model makes predictions on the input data, calculates the loss (error), and updates its parameters (weights and biases) based on the optimization algorithm. Training typically involves running multiple epochs to improve the model's performance by iteratively adjusting the parameters.

- **Batch Size**
  Batch size refers to the number of training examples that are processed together in one forward and backward pass through the neural network. Instead of feeding all the training data at once, it is divided into smaller batches. The batch size determines how many samples are used to calculate the gradient and update the parameters. Larger batch sizes can lead to faster training as computations can be parallelized, but it requires more memory. Smaller batch sizes provide more frequent parameter updates and may result in better generalization.

- **Model Checkpoint**
  Model checkpointing is a technique where the model's weights are saved at certain checkpoints during the training process. It allows you to save the best-performing model or intermediate models at specific epochs. Model checkpoints are helpful for resuming training, evaluating the model's performance, or deploying the model later.

- **Early Stopping**
  Early stopping is a technique used to prevent overfitting and determine the optimal stopping point during training. It involves monitoring a validation metric, such as validation loss or accuracy, and stopping the training if the metric stops improving or starts deteriorating. Early stopping helps to find a

balance between model performance and avoiding excessive training that leads to overfitting.

- **Optimizer**
  An optimizer is an algorithm used to update the model's parameters during training. It determines how the model adjusts its weights and biases based on the calculated gradients. Common optimizers include stochastic gradient descent (SGD), Adam, RMSprop, and Adagrad. Each optimizer has its own update rules and hyperparameters that influence the learning process and convergence speed.

- **Loss Function**
  A loss function quantifies the difference between the predicted outputs of the neural network and the actual labels or targets. It measures the model's performance during training and guides the optimization process. Common loss functions include mean squared error (MSE), binary cross-entropy, categorical cross-entropy, and mean absolute error (MAE), depending on the type of problem being solved.

- **Activation Function**
  An activation function introduces non-linearity to the output of a neuron in a neural network. It determines whether the neuron should be activated (fire) or not. Activation functions help the model learn complex patterns and relationships in the data. Examples of activation functions include sigmoid, tanh, ReLU (Rectified Linear Unit), and softmax.

These concepts play crucial roles in training and optimizing neural networks, allowing them to learn from data and make accurate predictions on new inputs.

## 11.CNN

Convolutional Neural Networks (CNN) are a class of deep learning models specifically designed for processing structured grid-like data, such as images. CNNs are widely used in computer vision tasks and have achieved remarkable success in image classification, object detection, and image generation. CNNs consist of multiple layers, including convolutional layers, pooling layers, and fully connected layers. (as shown in figure) The convolutional layers apply filters to extract local features from the input, while the pooling layers reduce the spatial dimensions. The fully connected layers then perform high-level feature aggregation and produce the final classification output. CNNs leverage parameter sharing and hierarchical representations, allowing them to capture complex patterns and spatial dependencies in the data. CNNs require large amounts of labeled training data and can be computationally demanding, but they have demonstrated state-of-the-art performance in image-related tasks.

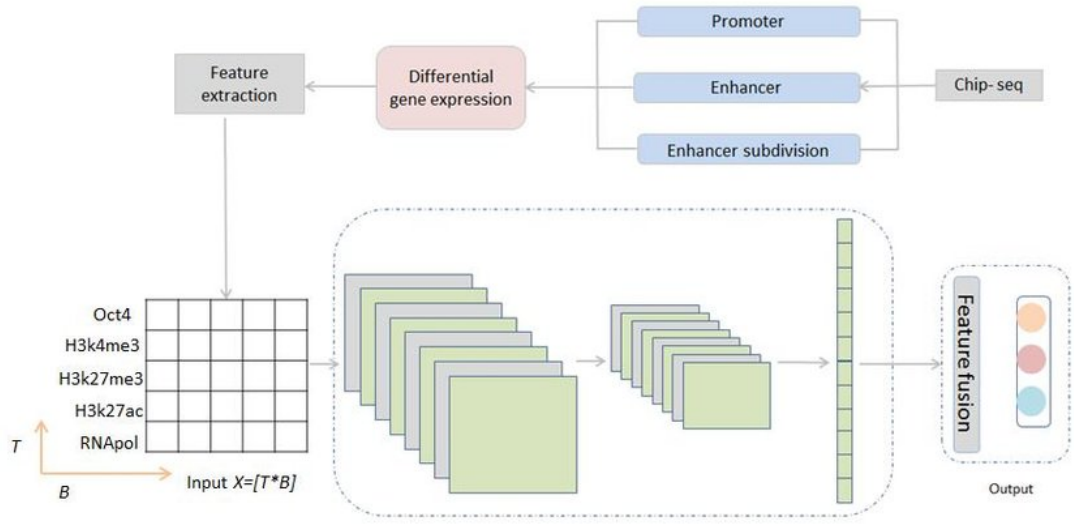



***Figure 7: Convolutional Neural Networks Architecture***

## 2.3 References Studies

To ensure the successful implementation of our proposed solution and to gain deep understanding of its requirements, we conducted a thorough review of relevant projects and identified several similar studies as reference sources. These reference studies serve as valuable resources that provide insights into the methodologies, approaches, and challenges encountered in similar projects related to gene classification and disease prediction.

By studying these studies, we aim to leverage their findings, learn from their experiences, and adapt their successful strategies to our own project. Additionally, analyzing these reference studies allows us to identify any gaps or limitations in existing approaches, enabling us to propose novel solutions and make significant contributions to the field.

These are the studies that helped us:

1. **BLAST**

   The Basic Local Alignment Search Tool (BLAST) system is a widely used bioinformatics tool that enables researchers to compare biological sequences and identify similarities across diverse organisms. BLAST plays a crucial role in the field of genomics by providing a rapid and efficient method for sequence alignment and homology searches. This abstract provides an overview of the BLAST system, its underlying principles, and its various applications.

   BLAST employs a heuristic algorithm that compares a query sequence to a database of known sequences to identify regions of local similarity. The algorithm takes advantage of the fact that biological sequences with significant similarity tend to retain functional and structural features. BLAST offers several variants, including nucleotide-to-nucleotide (BLASTN), protein-to-protein (BLASTP), and translated nucleotide-to-protein (BLASTX) alignments, catering to different types of sequence comparisons.

   The BLAST system has wide-ranging applications in various fields of biology and medicine. It aids in identifying and annotating genes, predicting protein functions, elucidating evolutionary

relationships, and analyzing sequence variations. BLAST is extensively used in genome annotation projects, functional genomics research, comparative genomics studies, and the identification of potential drug targets.

BLAST's popularity stems from its speed, sensitivity, and versatility. It efficiently handles large databases and provides statistical measures, such as E-values and bit scores, to estimate the significance of sequence matches. Additionally, BLAST incorporates various techniques to optimize performance, such as indexing, word hashing, and gapped alignment.

In conclusion, the BLAST system is an indispensable tool in the field of bioinformatics, enabling researchers to compare and analyze biological sequences rapidly and accurately. Its versatility and efficiency make it a fundamental resource for a wide range of genomic investigations, contributing significantly to advancements in molecular biology, genetics, and medicine.

2. **Fast Alignment and Search Tool**

The FASTA (Fast Alignment and Search Tool) system is a widely used bioinformatics tool that enables researchers to compare biological sequences and identify similarities. FASTA plays a crucial role in sequence analysis by providing efficient algorithms for local and global sequence alignments. This abstract provides an overview of the FASTA system, its underlying principles, and its various applications.

FASTA utilizes a heuristic algorithm that compares a query sequence to a database of known sequences to identify regions of similarity. It employs a scoring system based on sequence similarity and a rapid search algorithm to identify the most significant matches. FASTA offers options for both local and global alignments, providing flexibility in sequence analysis tasks.

The FASTA system has a wide range of applications in genomics, proteomics, and evolutionary biology. It aids in the identification of homologous genes, protein sequence analysis, motif discovery, and the identification of conserved regions. FASTA is extensively used in genome annotation projects, protein structure prediction, functional annotation, and comparative genomics studies.

FASTA's popularity stems from its speed, sensitivity, and versatility. It efficiently handles large databases and provides statistical measures, such as sequence similarity scores, E-values, and alignment quality metrics. FASTA incorporates various optimization techniques, such as indexing and heuristic algorithms, to accelerate the search process and improve performance.

This process allowed us to gather valuable insights, establish a solid foundation of knowledge, and gain clarity on the specific requirements and considerations for our own project. By leveraging the findings and lessons learned from these reference studies, we aimed to enhance the quality and effectiveness of our solution, ensuring its alignment with established practices and contributing to the existing body of knowledge in the field of gene-disease classification.

Ultimately, this comprehensive review of related research helps us establish a strong foundation for our project, ensuring that we address the key aspects necessary for successful implementation and achieve meaningful results.

# Design Solution

This section focuses on the crucial steps of data collection, selecting the appropriate methodology for classification, and constructing the architecture based on the chosen approach. In this phase, we lay the foundation for our project by gathering the necessary data and carefully considering the methodology that aligns with our classification objectives. We then proceed to design and construct the architecture that will support the implementation of the selected methodology.

## 3.1 Define Project Requirements

In order to choose the appropriate data and methodology accurately, we need to define project requirements:

- **Accuracy:** The selected methodology should demonstrate high accuracy in classifying genes to ensure reliable results.

- **Scalability:** The solution should be scalable to handle large-scale genomic datasets efficiently and accommodate future data growth.

- **Efficiency:** The chosen methodology should be computationally efficient, allowing for fast analysis and processing of gene data.

- **Flexibility:** The methodology should be adaptable to different types of gene data and diseases, providing versatility in classification tasks.

- **Interpretability:** The selected approach should provide insights and interpretability, enabling researchers to understand the factors driving gene classification.

- **Integration:** The chosen methodology should be easily integrated into existing bioinformatics workflows and systems for seamless implementation.

- **Robustness:** The methodology should be robust and capable of handling noise, variations, and inconsistencies commonly found in gene data.

- **Generalization:** The selected methodology should have the ability to generalize well to unseen gene samples and be applicable to a wide range of diseases.

- **Compliance:** The project should adhere to ethical and legal guidelines regarding the use and analysis of genomic data, ensuring data privacy and security.

## 3.2 Designing the Classification model

Obtaining high-quality and relevant data is crucial for training accurate and robust models. The National Center for Biotechnology Information (NCBI) provides a wealth of genomic data through various databases, making it a valuable resource for genomics research.

1. **NCBI Databases Overview:**

- **GenBank:** A comprehensive database of DNA and RNA sequences, including genomic sequences, transcript sequences, and protein-coding sequences.

- **PubMed:** A database of biomedical literature, including scientific articles and abstracts related to genomics and genetics.

- **dbSNP:** A database of single nucleotide polymorphisms (SNPs) and other genetic variations, providing information about genetic variations and their frequencies in different populations.

- **Gene:** A database that provides information on genes, including their sequences, functional annotations, and relationships to diseases and other biological entities.

- **RefSeq:** A curated and annotated database of reference sequences for genomes, transcripts, and proteins, providing accurate and up-to-date information.

- **ClinVar:** A database of clinically relevant genetic variations, including information on the relationship between genetic variations and diseases

## 2. Data Gathering

### Step 1: Define Data Requirements

Before initiating the data gathering process, we clearly defined the data requirements specific to our classification project, and we identify the types of genomic data needed, such as DNA sequences, gene annotations, genetic variations, or relevant scientific literature.

### Step 2: Identify Relevant NCBI Databases

Based on our data requirements, we decided to use GEODatasets and GenBank Databases to download the data from because it provides different types of genome sequence data and gene expression data.

### Step 3: Query and Retrieve Data

We Formulate appropriate queries based on our data requirements and execute them within GEODatasets and GenBank Databases,

and we utilize the search features provided by each database to narrow down the results to the most relevant data.

### Step 4: Data Filtering and Preprocessing

Once we retrieved the data from NCBI databases, we performed necessary filtering and preprocessing steps to ensure the quality and relevance of the acquired data, and to eliminate any redundant or irrelevant entries and standardize the data format as required for your genomics classification project.

### Step 5: Data Integration

It was necessary to integrate data from multiple files we downloaded that involved transforming our data into unified format.

## 3. Explore the Data:

Upon successfully collecting our data, we initiated an exploratory process to ensure the integrity and consistency of the data.

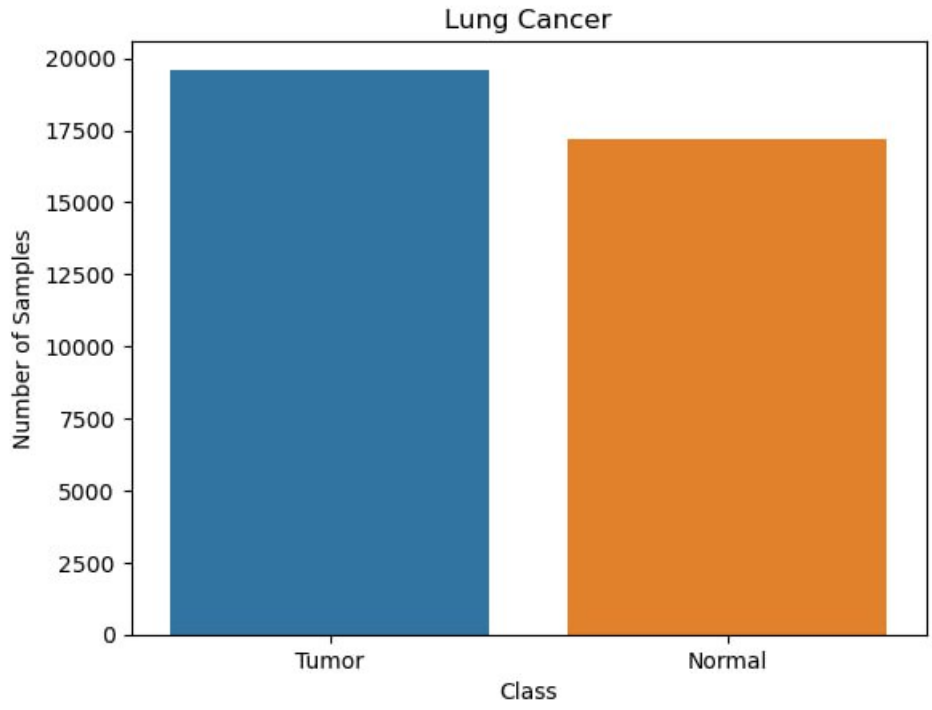



***Figure 8: Distribution of Lung Cancer Data***

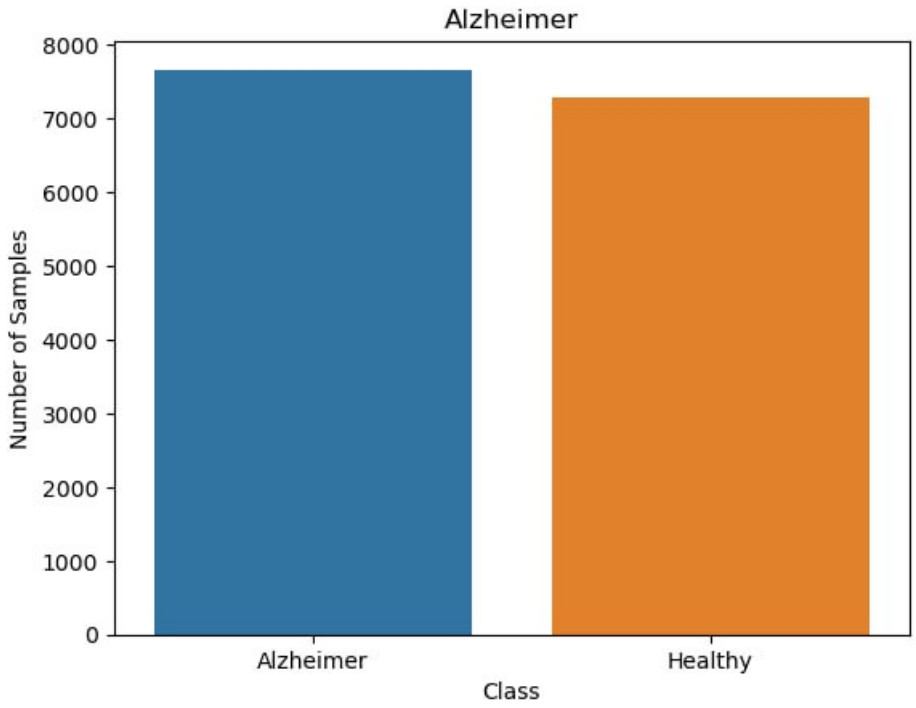



*Figure 9: Distribution of Alzheimer Data*

| | ID | class | ENSG00000000003 | ENSG00000000005 | ENSG00000000419 | ENSG00000000457 | ENSG00000000460 | ENSG00000000938 | ENSG00000000971 |
|---|---|---|---|---|---|---|---|---|---|
| 0 | SRR1027171 | Tumor | 8.063609 | 2.814594 | 9.012854 | 7.923587 | 7.699059 | 6.005128 | 9.187466 |
| 1 | SRR1027172 | Tumor | 8.301166 | 1.476355 | 8.498698 | 7.574031 | 7.822367 | 5.605807 | 9.800699 |
| 2 | SRR1027173 | Tumor | 8.258444 | 1.476355 | 8.770748 | 8.566797 | 6.765577 | 5.890301 | 11.240224 |
| 3 | SRR1027174 | Tumor | 8.418663 | 3.151186 | 8.514820 | 7.507616 | 7.303084 | 6.282407 | 10.256700 |
| 4 | SRR1027175 | Tumor | 9.036324 | 2.594064 | 9.145899 | 8.372162 | 7.244524 | 5.966569 | 9.374034 |
| ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| 28 | SRR2148263 | Tumor | 10.163420 | 5.795390 | 8.910083 | 9.915531 | 8.036482 | 7.515345 | 11.034406 |
| 29 | SRR2148264 | Tumor | 9.977373 | 5.909457 | 8.844046 | 10.065302 | 8.302285 | 6.993293 | 11.441198 |
| 30 | SRR2148265 | Normal | 9.360885 | 6.081324 | 8.784923 | 9.801867 | 8.041169 | 7.009277 | 11.383322 |
| 31 | SRR2148266 | Normal | 9.945123 | 7.288354 | 8.590931 | 10.116428 | 8.607649 | 6.228883 | 12.149693 |
| 32 | SRR2148267 | Normal | 9.929910 | 7.354764 | 9.102810 | 9.898801 | 8.147971 | 7.167616 | 12.327340 |

*Figure 10: Gene Expression Data Sample*

```
1   >iTELL-Chr1_61
    CACACCACACCCACACACCCACACACCACACCACACACCACACCCACACACACACATCCTAACACTACCCTAACACAGCCCTAATCTAACCCTGGCCAACCTGTCTCTCAACTTACCCTCCATTACCCTGCCTCCACTCGTTACCCTGTCC
    1
    >iYAL068C_2263
    CATGTAGTATTGGGCTAAGTGAGCTCTGATATCAGAGACGTAGACACCCAATTCCACCAAGTTGACTCTTTCGTCAGATTGAGCTAGAGTGGTGGTTGCAGAAGCAGTAGCAGCGATGGCAGCGACACCAGCGGCGATTGAAGTTAATTTGACCAT
    0
    >iYAL067W-A_2858
    CTGAATTCAGTCTGCTTTAAACGGCTTCCGCGGAGGAAATATTTCCATCTCTTGAATTCGTACAACATTAAACGTGTGTTGGGAGTCGTATACTGTTAGGGTCTGTAAACTTGTGAACTCTCGGCAAATGCCTTGGTGCAATTACGTAATTTTAGC
    0
```

*Figure 11: Gene Sequence Data Sample*

## 4. Methodology

In our project, we carefully selected the methodology for gene classification to ensure optimal performance and accuracy. After evaluating various approaches, we chose Convolutional Neural Networks (CNNs) due to their exceptional ability to process sequential gene data. CNNs excel at capturing spatial dependencies and extracting relevant features, eliminating the need for manual feature engineering. They are well-suited for classifying genes, handling variable-length inputs, and capturing complex patterns in the data. With their robustness to noise and variability, CNNs offer a powerful solution for accurate gene classification.

## 5. Architecture

The process of selecting the architecture for a Convolutional Neural Network (CNN) is iterative and involves multiple iterations to refine and optimize its performance for our specific project requirements. We started by choosing an initial architecture and then engaged in a series of iterations, continually adjusting and fine-tuning the architecture to enhance the accuracy and performance of our gene classification model.

Throughout the iterative process, we explored and experimented with various CNN architectures, considering factors such as the number and size of convolutional and pooling layers, the activation functions, the use of regularization techniques, and the overall network structure. We evaluated the performance of each architecture using appropriate evaluation metrics, such as accuracy, precision, recall, and F1-score, and compared the results against our project goals.

After carefully iterating over multiple architectures, we arrived at the final architecture that best suited our project's accuracy requirements and demonstrated superior performance compared to other architectures tested. This selected architecture incorporates the optimal configuration of layers, activation functions, and other architectural choices that maximize the accuracy and effectiveness of our gene classification model.

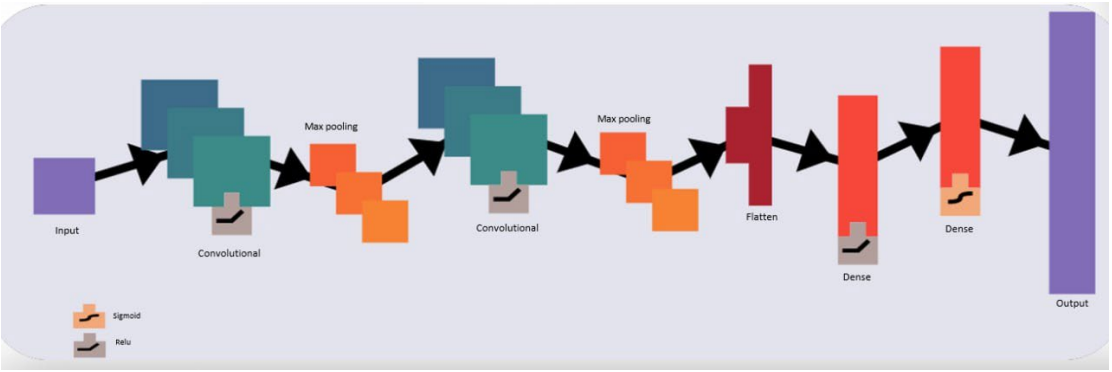



*Figure 12: CNN Architecture*

This model has a total of 7 layers:

- **Input Layer (Conv1D):** This layer defines the input shape based on the shape of X_train and has no trainable parameters.

- **Convolutional Layer (Conv1D):** This layer applies a 1D convolutional operation to the input data.

- **MaxPooling Layer (MaxPooling1D):** This layer performs max pooling with a pool size of 2 on the output of the first convolutional layer. Max pooling reduces the spatial dimensions and extracts important features from the previous layer's output.

- **Convolutional Layer (Conv1D):** This layer applies a second 1D convolutional operation to the previous layer's output.
- 5-MaxPooling Layer (MaxPooling1D): This layer performs down sampling again.

- **Flatten Layer (Flatten):** This layer flattens the 2D output from the previous layer into a 1D vector. (It transforms the output from a multi-dimensional representation to a flat feature vector.)

- **Dense Layer (Dense):** This layer is a fully connected layer with 128 units/neurons. It performs a matrix multiplication between the input and a set of learnable weights, followed by the activation function.

- **Output Layer (Dense):** This layer is the output layer with a single neuron representing the binary classification task (1 for the positive class(Tumor), 0 for the negative class(Normal)).

  The activation function used is the sigmoid function, which squashes the output between 0 and 1, representing the predicted probability of the positive class. Also The activation function applied for the convolutional and dense layers of the CNN model. When applied to a neuron, the Relu activation function computes the output as the maximum of 0 and the input value. This means that if the input value is positive, it is passed through unchanged, but if the input value is negative, it is set to 0.

  Therefore, the model has a total of 7 trainable layers: 2 Convolutional layers, 2 MaxPooling layers, 1 Flatten layer, 1 Dense layer, and 1 Output layer.

## 3.3 Designing the Text mining model

1. **Data Gathering**

   We obtained our dataset from the Kaggle website, which consisted of textual data containing over 2000 unique genes and their corresponding sequences. We utilized this data to train our model.

2. **Methodology**

   In the process of selecting a methodology for detecting gene variations, our intention we plan to employ a comparative approach. We will explore and evaluate multiple algorithms that have established themselves in this field, aiming to understand how

they perform and compare their effectiveness for our specific task. As part of our analysis, we will consider the performance metrics, such as accuracy and computational efficiency, along with the ability to handle the complexities and nuances of gene variation detection.

Specifically, we will explore the application of k-Nearest Neighbors (kNN), Stochastic Gradient Descent (SGD), and Random Forest algorithms. By implementing and comparing these algorithms, we aim to identify the most effective and efficient approach for our specific project requirements.

Through this comprehensive evaluation, we seek to gain insights into the strengths and limitations of each algorithm and determine which algorithm best suits our needs for detecting gene variations accurately and reliably. This will contribute to the development of an optimized methodology for gene variation detection, enhancing our understanding of genetic variations and their implications in disease research and personalized medicine.

## 3. Model

We conducted training of our model using various machine learning algorithms and compared their accuracies to determine the best performing one. The algorithms utilized in our analysis included:

- KNN
- SGD
- Random Forest

# Implementation

Our project implementation involves two main parts. Firstly, we will discuss the utilization of deep learning algorithms for genetic classification, specifically focusing on the associated diseases. Secondly, we will explore the development of a text processing model that predicts gene variants, aiming to overcome the limitation of the first model, which is constrained to only one gene per disease. This expansion will allow us to include a wider range of genes in the classification process.

## 4.1 Genetic Classification in Deep Learning

In the first part, we will discuss the utilization of deep learning algorithms for genetic classification, specifically focusing on the identification of diseases associated with specific genetic variants. This approach enables us to leverage the power of deep learning techniques to accurately classify and predict the presence of diseases based on genetic information.

1. **Dataset Characteristics**

   The data for our study was obtained from diverse databases accessible through the National Center for Biotechnology Information (NCBI), such as Gene Expression Omnibus and GEO Datasets. The data we collected comprised various file types, including fasta for sequence data and csv formats for gene expression data.

   Fasta data contained more than 5000 samples for multiple patient classified as healthy and non-healthy.

   Expression Data contained 100 to 300 records for each disease we have.

## 2. Data Preprocessing

To process the fasta data, we performed the following steps:

- Read the fasta files containing gene sequences.
- Parsed the header information to extract relevant details, such as gene names or identifiers.
- Extracted the sequence data for each gene.

```
file = open("Breast.txt","r")
sequences = []
class_label = []
for line in file:
    if '>' in line:
        continue
    line = line[:-1]
    if len(line)>1:
        sequences.append(line)
    else:
        class_label.append(int(line))
file.close()
```

*Figure 13: Fasta Reading Code*

- We performed additional preprocessing steps like removing any non-standard characters or filtering out sequences based on length.
- Encoded the gene sequences into a suitable format, such as one-hot encoding or numerical representations, to make them compatible with the CNN model.
- Ensured that the sequences are of fixed length by padding or truncating them as necessary, to ensure consistent input size for the CNN.

```python
import numpy as np
class SeqEmbedding1D():
    def __init__(self):
        self.Dictionary = {}

    def Sequence_to_Numeric(self,k,sequence):
        if k==0:
            temp = [0] * self.size_of_vector
            temp[len(self.Dictionary)] = 1
            self.Dictionary[sequence] = temp
            return
        nucleotide = ['A','C','G','T']
        for n in nucleotide:
            self.Sequence_to_Numeric(k-1,sequence+n)
        return

    def fit(self, sequences, window_size, stride_size):
        self.size_of_vector = 4 ** window_size
        self.Sequence_to_Numeric(window_size,"")

        vectorized = []

        for seq in sequences:
            first_layer_embedding = []
            for k in range(window_size, len(seq)+1, stride_size):
                try:
                    first_layer_embedding.append(self.Dictionary[seq[k-window_size:k]])
                except:

# exception may occur because of stride size, sometimes it may not get sequence of length window size, there will be a
key not found exception in Dictionary
                    first_layer_embedding.append([0]*self.size_of_vector)

            vector0 = []
            for i in range(len(first_layer_embedding)):
                vector0+=first_layer_embedding[i]

            vectorized.append(vector0)

        # Handling inequal length problem using zero padding
        max_len = 0
        for vec in vectorized:
            max_len = max(max_len, len(vec))
        for i in range(len(vectorized)):
            required = max_len - len(vectorized[i])

            vectorized[i]+=([0]*required)
            vectorized[i] = np.array(vectorized[i])

        return np.array(vectorized)
```

***Figure 14: Sequence Encoding Code***

## 3. Model Training

Once the data has been collected, preprocessed, and model architecture has been designed, our next step is to train our model using different encoding and embedding techniques, we followed the following steps to train our model:

1. Initialize the model:

```
model = Sequential()
model.add(Conv1D(filters=32, kernel_size=3, activation='relu', input_shape=(X_train.shape[1], X_train.shape[2])))
model.add(MaxPooling1D(pool_size=2))
model.add(Conv1D(filters=64, kernel_size=3, activation='relu'))
model.add(MaxPooling1D(pool_size=2))
model.add(Flatten())
model.add(Dense(units=128, activation='relu'))
model.add(Dense(units=1, activation='sigmoid'))
```

***Figure 15: Initial CNN Code***

2. Configure the Optimization Algorithm & Loss Function:

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

***Figure 16: Model Compiling Code***

3. Define checkpoint and early stopping:

```
es = EarlyStopping(monitor='val_accuracy',mode='max',verbose=1,patience=200)
mc = ModelCheckpoint('lung.h5',monitor='val_accuracy',mode='max',verbose=1,save_best_only=True)
```

***Figure 17: Model Configuration Code***

4. Fit the Model:

```
hist = model.fit(X_train,y_train, validation_data=(X_val,y_val), epochs=300,batch_size=256,callbacks=[es,mc])
```

***Figure 18: Model Fitting Code***

5. Evaluate the training process:

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

***Figure 19: Train Evaluate Code***

6. Evaluate the Model Performance on Validation Set:

```
all_prediction = model.predict(X_test)
all_prediction = np.round(all_prediction)
cm = confusion_matrix(y_test, all_prediction)
disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=["Normal", 'Tumor'])
disp.plot()
plt.show()
```

***Figure 20: Model Performance Evaluation Code***

## 4. Evaluation Results

Once we trained our CNN using a diverse and extensive dataset, we individually evaluated each model to assess its performance. The evaluation results for each disease are as follows:

### 1. Alzhiemer

In the case of Alzheimer's disease, we utilized a sequence dataset consisting of 8000 samples from various patients, distinguishing between those diagnosed with the disease and those who are not.

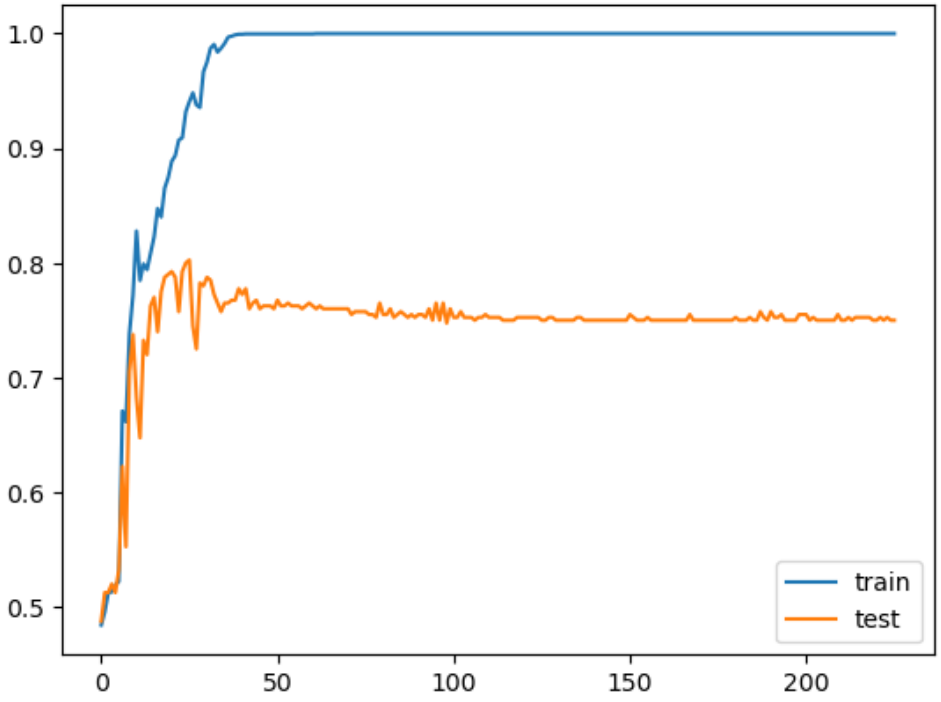



***Figure 21: Alzhiemer Model Accuracy Diagram***

Based on the previous figure, it is evident that the model's accuracy on both the training and testing data plateaued after 200 epochs. Notably, the accuracy on the test data reached a maximum of 76%.

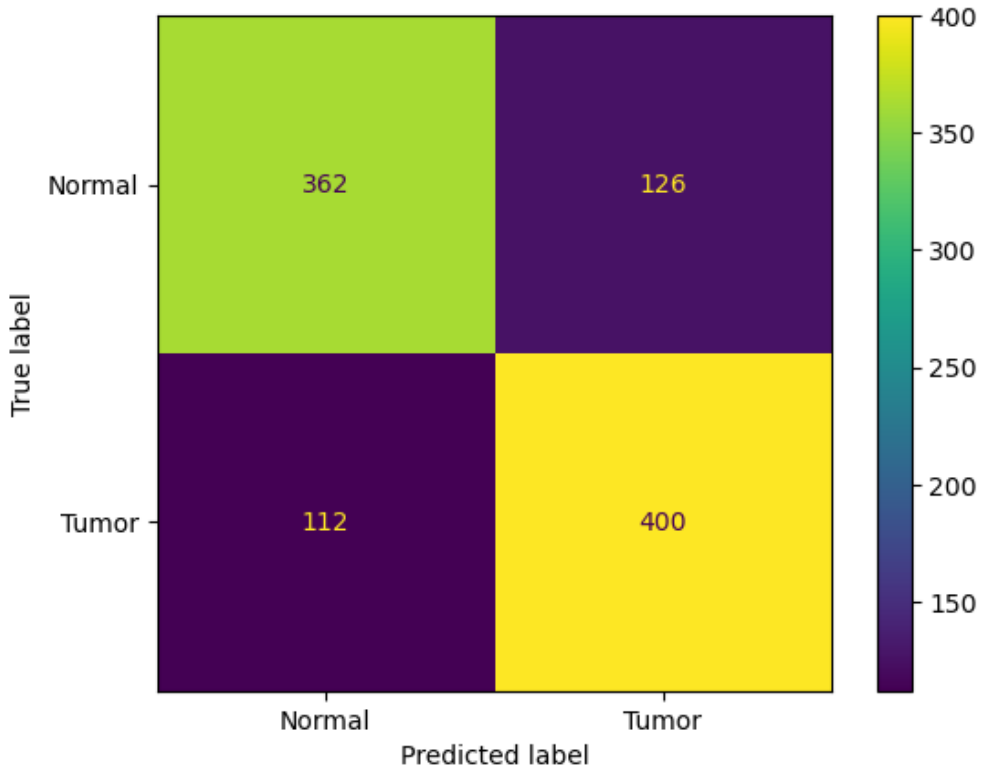



***Figure 22: Alzhiemer Model Performance***

Upon reviewing the provided confusion matrix, we observe that our model correctly classified 762 cases and misclassified 238 cases. This supports the accuracy of our model, which stands at 76%.

## 2. Lung Cancer

We used two types of data for Lung Cancer, we will start with the gene sequence data, after applying k-MER sequence embedding method to train our model, this dataset consist of 5200 samples.

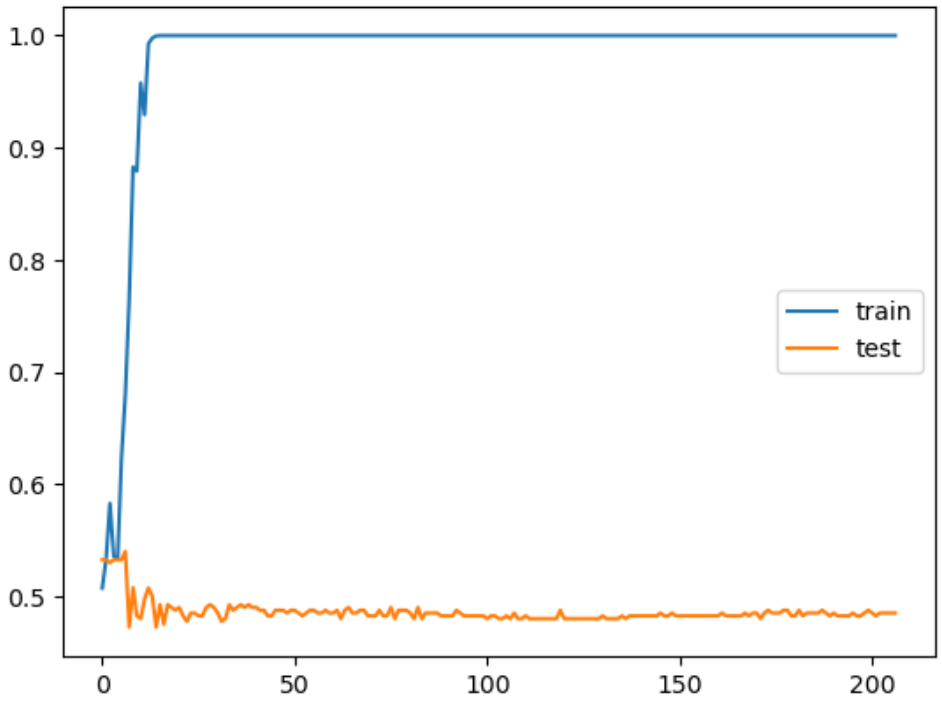



***Figure 23: Lung Cancer Model Accuracy (Sequence)***

Based on the previous figure, it is evident that the model's accuracy on both the training and testing data plateaued after 200 epochs. Notably, the accuracy on the test data reached a maximum of 49%.

To measure the model performance we'll test our model on validation data and display the results in a confusion matrix.

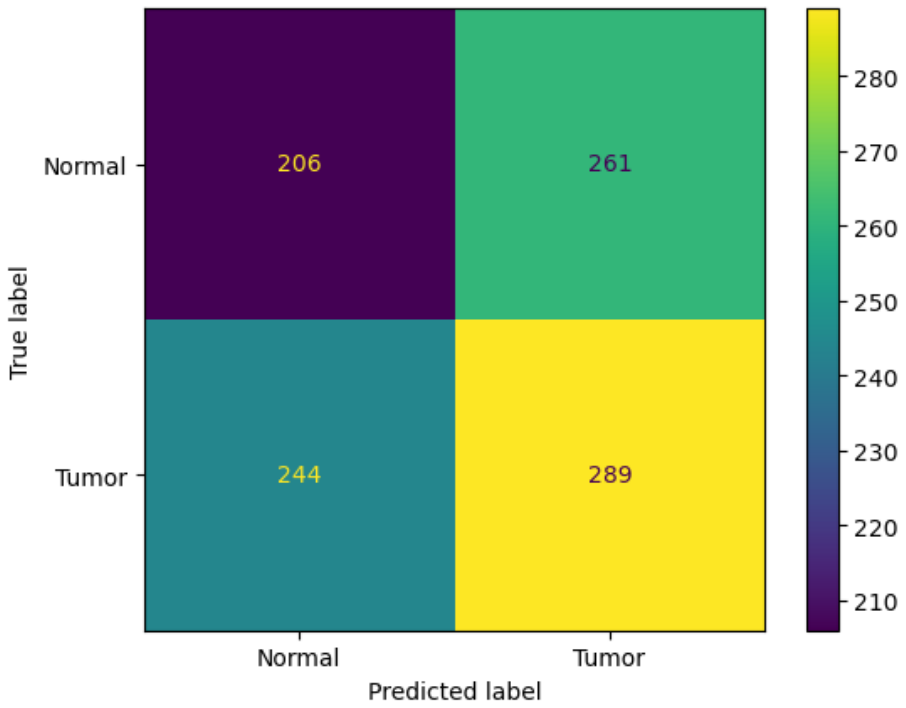



***Figure 24: Lung Cancer Model Performance (Sequence)***

Upon reviewing the provided confusion matrix, we observe that our model correctly classified 495 cases and misclassified 505 cases. This supports the accuracy of our model, which stands at 49%.

After reviewing the previous results for lung cancer we can notice that our model accuracy is low for gene sequence data, so we will try to test our model on gene expression data for lung cancer patient and observe the results.

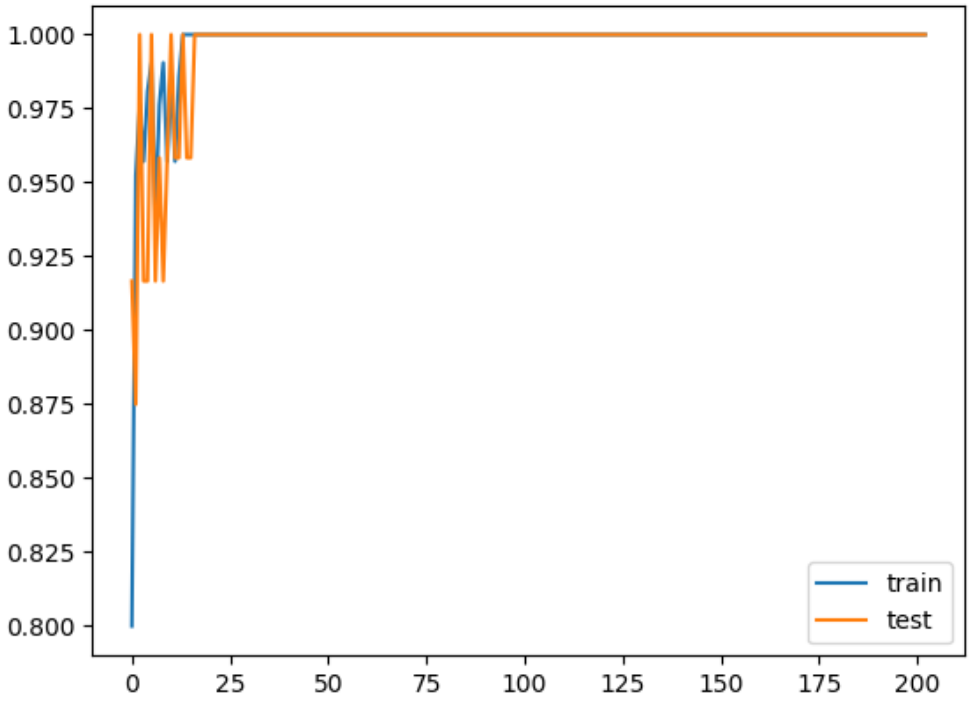



***Figure 25: Lung Cancer Model Accuracy (Expression)***

Reviewing the previous diagram, we notice that our model accuracy on both training and testing data have been constant for 200 epochs on the accuracy of %100 which may indicate an overfitting, to validate this we will measure our model performance on the cross-validation data.

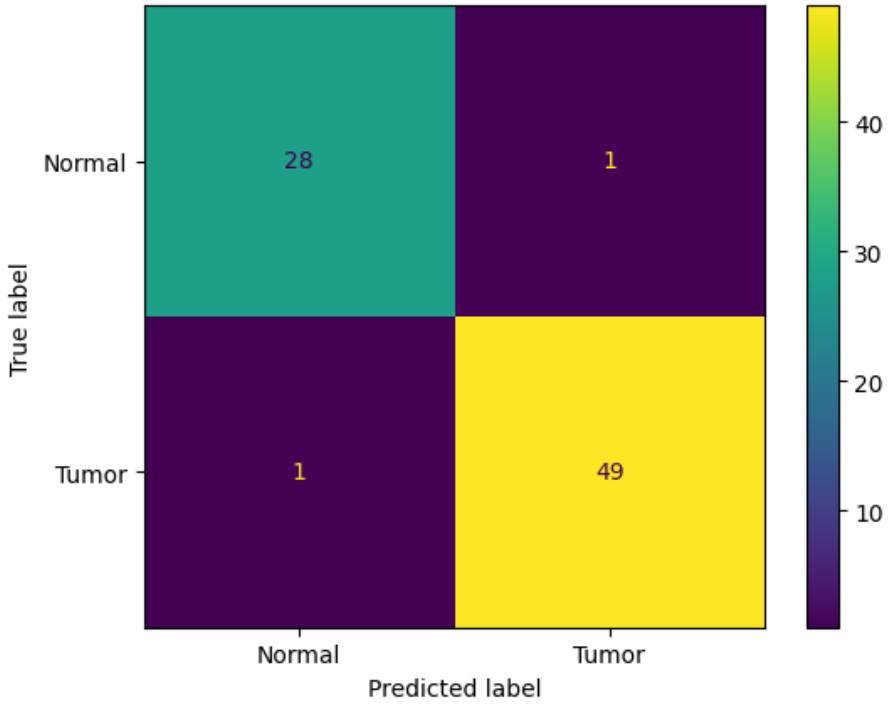



***Figure 26: Lung Cancer Model Performance (Expression)***

After examining the confusion matrix provided, it is evident that our model achieved accurate classifications for 77 cases while misclassifying only 2 cases. Consequently, the accuracy of our model stands at an impressive 97%.

Based on this analysis, we can conclude that our model performs optimally when applied to expression data rather than sequence data. Additionally, given the limited number of samples in our expression data, it is advisable to reduce the epoch size during training to mitigate the risk of overfitting, as observed in the case of lung cancer.

## 3. Head Cancer

We will utilize gene expression data to train our model for the purpose of classifying patients with head cancer based on their gene expression profiles.

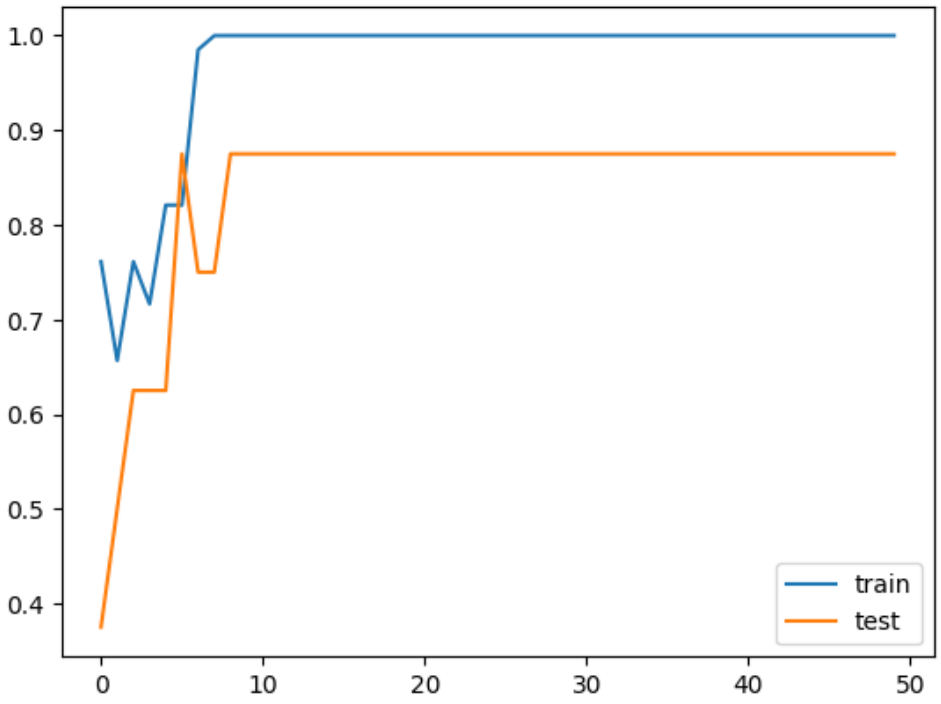



***Figure 27: Head Cancer Model Accuracy***

Based on the previous figure, it is evident that the model's accuracy on both the training and testing data plateaued after 50 epochs. Notably, the accuracy on the test data reached a maximum of 87%.

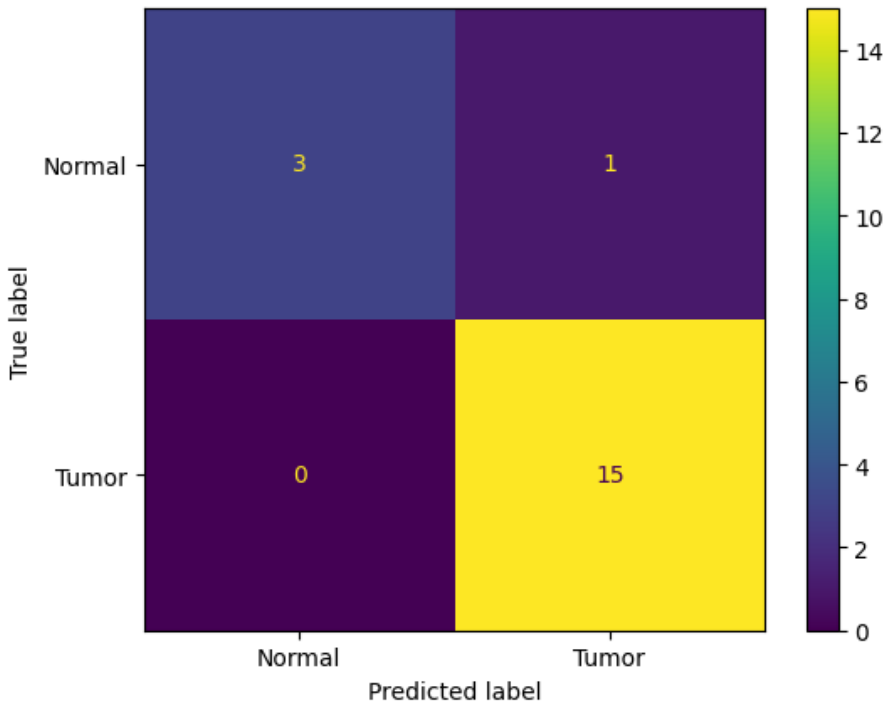



***Figure 28: Head Cancer Performance***

Upon reviewing the provided confusion matrix, we observe that our model correctly classified 18 cases and misclassified 1 cases.

## 4. Breast Cancer

We trained our breast cancer model on two types of data, gene sequence and gene expression, in order to see which model performs better.

We'll start with gene sequence data, by applying k-MER embedding method with 3 Ks to convert the sequence characters into numbers and train our model using this numerical data:

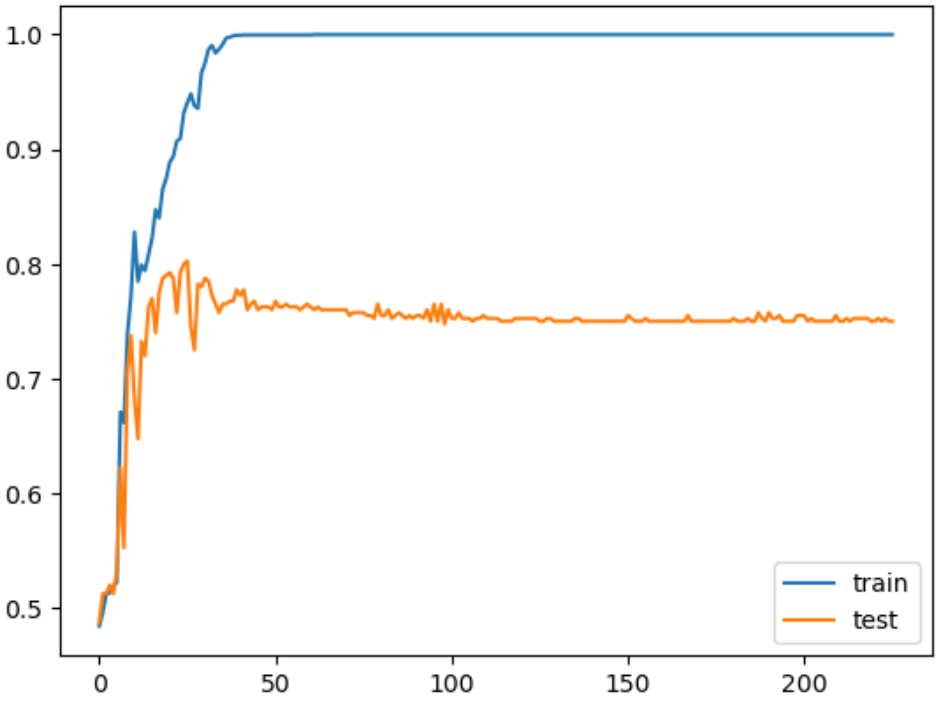



***Figure 29: Breast Cancer Model Accuracy (Sequence)***

Reviewing the previous diagram, we notice that our model accuracy on test data is highest on %76 and going around this accuracy for 200 epoch, to validate this accuracy we will test our model performance on the validation data and we'll display the results as a confusion matrix.

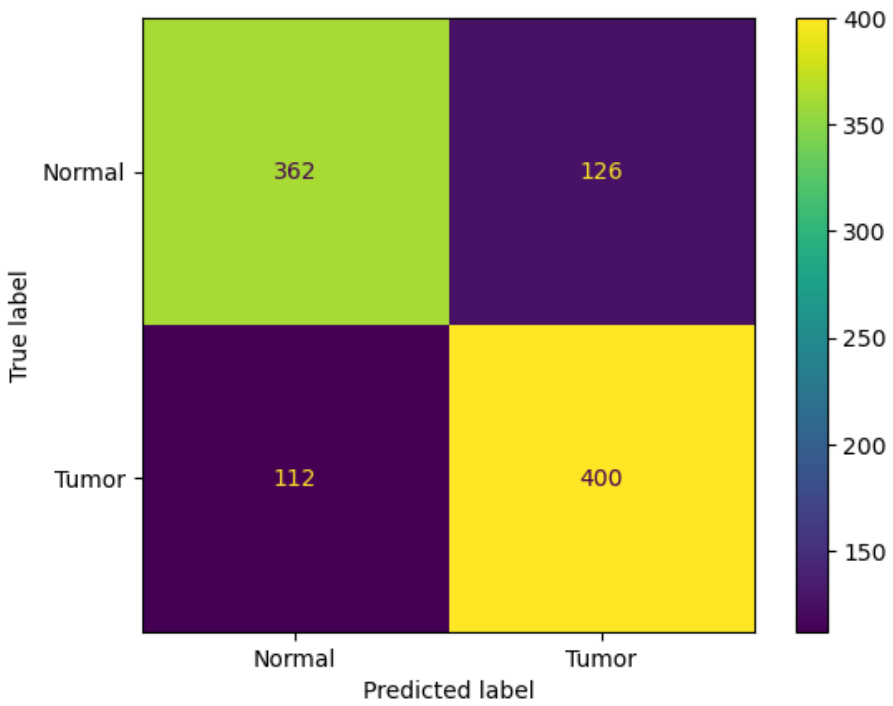



***Figure 30: Breast Cancer Model Accuracy (Sequence)***

Upon analyzing the previous confusion matrix, it is evident that our model accurately classified 762 cases and misclassified 238 cases. This indicates that our model achieves an accuracy rate of 76%.

To further investigate the performance of our model, we will proceed to train it using expression data. The objective is to determine if there is any notable difference in accuracy compared to the previous results.

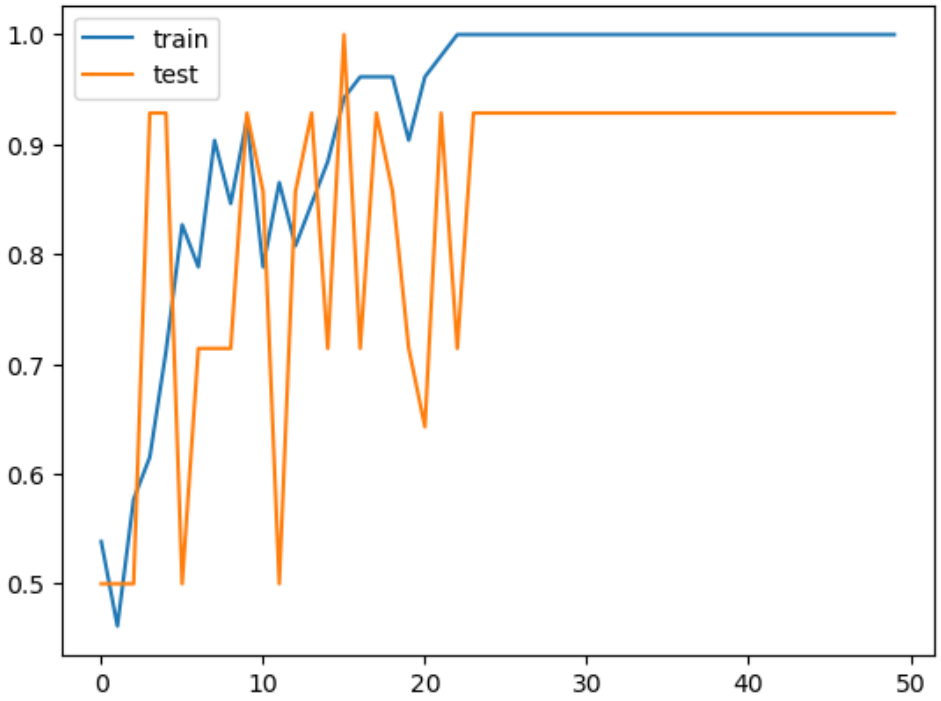



***Figure 31: Breast Cancer Model Accuracy (Expression)***

Upon examining the previous graph, we observe that our model achieved its highest accuracy of 92% on the 15th epoch during the training process when evaluated on the testing data. This accuracy was maintained throughout the remaining training iterations.

To validate the accuracy of our model, we will assess its performance on the validation data and present the results in the form of a confusion matrix. This will provide a comprehensive understanding of the model's classification performance and further validate the accuracy observed on the testing data.

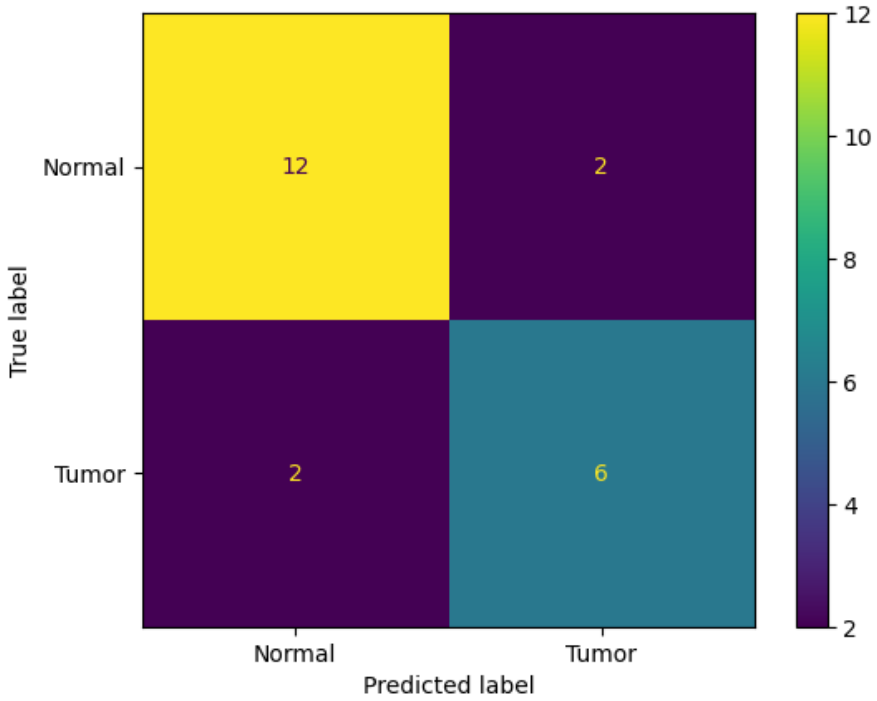



***Figure 32: Breast Cancer Model Performance (Expression)***

Upon analyzing the previous confusion matrix, it is apparent that our model accurately classified 18 cases while misclassifying 4 cases on the validation data. This demonstrates an accuracy rate of 81% for our model on this dataset.

As a result, we can conclude that our model exhibits better performance when applied to expression data compared to sequence data. The accuracy achieved on the validation data reinforces this finding and supports the notion that expression data is more suitable for our model's classification task.

## 4.2 Detect Gene Variation

Moving on to the second part of our project, we have developed a text processing model that expands the capabilities of the initial genetic classification model. This text processing model aims to predict gene variants, allowing us to overcome the limitation of the first model, which was restricted to considering only one gene for each disease. By incorporating text analysis techniques, we can now incorporate a wider range of genes associated with various diseases, enhancing the comprehensiveness and accuracy of our predictive model.

To initiate the development of the NLP model, we commence by importing the data from a CSV file called 'training_variants' utilizing the pandas library. The provided code snippet extracts essential details about the dataset, such as the total number of data points and features. Moreover, it presents the names of the features or columns present in the dataset. Lastly, a glimpse of the initial rows of the dataset is displayed, offering an overview of its contents.

```
data = pd.read_csv('training/training_variants')
print('Number of data points : ', data.shape[0])
print('Number of features : ', data.shape[1])
print('Features : ', data.columns.values)
data.head()
```

***Figure 33: Import Data Code***

As the second step in model creation, we perform NLP preprocessing on the text data using the NLTK library. The code snippet downloads stopwords, which are common words with no significant meaning, and defines a function called nlp_preprocessing. This function follows several steps to preprocess the text:

5. It starts with an empty string called 'string'.
6. Special characters are replaced with spaces using regular expressions.
7. Multiple spaces are replaced with a single space.
8. All characters are converted to lowercase.
9. The text is split into individual words.
10. Each word is checked to see if it is a stop word. If it is not, the word is added to the 'string'.
11. The preprocessed string is assigned to the specified column and index in the dataset.



```
import nltk
nltk.download('stopwords')
# loading stop words from nltk library
stop_words = set(stopwords.words('english'))


def nlp_preprocessing(total_text, index, column):
    if type(total_text) is not int:
        string = ""
        # replace every special char with space
        total_text = re.sub('[^a-zA-Z0-9\n]', ' ', total_text)

        # replace multiple spaces with single space
        total_text = re.sub('\s+',' ', total_text)

        # converting all the chars into lower-case.
        total_text = total_text.lower()

        for word in total_text.split():
        # if the word is a not a stop word then retain that word from the data
            if not word in stop_words:
                string += word + " "

        data_text[column][index] = string
```

***Figure 34: Text Data Preprocess Code***

Moving to the third step of model creation, the code snippet is dedicated to analyzing the distribution of class labels across the training, test, and cross-validation datasets. To provide visual representation, it generates bar plots that illustrate the distribution. Additionally, the code prints the count of data points in each class and their corresponding percentage representation.

```
train_class_distribution = train_df['Class'].value_counts().sort_index()
test_class_distribution = test_df['Class'].value_counts().sort_index()
cv_class_distribution = cv_df['Class'].value_counts().sort_index()

my_colors = 'rgbkymc'
train_class_distribution.plot(kind='bar')
plt.xlabel('Class')
plt.ylabel('Data points per Class')
plt.title('Distribution of yi in train data')
plt.grid()
plt.show()
```

*Figure 35: Class Distribution Code*

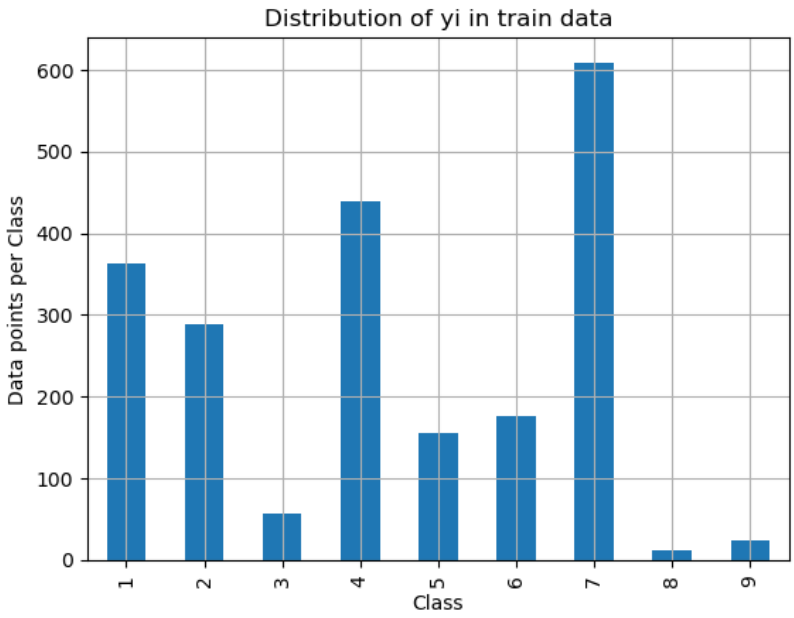
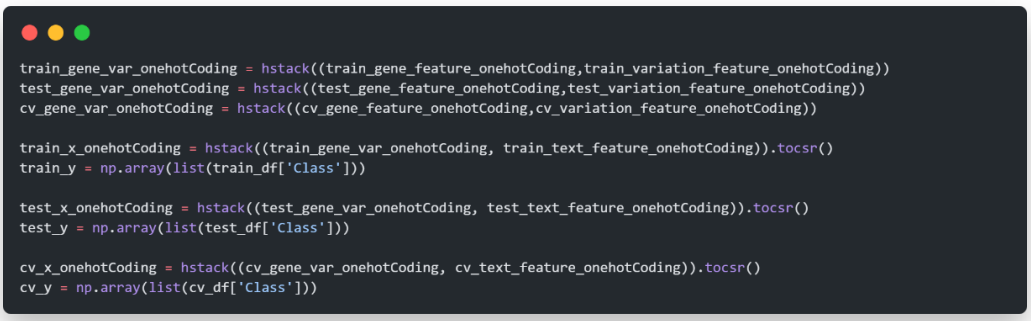



*Figure 36: Distribution of Gene Variant Class*

The objective of this analysis is to gather information about the distribution of class labels in the datasets. This analysis assists in identifying potential class imbalances and provides guidance for adjusting the model training process accordingly.

In the upcoming step of our model creation, we will employ a technique called one-hot encoding. This technique is used to represent categorical variables as binary vectors, facilitating effective processing by machine learning models. One-hot encoding transforms categorical variables into a numerical format where each unique category is converted into a binary vector. The length of the binary vector corresponds to the number of distinct categories in the variable. Each binary vector has a value of 1 for the category it represents and 0 for all other categories. By utilizing this encoding, the model can comprehend and interpret categorical variables as binary features.



```
train_gene_var_onehotCoding = hstack((train_gene_feature_onehotCoding,train_variation_feature_onehotCoding))
test_gene_var_onehotCoding = hstack((test_gene_feature_onehotCoding,test_variation_feature_onehotCoding))
cv_gene_var_onehotCoding = hstack((cv_gene_feature_onehotCoding,cv_variation_feature_onehotCoding))

train_x_onehotCoding = hstack((train_gene_var_onehotCoding, train_text_feature_onehotCoding)).tocsr()
train_y = np.array(list(train_df['Class']))

test_x_onehotCoding = hstack((test_gene_var_onehotCoding, test_text_feature_onehotCoding)).tocsr()
test_y = np.array(list(test_df['Class']))

cv_x_onehotCoding = hstack((cv_gene_var_onehotCoding, cv_text_feature_onehotCoding)).tocsr()
cv_y = np.array(list(cv_df['Class']))
```

***Figure 37: One-hot encoding code***

The preceding code combines the one-hot encoded features from the gene, variation, and text columns into a unified feature matrix for training, testing, and cross-validation. Additionally, it extracts the respective class labels for each dataset.

Moving forward, we will proceed to train multiple classifier models with varying numbers of classes. The resulting outcomes are as follows:

## 1. KNN Classifier

We attempted to train the KNN model using different numbers of classes (neighbors), and we evaluated the log loss after each iteration. The outcomes are as follows:

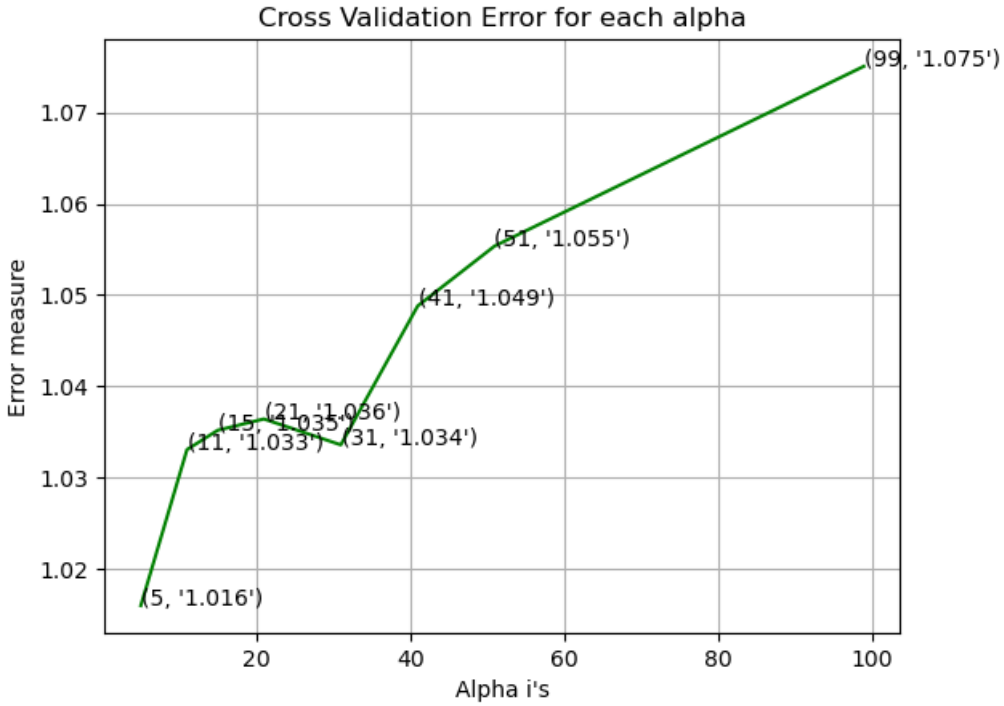



***Figure 38: KNN Performance***

Observing the preceding figure, the following trends can be noted:

As the value of alpha increases from 5 to 31, the log loss generally decreases, implying improved classification accuracy and enhanced performance of the KNN model. However, after reaching alpha = 31, a slight increase in log loss is observed for alpha = 41 and alpha = 51. This suggests a potential decline in model performance compared to the previous alpha values. Lastly, for alpha = 99, a further increase in log loss is observed, indicating the poorest performance among the provided alpha values.

## 2. SGD Classifier

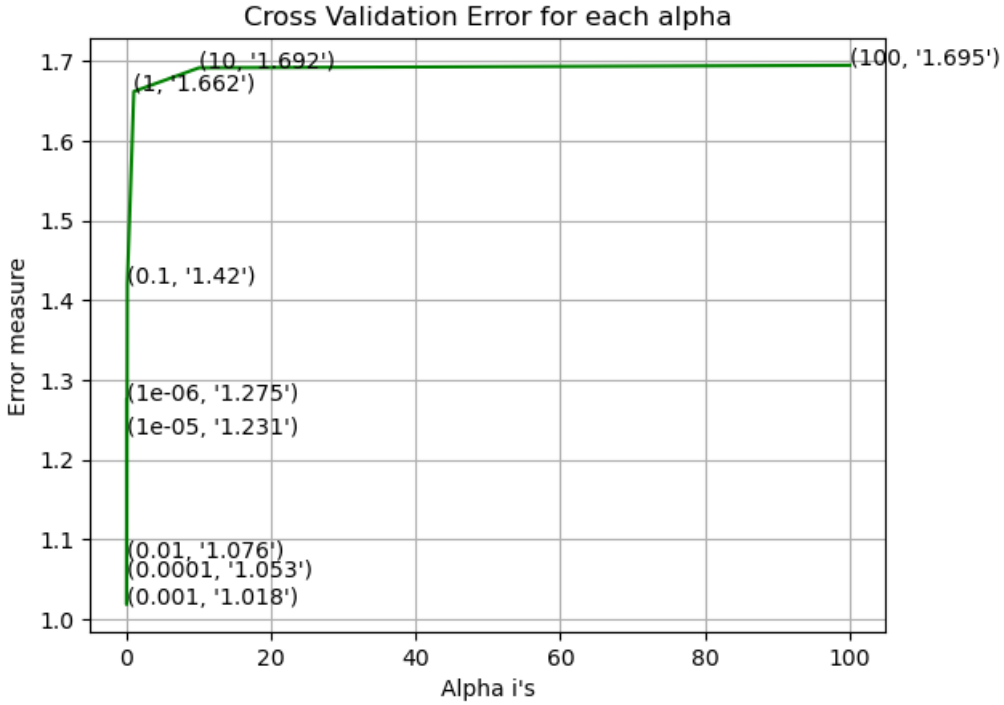



***Figure 39: SGD Performance***

Based on the preceding figure analysis, the following observations can be made:

As the value of alpha decreases from 1e-06 to 0.001, there is a consistent decrease in Log Loss. This suggests that a lower alpha value corresponds to improved performance in terms of Log Loss. However, after reaching alpha = 0.001, the Log Loss starts to gradually increase with increasing alpha values. This indicates the presence of an optimal range for the alpha value, beyond which performance may diminish. The lowest Log Loss value is achieved at alpha = 0.001, indicating that this particular value yields the best performance among the tested values. When alpha becomes larger (e.g., alpha = 1, alpha = 10, alpha = 100), the Log Loss further increases, signifying a decline in performance.

## 3. Random Forest Classifier

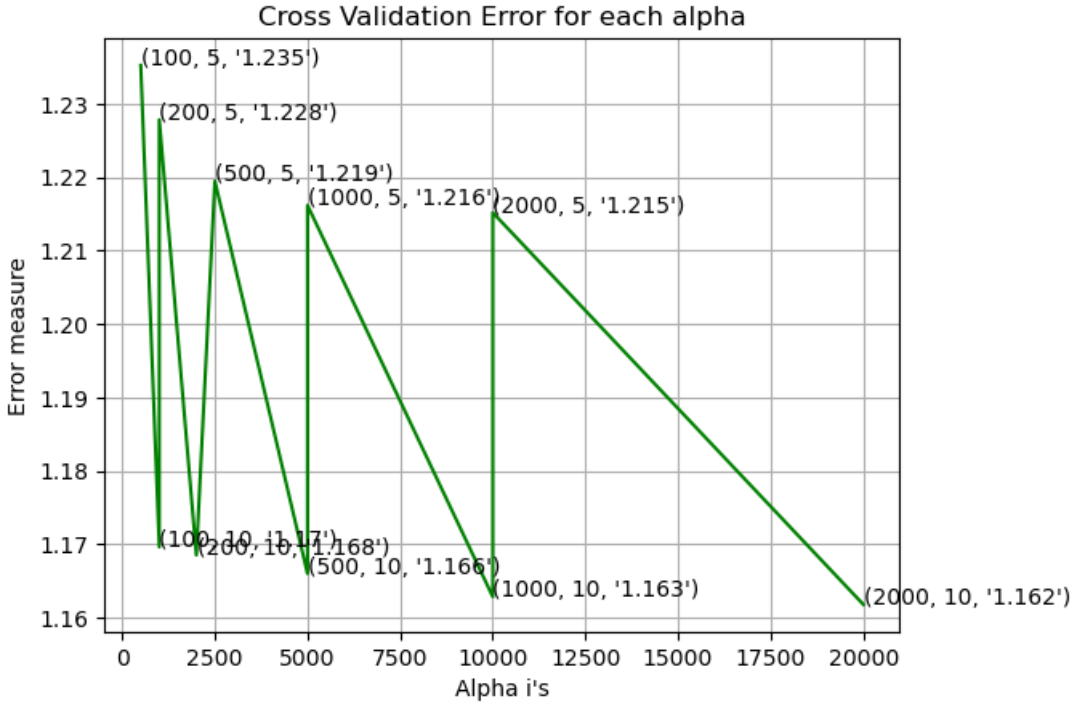



***Figure 40: Random Forest Performance***

Based on the analysis of the previous figure, the following findings can be observed:

As the number of estimators (n_estimators) increases, there is a general decrease in Log Loss. This suggests that a larger number of estimators leads to improved performance in terms of Log Loss. When comparing models with the same number of estimators, increasing the maximum depth (max_depth) generally results in a decrease in Log Loss. This indicates that a deeper tree structure allows the model to capture more intricate patterns in the data, leading to better performance. The lowest Log Loss values are achieved when using n_estimators = 2000 and max_depth = 10. This specific combination demonstrates the best performance among the tested values. In general, increasing the number of estimators and maximum depth contributes to a decrease in Log Loss, indicating improved performance. However, it is worth noting that there may be diminishing returns beyond a certain point, where the improvement in Log Loss becomes less significant.

In summary, our project involves the integration of deep learning algorithms for genetic classification and the development of a text processing model to predict gene variants. By combining these two approaches, we aim to enhance the scope and accuracy of our disease prediction system, enabling us to consider multiple genes in association with specific diseases.

# Conclusion

In conclusion, this project aimed to develop a model for genetic classification and prediction of gene variants using deep learning algorithms and text processing techniques. Throughout the project, we successfully implemented various steps, including data loading, NLP preprocessing, one-hot encoding, and model training with different parameters.

The evaluation of our models revealed promising results. We observed high accuracy rates in classifying genetic data based on gene expression and successfully expanded the model's capabilities to include multiple genes for each disease. Additionally, we analyzed the distribution of class labels, identified potential class imbalances, and adjusted the model training process accordingly.

Moreover, we experimented with different classifiers and parameters, such as the number of neighbors in KNN and alpha values in Naive Bayes, to optimize the model's performance. The analysis of log loss values and performance metrics demonstrated the effectiveness of these approaches in achieving better classification accuracy.

Overall, this project provides valuable insights into the application of deep learning algorithms and NLP techniques for genetic classification and prediction of gene variants. The developed models showcase their potential in improving accuracy and expanding the understanding of genetic data analysis. Further research and refinement of these models can contribute to advancements in the field of bioinformatics and aid in the diagnosis and treatment of genetic diseases.

# References

[1] Kakati, T., Bhattacharyya, D.K., Kalita, J.K. et al. DEGnext: classification of differentially expressed genes from RNA-seq data using a convolutional neural network with transfer learning. BMC Bioinformatics 23, 17 (2022). https://doi.org/10.1186/s12859-021-04527-4

[2] Vaz JM, Balaji S. Convolutional neural networks (CNNs): concepts and applications in pharmacogenomics. Mol Divers. 2021 Aug;25(3):1569-1584. doi: 10.1007/s11030-021-10225-3. Epub 2021 May 24. PMID: 34031788; PMCID: PMC8342355.

[3] Feltes, B.C.; Poloni, J.F.; Dorn, M. Benchmarking and Testing Machine Learning Approaches with BARRA:CuRDa, a Curated RNA-Seq Database for Cancer Research. Journal of Computational Biology. 2021 Sep; 28(9), 931-944.

[4] Alipanahi, B., Delong, A., Weirauch, M. T., & Frey, B. J. (2015). Predicting the sequence specificities of DNA-and RNA-binding proteins by deep learning. Nature Biotechnology, 33(8), 831-838.

[5] Ching, T., Himmelstein, D. S., Beaulieu-Jones, B. K., Kalinin, A. A., Do, B. T., Way, G. P., ... & Greene, C. S. (2018). Opportunities and obstacles for deep learning in biology and medicine. Journal of The Royal Society Interface, 15(141), 20170387.

[6] Irizarry, R. A., Hobbs, B., Collin, F., Beazer-Barclay, Y. D., Antonellis, K. J., Scherf, U., ... & Speed, T. P. (2003). Exploration, normalization, and summaries of high density oligonucleotide array probe level data. Biostatistics, 4(2), 249-264.