Republic of Iraq

Ministry of Higher Education and Science Research

University of Babylon

College of Science for Women

Department of Computer Science

Diabetic Mellitus Detection Based on DNA Data and Machine Learning Techniques

A Thesis

Submitted to the council of College of Science for woman,

University of Babylon in Partial Fulfilment of the Requirement

For Degree of Master of Science in Computer Science

By:

Lena Abed Al- Raheem Hamza

Supervisors

Prof. Dr. Hussein Attia Lafta

Asst. Prof. Dr. Sura Zaki AL-Rashid

2023 A.D. **1445 A.H.**

بِسْمِ ٱللَّهِ ٱلرَّحْمَٰنِ ٱلرَّحِيمِ

﴿ هُوَ الَّذِي جَعَلَ الشَّمْسَ ضِيَاءً وَالْقَمَرَ نُورًا وَقَدَّرَهُ مَنَازِلَ لِتَعْلَمُوا عَدَدَ السِّنِينَ وَالْحِسَابَ مَا خَلَقَ اللَّهُ ذَلِكَ إِلا بِالْحَقِّ يُفَصِّلُ الآيَاتِ لِقَوْمٍ يَعْلَمُون ﴾

صدق الله العلي العظيم

" سورة يونس ،آية (٥) "

Declaration

I hereby declare that this dissertation entitled "**Diabetic Mellitus Detection Based on DNA Data and Machine Learning Techniques**", submitted to the University of Babylon in partial fulfilment of requirements for the degree of Master collage of science for women\department of computer science , has not been submitted as an exercise for a similar degree at any other University. I also certify that this work described here is entirely my own except for experts and summaries whose source is appropriately cited in the references.

Signature:

Name: Lena Abed Al- Raheem Hamza

Date: / / 2023

Dedication

I dedicate this work to my heroes: my mother and father, for their everlasting support and love throughout my entire life. Moreover, I dedicate this work to my husband, for his constant support, understanding, and caring nature.

Lena

Supervisors Certification

we certify that this thesis entitled "**Diabetic Mellitus Detection Based on DNA Data and Machine Learning Techniques**" was done by (*Lena Abed Al-Raheem Hamza*) under our supervision.

Signature:

Name: ***Prof. Dr. Hussein Attia Lafta***

Date: / / 2023

Address: University of Babylon/College of Information Technology

Signature:

Name: **Assistant *Prof. Dr.Sura Zaki AL-Rashid***

Date: / / 2023

Address: University of Babylon/College of Information Technology

The Head of the Department Certification

*In view of the available recommendations, I forward the dissertation entitled "***Diabetic Mellitus Detection Based on DNA Data and Machine Learning Techniques***" for examining committee.*

Signature:

Name: **Assistant Prof. Dr. Saif M. Al-Alak**

Date: / / 2023

Address: University of Babylon/College of Science for Women

Certification of the Examination Committee

We are the members of the examination committee ,certify that we have read this thesis entitled (Diabetic Mellitus Detection Based on DNA Data and Machine Learning Techniques) and after the examining the master student (Lena Abed Al-Raheem Hamza) in its contents in 23/10/2023 and that in our opinion it is adequate as a thesis for the degree of Master in Science\ Computer Science with degree (Excellent).

Committee Chairman
Signature:
Name:**Dr. Dheyaa Shaheed Al-Azzawi**
Scientific Order: Professor. Dr.
Address: University of Wasit\ College of Computer Science and Information Technology
Date: / /2023

Committee Member
Signature:
Name: **Dr. Sahar Adill Kadum**
Scientific Order: Professor Dr.
Address: University of Babylon\College of Science for Women
Date: / /2023

Committee Member
Signature:
Name: **Dr. Ali Yakoob Al-Sultan**
Scientific Order: Assistant Prof. Dr.
Address: University of Babylon\College of Science for Women
Date: / /2023

Committee Member (Supervisor)
Signature:
Name: **Dr. Hussein Attia Lafta**
Scientific Order: Professor Dr.
Address: University of Babylon\College of Science for Women
Date: / /2023

Committee Member (Supervisor)
Signature:
Name: **Dr. Sura Zaki AL-Rashid**
Scientific Order: Assistant Prof. Dr.
Address: University of Babylon\College Information Technology
Date: / /2023

Date of Examination: / /2023

Deanship Authentication of college of Science for Women

Approved for the College Committee of graduate studies.

Signature

Name: **Abeer F. Murad**,

Scientific Order: **Professor. Dr.**

Address: **Dean of College of Science for Women**

Date: / /2023

Acknowledgments

All thanks and praise to Allah, the Lord of the world, who gave me courage and enabled me to achieve this work.

My thanks and gratitude go to my supervisors ***Prof. Dr. Hussein Attia Lafta*** and ***Asst. Prof. Dr.Sura Zaki AL-Rashid*** for the support and guidance they have given me and the effort and time to complete this research.

Thanks, and gratitude to all my professors and all the staff of the Department of Computer Sciences \ College of Sciences for Women \University of Babylon for their help.

All Thanks and Gratitude to all my family and friends for their support and encouragement.

Lena Abed Al- **Raheem...** ***(2023)***

Abstract

The Diabetes Mellitus (DM) is a chronic metabolic disorder characterized by high blood sugar levels. It occurs when the body either does not produce enough insulin or fails to effectively use the insulin it produces. Diabetes can be classified into three main types: Type 1, Type 2, and gestational diabetes. If left uncontrolled, diabetes can lead to serious health complications and become dangerous. Prolonged high blood sugar levels can damage various organs and systems, including the heart, blood vessels, kidneys, eyes, and nerves. Diabetes increases the risk of cardiovascular diseases, stroke, kidney disease, vision loss, and lower limb amputations.

DNA analysis, also known as DNA testing or genetic testing, is a scientific technique used to examine and analyse an individual's DNA (deoxyribonucleic acid). It involves the study of specific DNA sequences, genes, or chromosomes to gain insights into various aspects of human genetics and identify potential genetic variations or mutations. By unlocking the genetic information encoded in our DNA, DNA analysis has provided a deeper understanding of our genetic makeup and has contributed significantly to advancements in personalized medicine and genetic research.

In this thesis, DNA analysis is used to classify and predict the DM in patients. Resulting in early detection of the DM disease and preventing it complications.

The dataset is containing the DNA of 14571 people. The data is pre-processed to eliminate the missing data. And apply k-mer technique with various sizes. After that random over-sampling for the imbalance in the dataset. Apply ordinal encoding, and finally Min-Max transformation.

After that three Feature selection techniques are applied separately. Which are Mutual Information, ANOVA, and Univariate Linear Regression.

Finally, Machine Learning and Deep Learning models are developed to classify the condition of DM based on the DNA sequence. The Machine Learning models are Random Forest, Support Vector Machine, Decision Tree, and Gaussian Naïve Bayes. The Deep Learning Model is a combination of CNN-LSTM.

The developed system is built and tested on PC. The processor is Intel® Core™ i7-6600U CPU, with 2.60 GHz. The installed RAM is 8 GB. The best results obtained using the Machine Learning approach is by using Random Forest, with 90% accuracy. And best result obtained using the Deep Learning approach is by using CNN-LSTM model, with 100% accuracy, precision, recall, and F1-score.

Table of Contents

Chapter One: Introduction

Chapter Two: Theoretical Background

Chapter Five: Conclusions and Future Works

List of Figures

List of Tables

List of Algorithms

List of Equations

List of Abbreviations

Abbreviations	Meaning
1DCNN	1 Dimensional Convolutional Neural Network
AdaBoost	Adaptive Boosting
AdaGrad	Adaptive Gradient Algorithm
Adam	Adaptive Moment Estimation
AI	Artificial Intelligence
ANNs	Artificial Neural Networks
ANOVA	Analysis Of Variance
AUC	Area UnderThe Curve
BILSTM	Bidirectional Long Short Term Memory
CART	Classification And Regression Trees
CffDNA	Cell-free fetal DNA
CNN	Convolutional Neural Network
dbGaP	Database Of Genotypes And Phenotypes
DEGs	Differentially Expressed Genes
DL	Deep Learning
DM	Diabetes Mellitus
DNA	Deoxyribonucleic Acid
DS	Decision Stump
DT	Decision Tree
EM	Expectation-Maximization
GANs	Generative Adversarial Networks
GRU	Gated Recurrent Unit
INSR	Insulin Receptor
K*	K-Star
K-mer	Free Alignment Method
k-NN	K-Nearest Neighbour
LSTM	Long-Short Term-Memory
MAE	Mean Absolute Error
MAR	Missing At Random
MCAR	Missing Completely At Random
MI	Mutual Information
ML	Machine Learning
MLP	Multi-Layer Perceptron
MNAR	Missing Not At Random
MSE	Mean Squared Error

MTB	Mycobacterium Tuberculosis
NB	Naïve Bayesian
NLP	Natural Language Processing
NNs	Naïve Nets
OS	Operating System
PC	Personal Computer
PCA	Principal Component Analysis
PIDD	Pima Indian Diabetes Dataset
PNN	Probabilistic Neural Network
PPI	Protein-Protein Interaction
RAM	Random Access Memory
RBF	Radial Basis Function
RF	Random Forest
RFE	Recursive Feature Elimination
RMSProp	Root Mean Square Propagation
RNN	Recurrent Neural Network
RNN	Recurrent Neural Network
SGD	Stochastic Gradient Descent
SMOTE	Synthetic Minority Oversampling Technique
SVM	Support Vector Machines
T1D	Type 1 Diabetes
T2D	Type 2 Diabetes
TB	Effective Tuberculosis
t-SNE	t-distributed Stochastic Neighbour Embedding
VC	Vapnik-Chervonenkis
XGBoost	Extreme Gradient Boosting

Chapter One

General Introduction

1.1 Introduction

The Diabetes mellitus (DM) is a chronic disease that is considered to be life threatening. It can affect any part of the body over time, resulting in serious complications such as nephropathy, neuropathy, and retinopathy. Diabetes mellitus, or diabetes for short, occurs either when the pancreas does not produce enough insulin or when the body cannot effectively use the insulin it produces. Diabetes has two main types called type 1 and type 2. In type 1 diabetes (also known as insulin-dependent or childhood-onset), there is insulin production deficiency in the body, which requires daily administration of insulin, whereas in type 2 diabetes (known formally as non-insulin-dependent or adult-onset), the body cannot use insulin effectively[1].

Deoxyribonucleic acid (DNA) is a biological macromolecule. Its main function is information storage. At present, the advancement of sequencing technology had caused DNA sequence data to grow at an explosive rate, which has also pushed the study of DNA sequences in the wave of big data. Moreover, machine learning is a powerful technique for analysing large-scale data and learns spontaneously to gain knowledge. It has been widely used in DNA sequence data analysis and obtained many research achievements. The construction of DNA is based on nucleotide units. Each nucleotide unit consists of three sub-units: a nitrogenous base, a five-carbon sugar, and at least one phosphate group. There are four types of nitrogenous bases in DNA; Adenine (A), Guanine (G), Cytosine (C), and Thymine (T). The sequence of these nitrogenous bases encodes biological information, and variations in these nucleotide sequences lead to biological diversity among human beings and every living organism [2].

The mechanism by which proteins are created by their related genes is a two-step process; transcription and translation, as displayed in Figure 1.1. In the

transcription step, a particular segment of DNA is copied into a temporary mRNA molecule. Both DNA and RNA are nucleic acids, which utilize base pairs of nucleotides as a corresponding language. In the translation step, the resulting RNA, a single-stranded copy of the gene, is translated into a protein molecule. These two steps are called gene expression. Insulin is a protein produced by pancreatic β-cells to regulate glucose levels in the blood. In the body's normal state, whenever the blood glucose levels begin to rise (e.g., during the digestion process), β-cells rapidly react by emitting some of their stored insulin and increasing the creation of the hormone simultaneously. But ruefully, sometimes the β-cells become unable to produce a sufficient amount of insulin needed to control the blood glucose levels, as in diabetes type-2[3]. Revealing these disrupted genes that results in diabetes will be the focus of this study.

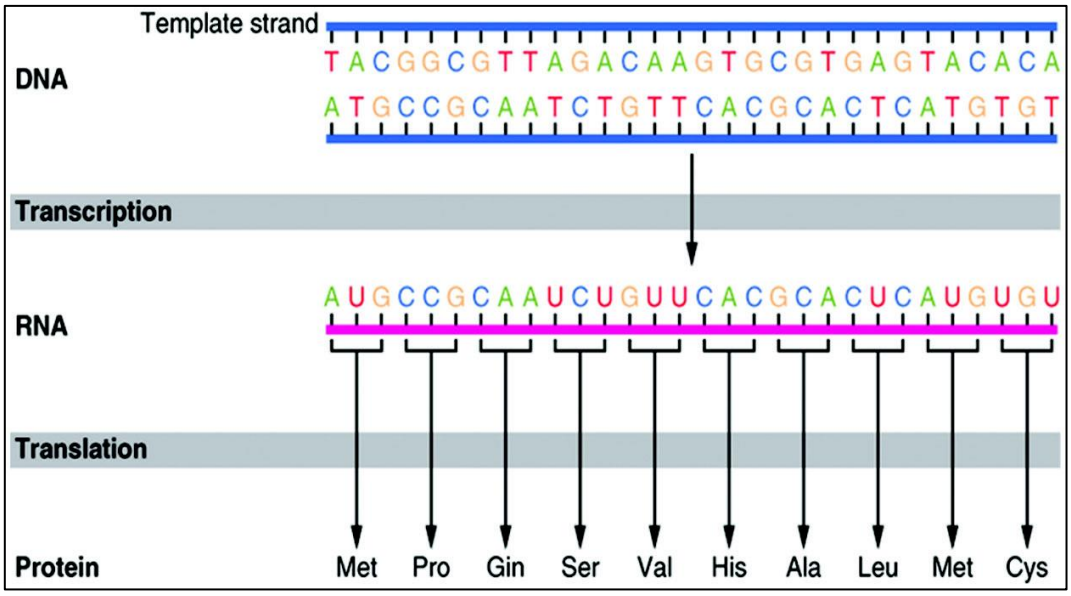

Figure 1.1: DNA transcription and translation[4]

The fields of DNA analysis and gene expression are rapidly developed due to the massive improvements in Machine Learning techniques. And this development led to the emerge of new field of bioinformatics. In this field, advanced Machine Learning techniques (i.e. Deep Learning) are utilized to

explore and analysis the massive amount of genetic related data. Since these data are hard to analysis and explore following the traditional approaches.

1.2 Thesis Motivations

As explained in the previous section, diabetes mellitus is a chronic metabolic disorder that affects millions of people worldwide, imposing a significant burden on both individuals and healthcare systems. In this context, the utilization of DNA sequence data emerges as a powerful tool to unlock the underlying genetic mechanisms, ultimately leading to improved diabetes classification and personalized treatment strategies.

Classifying diabetes using DNA sequences is a compelling and transformative approach that holds immense promise in understanding the genetic intricacies of this complex disease. As we unravel the genetic landscape, we inch closer to a future where diabetes classification is more accurate, treatment is highly personalized, and the burden of diabetes on individuals and healthcare systems is alleviated. By harnessing the power of genomics, we embark on a journey of discovery that has the potential to revolutionize diabetes care and pave the way for a healthier world.

1.3 Problem Statement

Diabetes mellitus is a prevalent chronic disease characterized by abnormal insulin production or utilization, resulting in elevated blood glucose levels. Despite advancements in diabetes management, predicting the onset of diabetes remains a challenging task. The project aims to address this challenge by leveraging machine learning techniques for predicting diabetes based on DNA analysis.The successful implementation of machine learning in predicting diabetes through DNA analysis will contribute to advancing the field of predictive medicine. It has the potential to revolutionize early intervention

strategies, facilitate personalized treatment plans, and ultimately improve the quality of life for individuals at risk of developing diabetes.

1.4 Thesis Questions

This thesis is intended to answer the following questions:

1. What is the performance of different Machine Learning models in predicting DM disease by using DNA sequence classification. And Which give the best performance of Machine Learning and Deep Learning models.
2. What is the best feature selection technique to eliminate the unnecessary data from DNA sequence.

1.5 Aim and Objectives of Thesis

The aim of this study is to analysis the DNA sequences in healthy and DM patients, to explore the difference between them.

The objectives of this aim is as follows:

- Collect DNA sequence data for both DM patients and healthy people.
- Pre-process and encode the data to be suitable for the Machine Learning models.
- Feature selection to determine the relevant and necessary features of the DNA sequence.
- Build and train Machine and Deep Learning models on the selected data.

1.7 Challenges

The challenges that occur requiring hard efforts to overcomes them could be summarized in the following:

- Although DNA sequence datasets are available online for different purposes. But there are small number of high quality dataset containing DNA

sequences for DM patients and healthy people to train and test intelligent systems to classify these two classes of people based on DNA sequence.

- Since the datasets of biological nature, and especially DNA datasets can vary a lot. One general approach to solve a similar problem will not produce the best results all the times.
- Curse of dimensionality is a big issue facing the problems related to DNA data. Since there are massive number of genes as features in the data. Only a few of these features are relevant to the importance of the discovered problem. This requiring to carefully selecting the best fitting features.

1.8 Related Works

Alehegn, M., Joshi, R. and Mulay, P, 2018[5] used tabular dataset with 768 records. The dataset is PIDD (Pima Indian Diabetes Dataset). The goal of the work it topredict the disease of DM from the data by using multiple classification methods combined together (Ensemble method). First, pre-processing step is done by applying fill missing values, remove duplication, and standardization. After that, three classifiers are trained and tested. Which are SVM, Naïve Nets (NNs), and Decision Stump (DS). For the prediction phase, not one algorithm is used, but the one with the most accurate result (which is known as Ensemble model Accuracy: 90.36%.

Abdulaimma, B. *et al*, in 2018[6] used the data set (Gene Expression) Database of Genotypes and Phenotypes (dbGaP). In which have total of 2813 diabetics people genes are collected, and 3228 as control group. Use the genetic and clinical and sociodemographic risk factor to predict and classify type 2 diabetes. First, data reprocessing is done. This included removing the duplicate, filling missing data, remove outliers, and select only white ancestry people to remove unexpected variables. Second, Allelic and Logistic association are used to relate SNPs (gene location) to type 2 diabetes. Third, group of machine

learning algorithms are trained and tested on 80%-20% training testing split. The algorithms are Stochastic Gradient Boosting (GBM), SVM, Random Forest, K-Nearest Neighbour (k-NN), Classification and Regression Trees (CART), and Multi-Layer Perceptron Neural Network (MLP). Area Under the Curve (AUC): 73.96%, Sensitivity: 68.42%, and specificity: 78.67%.

Zhou, H., Myrzashova, R. and Zheng, R, in 2020[7] used the data set Pima Indians diabetes for diagnosing diabetes.Which have the features: Number of times pregnant, Plasma glucose concentration, Diastolic blood pressure, Triceps skin fold thickness, 2-h serum insulin, Body mass index, Diabetes pedigree function, and Age. The second dataset is Diabetes type. To develop a deep learning model that will be able to predict if a person will have diabetes in the future. Furthermore, it can predict the type of diabetes (type 1 or 2). First, the data is divided into 70% training and 30% testing. After that, deep neural model is built and trained. DropOutis used to control the problem of over fitting. Dataset Diabetes type: 94%. Pima Indians diabetes dataset: 99.4%.

Hatmal, M. M. *et al*, in 2020[8]used the dataset(Gene Expression) Total of 164 people. 82 diabetics and 82 healthy people To show if there is a relationship between FokI polymorphism and T2DM. Data are divided into 80% training and 20% testing. Fully-connected Neural Network is developed and trained to classify T2D people. The FNN structure is 10 nodes for the input layer, 6 for the first hidden layer, 6 for the second hidden layer and 1 for the output layer. ReLU activation functions used for all layers, except for the output layer with sigmoid function. Accuracy: 88%.

Hatmal, M. M. *et al.*, in 2021[9] used the dataset (Genetic Expression) 177diabetics persons with 173 control group get their data genome data the involvement of CD36 gene rs1761667 (G>A) and rs1527483 (C>T) polymorphisms in the pathogenesis of T2DM. First, the researchers uses statistical analysis to show if there is a relationship between the CD36 gene and

T2D, and there no such relationship. After that, they trained multiple machine learning algorithms to classify T2D. The used machine learning algorithms are: Random Forest, eXtreme gradient boosting (XGBoost), k-Nearest Neighbours (kNN), Probabilistic Neural Network (PNN), Naïve Bayesian (NB), Multilayer Perceptron (MLP), Support Vector Machine (SVM), K-star (K*), Gradient-boost, and Adaptive boosting (AdaBoost). MLP: 75% accuracy. K*: 73% accuracy.

Muneeb, M. and Henschel, A, in 2021[10] used the dataset(Gene Expression) OPENSNP dataset. Total of 806 people. 404 people have Blue-Green eyes where 402 people have Brown eyes. Use statistical analysis to relate SNPs with diabetics. Then, the genomes are used in multiple machine learning algorithms to predict diabetes. First, diabetics related SNPs are selected using statistical analysis. After that, the data are trained on different machine learning algorithms. Which are Random Forest,Extreme Gradient boosting, Artificial Neural Network (ANN), Long-Short Term-Memory (LSTM), Gated Recurrent Unit (GRU), Bidirectional LSTM (BILSTM), 1 dimensional Convolutional Neural Network (1DCNN), ensembles of ANN, and ensembles of LSTM. Accuracy 96%.

Gunasekaran, et, al. in 2021[11], their study utilizes various architectures, namely CNN, CNN-LSTM, and CNN-Bidirectional LSTM, for DNA sequence classification. The classification task employs Label and k-mer encoding techniques. Multiple classification metrics are employed to assess the performance of these models. Based on the experimental results, the CNN and CNN-Bidirectional LSTM models, both utilizing k-mer encoding, demonstrate superior accuracy on the testing data, achieving 93.16% and 93.13% respectively.

Jian, Y. *et al*, in 2021[12] used tabular dataset,collected from Rashid Centre for Diabetes and Research in UAE. The dataset contains the records of 884

persons, with 79 variable for each one of them. The goal of the research is to predict and classify 8 complications of DM. First of all, pre-processing step is considered to clean the data from unnecessary records, data imputation, categorical encoding, data balancing, data normalization, and feature selection. After that, group of Machine Learning algorithms are used which are: logistic regression, SVM, decision tree, random forest, AdaBoost, and XGBoost, Highest Accuracy and F1-score reached are: 97.8% and 97.7% respectively.

Karaglani, M. *et al.*, 2022[13] used the data set of type Cell-free fatal DNA (cffDNA),with 96 patients with T2DM, and 71 healthy people (control group), in which the methylation related to T2DM could be found. Auto ML pipeline is used for the analysis. The proposed Auto ML consists of the following steps. First step is data pre-processing. Which includes Mean Imputation, Mode Imputation, Constant removal, Standardization. Second step is feature selection by using LASSO. And the classification is carried out by using multiple algorithms. Which are Random Forests, Support Vector Machines (SVM), Ridge Logistic Regression and Decision Trees. The highest accuracy obtained from this Auto ML model is 0.927.

Li, J. *et al.*, 2022[14] used the data set (Gene Expression) microarray datasets, GSE164416, GSE156993, GSE161355, GSE163980, GSE76895, GSE9006, and GSE78721 from GEO website by using differentially expressed genes (DEGs), the effective biomarkers related to T2DM will be discovered to help early diagnosis. First, Limma package in R language is used to perform Differentially Expressed Genes between the DM patients and healthy persons. R cluster Profiler package is then used to perform Functional Enrichment Analysis. After that, Protein-Protein Interaction (PPI) Networks is built to discover the most important genes (the one with the most connectivity). After that SVM is trained to classify the DM related genes. AUC 1.0.

E. El-Attar, N., M. Moustafa, B. and A. Awad, W,2022[15] they used the dataset (DNA sequence) 300 human insulin sequence are selected from GenBank website To classify the insulin DNA sequence, to predict T2DM. based on gene sequence mutation. First, data is divided into 70%-30% training-testing groups. Then, pre-processing is done by applying one-hot vector encoding to the genes. Resulting in binary representation of the gene. After that, hybrid CNN-LSTM model is proposed. The proposed CNN is based on LeNet-5, and the LSTM is completely proposed by the authors. The data is input in LSTM first, which characterize the complex structure of the insulin gene. After that, CNN will interpret the patters in the insulin gene. The obtained training accuracy is 99%, 97.5%, and 95% for LSTM-CNN, CNN, and LSTM, respectively.

Li, Pan, and Cai in 2022[16], In this study, the researchers examined the expression profiles of over 1600 individual cells, comprising 949 cells from individuals with Type 2 Diabetes (T2D) and 651 cells from healthy controls. They aimed to identify distinctive characteristics and differential expression patterns at the transcriptomics level that could discern these two groups of cells. To analyse the expression profiles, the researchers employed various machine learning algorithms, such as Monte Carlo feature selection, support vector machine, and repeated incremental pruning to produce error reduction (RIPPER). Through these methods, they identified several T2D-associated genes (MTND4P24, MTND2P28, and LOC100128906).

The related works for this thesis is summarized in Table 1.1.

Table 1.1: related works comparison

Reference	Problem	Aim	DM	Method	Dataset	Result
[13](1**) Liquid Biopsy in Type 2 Diabetes Mellitus Management: Building Specific Biosignatures via Machine Learning**	There is a growing demand for less invasive biomarkers that can be used to diagnose type 2 diabetes (T2DM) at an early stage before symptoms appear and to monitor the loss of β-pancreatic cells.	From cffDNA data, we could find the methylation related to T2DM	T2D	Random Forests, SVM, Ridge Logistic Regression and Decision Trees	**DNA sequence.** 96 patients with T2DM, and 71 healthy person (control group)	AUC 0.927
[14](2**)Identification of Type 2 Diabetes Based on a Ten-Gene Biomarker Prediction Model Constructed Using a Support Vector Machine Algorithm'**	Identify reliable biomarkers that can be used for an accurate diagnosis of type 2 diabetes.	By using differentially expressed genes (DEGs), the effective biomarkers related to T2DM will be discovered.	T2D	Clustering, SVM	**Gene Expression.**	AUC 1.0
[15](3**)Deep Learning Model to Detect Diabetes Mellitus Based on DNA Sequence'**	DM can occur due to any alterations in the sequence of the insulin gene.	To classify the insulin DNA sequence, to predict T2DM.	DM	CNN-LSTM	**DNA sequence.** 300 human insulin sequence.	Accuracy is 99%, 97.5%, and 95% for LSTM-CNN, CNN, and LSTM, respectively.

[5](4)**Analysis and Prediction of Diabetes Mellitus using Machine Learning Algorithm'**	Utilize diverse data mining techniques at various times to cluster and forecast symptoms within medical data.	Predict the disease of DM from the data by using Ensemble method.	DM	SVM, Naïve Nets, and Decision Stump	Tabular. 768 records, data set from PIDD	Accuracy: 90.36%
[12](5)'**A Machine Learning Approach to Predicting Diabetes Complications',**	DM is a persistent condition that carries potential risks to life. As time progresses, it can impact various parts of the body.	Predict and classify 8 complications of DM.	DM	Logistic regression, SVM, decision tree, random forest, AdaBoost, and XGBoost.	**Tabular.** Collected dataset contains the records of 884 persons.	Highest Accuracy and F1-score reached are: 97.8% and 97.7% respectively.
[6](6)'**Improving Type 2 Diabetes Phenotypic Classification by Combining Genetics and Conventional Risk Factors**	Development of T2DM involves a complex interplay factors. This condition is characterized by multiple contributing factors.	Use the genetic and clinical and sociodemographic risk factor to predict and classify type 2 diabetes	T2D	Random Forest	**Gene Expression.** Database of Genotypes and Phenotypes	AUC: 73.96%, Sensitivity: 68.42%, and specificity: 78.67%
[9](7) **investigating the association of CD36 gene polymorphisms (rs1761667 and rs1527483) with T2DM and dyslipidemia: Statistical analysis, machine learning based prediction, and meta-analysis'**	The aim of this study is to investigate the potential correlation between the CD36 gene and diabetes. Additionally, the study seeks to develop machine learning algorithms for the prediction and classification of diabetes.	The involvement of CD36 gene rs1761667 (G>A) and rs1527483 (C>T) polymorphisms in the pathogenesis of T2DM	DM	Multilayer Perceptron(MLP), K-star (K*)	**Genetic Expression.** 177diabetics persons with 173 control group get their data genome data.	MLP: 75% accuracy. K*: 73% accuracy.
[8](8)**Artificial Neural Networks Model for**	Detect if the is a relationship between some genes and	To show if there is a relationship between FokI polymorphism and	T2D	feed-forward neural network (FNN)	**Gene Expression.** Total of 164	Accuracy: 88%

Predicting Type 2 Diabetes Mellitus Based on VDR Gene FokI Polymorphism, Lipid Profile and Demographic Data'	diabetics. And predict diabetics from genes.	T2DM. And classify T2D based on FokI polymorphism, lipid profile, gender and age.			people. 82 diabetics and 82 healthy people.	
[10](9)**Eye-colour and Type-2 diabetes phenotype prediction from genotype data using deep learning methods'**	Predict diabetics from gene profiles.	Use statistical analysis to relate SNPs with diabetics. Then, the genomes are used in multiple machine learning algorithms to predict diabetes.	T2D	Ensembles of LSTM	**Gene Expression.** OPENSNP dataset.	Accuracy 96%.
[7](10)**Diabetes prediction model based on an enhanced deep neural network'**	Predict the diabetes type accurately before the occurrence of the disease.	To develop a deep learning model to predict DM.	DM	Deep Neural Network	Pima Indians	Pima Indians diabetes dataset: 99.4%
[16](12) **Identification of Type 2 Diabetes Biomarkers From Mixed Single-Cell Sequencing Data With Feature Selection Methods**	Discover the genes associated with T2D	Analyse the expression profiles by using Machine Learning.	T2D	Monte Carlo feature selection, support vector machine, and repeated incremental pruning	RNA sequencing data from the GEO	identified T2D-associated genes (MTND4P24, MTND2P28, and LOC100128906)
[11](13) **Analysis of DNA Sequence Classification Using CNN and Hybrid Models**	Identification and classification of COVID-19 viruses	Develop Deep Learning method to identify COVID-19 by using DNA sequence	Covid-19	CNN-LSTM	DNA sequence. (NCBI)	93.16% and 93.13% accuracy,

1.9 Structure of the Thesis

The remaining part of the thesis is organized as the following:

- **Chapter two**: A theoretical Background.

This chapter will provide a background of DM and DNA sequence importance in the use of disease prediction. Furthermore, it explain the theory of techniques used in data pre-processing, feature selection, and Machine and Deep Learning.

- **Chapter three**: The proposed System

Detailed explanation of the proposed system and the used techniques. Starting from the pre-processing steps. Which includes dealing with missing values, k-mer encoding, random oversampling, and Min-Max transformation. Furthermore, the used feature selection approaches. which includes Mutual Information, ANOVA, and Univariate linear regression. Finally, the techniques of Machine and Deep Learning, including Support Vector Machine, Random Forrest, Decision Tree, Gaussian Naïve Bayes, and CNN-LSTM model.

- **Chapter four**: Experimental Results

The experimental results of the proposed system will be presented in this chapter.

- **Chapter five**: Conclusions and Future Works

This chapter shows the major conclusions gotten from the results of the proposed system in addition to the suggestions of some future works.

Chapter Two

Theoretical Background

2.1 Introduction

In this chapter the theoretical background required to understand this thesiswill be explained properly.

These includes the Diabetes mellitus disease, the important of DNA sequence in disease prediction, the pre-processing importance and techniques, the feature selection importance and techniques. And finally, the model building and evaluation methods.

2.2. Diabetes Mellitus Disease

The Diabetes mellitus, commonly referred to as diabetes, is a chronic metabolic disorder characterized by high blood glucose levels. It affects millions of people worldwide and can lead to serious complications if not properly managed. This section provides an overview of diabetes, including its causes, types, symptoms, and management strategies, drawing on current research and medical knowledge[17][18].

Diabetes arises from a combination of genetic and environmental factors. In type 1 diabetes, an autoimmune response destroys the insulin-producing cells in the pancreas, leading to a deficiency of insulin, a hormone responsible for regulating blood sugar levels. Type 1 diabetes typically develops in childhood or adolescence and requires lifelong insulin therapy[17][18].

Type 2 diabetes, the most common form, occurs when the body becomes resistant to insulin or fails to produce enough insulin to maintain normal blood sugar levels. This type is often associated with lifestyle factors such as obesity, sedentary behavior, poor diet, and genetic predisposition. Type 2 diabetes can develop at any age and is increasingly prevalent among younger individuals[17][18].

The symptoms of diabetes vary depending on the type and severity but may include increased thirst and urination, unexplained weight loss, excessive

hunger, fatigue, blurred vision, slow wound healing, and recurrent infections. However, some individuals with type 2 diabetes may be asymptomatic, making early detection challenging[17][18].

2.2.1. DNA Sequence in Disease Prediction

Advancements in genetic research and technology have opened up new avenues for disease prediction and personalized medicine. One of the most powerful tools in this field is the analysis of DNA sequences. By examining an individual's genetic makeup, scientists can gain valuable insights into their predisposition to certain diseases and develop targeted prevention and treatment strategies. The use of DNA sequences in disease prediction has the potential to revolutionize healthcare by enabling early intervention and reducing the burden of illness[19]–[21].

DNA, or deoxyribonucleic acid, is the genetic material that carries the instructions for the development, functioning, and reproduction of all living organisms. The sequence of nucleotide bases within DNA determines the unique characteristics of each individual, including their susceptibility to certain diseases. Advances in DNA sequencing technology have made it possible to read and interpret an individual's entire genome, allowing researchers to identify specific genetic variations associated with various diseases [19]–[21].

One of the key applications of DNA sequencing in disease prediction is the identification of genetic markers or variants that are associated with increased risk for specific conditions. Scientists have discovered numerous genetic variations that are linked to various diseases, including heart disease, cancer, diabetes, and Alzheimer's disease, among others. By analyzing an individual's DNA sequence, these genetic markers can be identified, and the person's risk for developing a particular disease can be assessed [19]–[21].

2.4 Pre-Processing

Because of the increasing amount of heterogeneous data, data sets often have missing data and inconsistent data. Low data quality will have a serious negative impact on the pattern recognition process. The relevant data pre-processing techniques are handling missing data, Free Alignment Method, Oversampling, Ordinal Encoding, and transformation. Below these techniques that are used in the pre-processing phase in this thesis are explained deeply.

2.4.1 Missing Data Handling

Missing data is a common challenge in machine learning that occurs when one or more values are absent or unavailable in a dataset. The presence of missing data can significantly impact the accuracy and reliability of machine learning models, as they rely on complete and consistent data for effective learning and prediction. Dealing with missing data is a crucial step in the data pre-processing phase to ensure robust and meaningful analysis. There are various reasons for missing data, including human error during data collection, equipment malfunction, survey non-response, or simply the absence of certain measurements. To handle missing data, several techniques can be employed:

1. Deletion: In cases where the missing data is minimal, one approach is to remove the instances or variables with missing values. This approach can be effective when the missingness is missing completely at random (MCAR), but it can lead to biased results if the missingness is missing at random (MAR) ormissing not at random(MNAR)[22].
2. Imputation: Imputation involves estimating and replacing the missing values with reasonable values. Common imputation methods include mean imputation (replacing missing values with the mean of the available values), regression imputation (predicting missing values based on the relationship with other variables), and multiple imputation (generating multiple plausible values for missing data using statistical models)[22].

2.4.2 Free Alignment Method (K-mer):

K-mer encoding is a technique for generating substrings of a specific length (K) from a biological sequence. These K-mers consist of nucleotides (A, T, G, and C) and find various applications such as DNA sequence assembly, enhancement of heterologous gene expression, identification of species in metagenomic samples, and the development of attenuated vaccines. Typically, the term "K-mer" encompasses all possible subsequence's of length K within a sequence. For example, the sequence AGAT would have four monomers (A, G, A, and T), three 2-mers (AG, GA, AT), two 3-mers (AGA and GAT), and one 4-mer (AGAT)[23]. Figure 2.1 shows how a DNA sequence with 6 k-mer will be performed.

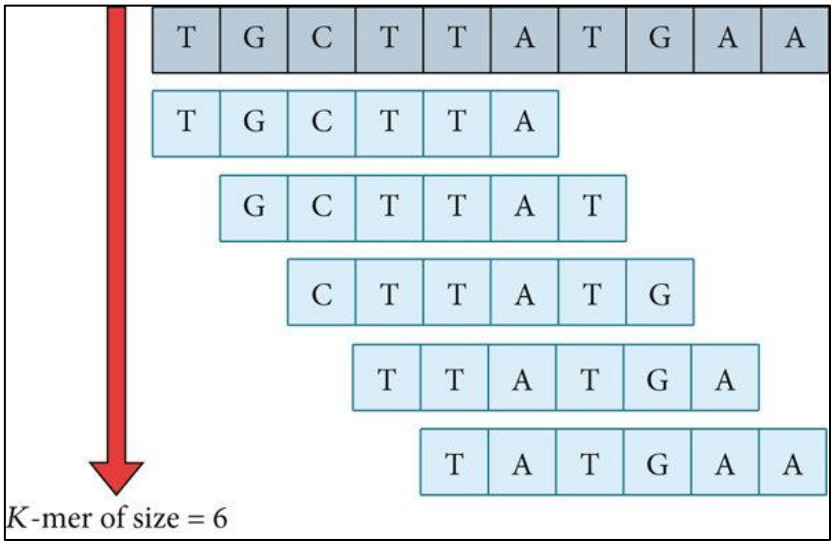

Figure 2.1: K-mer example[24]

K-mer counting tools can be classified according to their approach and the data structures they employ [25]. Equation 2.1 shows how to calculate the number of generated features (i.e. words).

$$n = L - K + 1 \qquad ..(2.1)$$

Where, n is the total number of words, L is the sequence length, K is k-mer size.

Algorithm 2.1 shows the steps used in k-mer approach[26].

Algorithm 2.1: k-mer algorithm

Inputs: – Sequence: A string representing the input sequence – k: An integer representing the desired k-mer length
Output: – kMers: A list of k-mers generated from the input sequence
Steps: 1. Initialize an empty list kMers to store the generated k-mers. 2. Check if the length of the input sequence is less than or equal to k. If true, return an empty list (since the input sequence is too short to generate any k-mers). 3. Iterate over the input sequence from index 0 to length of the sequence minus k: a. Extract the substring of length k starting from the current index. b. Append the extracted k-mer to the kMers list. 4. Return the kMers list.

2.4.3 Random Oversampling for Imbalanced Data:

Random oversampling is a common technique used in data analysis and machine learning tasks to create representative samples from a larger dataset. It involves randomly selecting observations from the dataset to form a subset that can be analysed or used for model training. While random sampling is a straightforward approach, it may encounter challenges when dealing with imbalanced data, where the distribution of the target variable is skewed towards one class.

Imbalanced data refers to a situation in which the classes of interest in a dataset have significantly different frequencies. For example, in a medical dataset, the majority of patients may belong to the "healthy" class, while a minority may belong to the "disease" class. Imbalanced data can also arise in fraud detection, anomaly detection, or other domains where rare events are of interest[27].

When applying random sampling to imbalanced data, a potential issue is that the resulting sample may not adequately represent the minority class. The random selection process is unbiased, meaning it does not consider the class distribution and can easily end up with a sample that is heavily skewed towards the majority class.

To address this issue, several techniques can be employed to ensure a more balanced representation of both classes in the sample. Some of these techniques include:

1. Random Oversampling: This technique involves randomly duplicating observations from the minority class to increase its representation in the sample[28].
2. Random Under-sampling: This technique involves randomly removing observations from the majority class to reduce its dominance in the sample [28].

Figure 2.2 shows a comparison between two common sampling approaches.

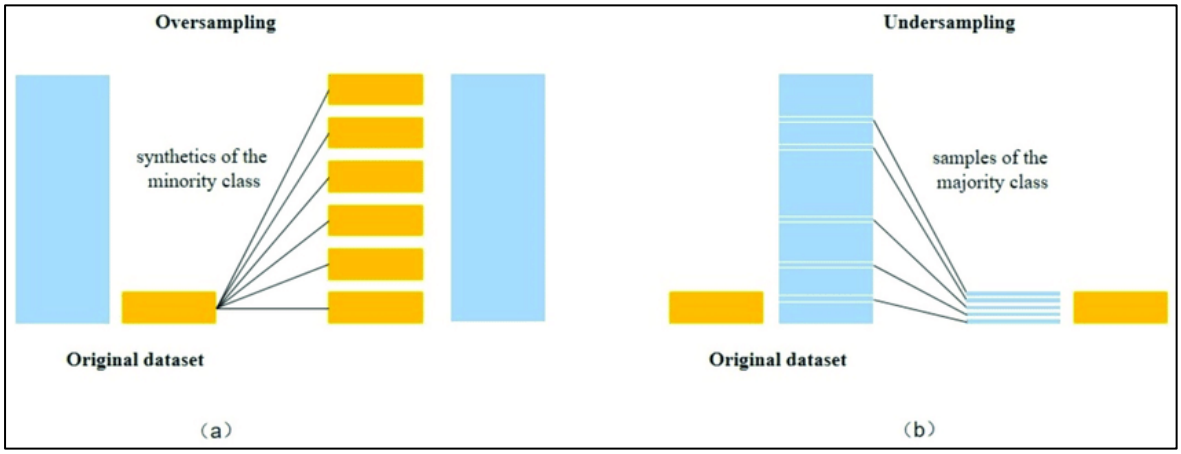

Figure 2.2: comparison between oversampling and under sampling [29]

Algorithm 2.2 shows the steps to implement random over sampling for imbalanced data[28].

Algorithm 2.2: Random over-Sampling for imbalanced data algorithm
Inputs: – Dataset with imbalanced classes (class distribution is not uniform) – Target class to sample (the minority class)
Output: – Sampled dataset with balanced classes (approximately equal distribution of classes)

Steps:

1. Calculate the class distribution of the target class in the dataset.
2. Determine the number of instances needed to balance the dataset by selecting the maximum class count as the target count for each class.
3. Initialize an empty sampled dataset.
4. For each class in the dataset:
    a. If the class is the target class:
        i. Randomly sample instances from the class to match the target count determined in step 2.
        ii. Add these instances to the sampled dataset.
    b. If the class is not the target class:
        i. Add all instances from the class to the sampled dataset.
5. Shuffle the sampled dataset to ensure randomness.
6. Return the sampled dataset with balanced classes.

2.4.4 Ordinal Encoding:

Ordinal encoding is a popular technique used in data pre-processing and feature engineering tasks to encode categorical variables. It assigns a numerical value to each category, preserving the ordinal relationship between them. This encoding method is commonly employed when dealing with machine learning algorithms that require numeric inputs[30].

Ordinal encoding involves assigning a unique integer value to each category based on its position in the predefined order. The encoding process typically follows these steps[31]:

1. Identify the unique categories within the categorical variable.
2. Define an order or hierarchy for the categories based on their inherent ranking or meaning. This order can be predefined based on domain knowledge or determined through statistical analysis.
3. Assign numerical values to each category, starting from 1 or 0, based on their position in the defined order. The category with the lowest rank is usually assigned the smallest value, and subsequent categories are assigned higher values accordingly.

Algorithm 2.3 shows the steps used in ordinal encoder[32].

Algorithm 2.3: Ordinal encoder algorithm

Inputs:

- Dataset: A dataset with categorical features
- Categorical Features: A list of categorical feature names

Output:

- Encoded Dataset: A dataset with categorical features replaced by integer ordinal encoded values

Steps:

1) Initialize an empty dictionary, *ordinal_mapping*, to store the ordinal mapping for each unique category in each categorical feature.
2) For each categorical feature in the list of Categorical Features, perform the following steps:
   - Initialize an empty dictionary, *category_mapping*, to store the mapping of categories to ordinal values for the current feature.
   - Extract the unique categories from the current categorical feature in the Dataset and sort them in ascending order.
   - For each unique category *category* in the current categorical feature, assign an ordinal value starting from 0 and increment by 1 for each subsequent category.
   - Store the mapping of each category to its corresponding ordinal value in the *category_mapping* dictionary.
   - Store the *category_mapping* dictionary in the *ordinal_mapping* dictionary with the current categorical feature name as the key.
3) Create a copy of the original Dataset and initialize it as the Encoded Dataset.
4) For each categorical feature in the list of Categorical Features, perform the following steps:
   - Retrieve the *category_mapping* dictionary for the current feature from the *ordinal_mapping* dictionary.
   - Replace the categorical values in the corresponding column of the Encoded Dataset with their ordinal values using the *category_mapping*dictionary.
5) Return the Encoded Dataset as the result.

2.4.5 Min-Max Normalization:

Min-max transformation or normalization, is a common data pre-processing technique used to rescale the values of a dataset to a specific

range. It is useful when working with variables that have different scales or ranges of values. The purpose of min-max transformation is to bring all the values within a desired range, typically between 0 and 1, although other ranges can also be chosen[33].

The min-max transformation is based on the minimum and maximum values present in the feature. The transformation process involves subtracting the minimum value from each data point and then dividing it by the difference between the maximum and minimum values (as shown in equation 2.2 below). This results in a transformed value that falls within the desired range[33].

Equation 2.2 shows the mathematical formula for Min-Max transformation.

$$\hat{x} = \frac{x - \min(X)}{\max(X) - \min(X)} \qquad (2.2)$$

Where: x is an individual data point in the dataset.

$\hat{x}$ is the transformed value of x.

$\min(X)$ is the minimum value in the feature.

$\max(X)$ is the maximum value in the feature.

2.5 Feature Selection

In machine learning, feature selection plays a vital role as it entails the selection of a subset of pertinent features from a larger pool of available features. The objective is to identify the most informative and distinguishing features that significantly contribute to the predictive capability of a machine learning model, all while minimizing complexity, computation time, and the risk of overfitting[34].

In many real-world applications, datasets often contain a large number of features, and not all of them are necessarily useful for making accurate

predictions. Feature selection techniques aim to address these issues by identifying and retaining only the most important features, improving model interpretability and generalization[34][35].

Feature selection methods can be broadly categorized into three main types: filter methods, wrapper methods, and embedded methods[35].

1. **Filter methods:**Filter methods evaluate the importance of features by analyzing their statistical characteristics, such as correlation, mutual information, or significance tests, without considering any particular learning algorithm.Filter methods encompass techniques like chi-square test, information gain, and correlation-based feature selection.
2. **Wrapper methods:**Wrapper methods assess the effectiveness of a feature subset by directly training and evaluating a machine learning model using various subsets of features. These methods employ a particular learning algorithm and search strategy to identify the optimal feature subset that maximizes the model's performance. Popular examples of wrapper methods include Recursive Feature Elimination (RFE) and Genetic Algorithms.
3. **Embedded methods:**Embedded methods integrate feature selection into the model training process. These methods incorporate feature selection within the learning algorithm itself, leveraging the inherent feature selection capabilities of specific models. For instance, decision tree-based models such as Random Forests and Gradient Boosting.

2.5.1 Mutual Information

Mutual information (MI) is a statistical measure widely used in feature selection to quantify the dependence or information shared between two random variables. Mutual information measures the amount of information that knowing the value of one variable provides about the other variable[36].

In the context of feature selection, mutual information can be used to assess the relationship between each feature and the target variable. The higher the mutual information between a feature and the target, the more information the feature provides for predicting the target variable. By calculating the mutual information for each feature, we can rank them based on their relevance and select the most informative ones[36].

Mathematically, mutual information between two random variables *X* and *Y* is defined as in equation 2.3:

$$I(X;Y) = \sum\sum p(x, y) \log(p(x, y) / (p(x) * p(y))) \quad (2.3)[36]$$

where $p(x, y)$ is the joint probability distribution of *X* and *Y*, and $p(x)$ and $p(y)$ are the marginal probability distributions of *X* and *Y*, respectively.

In practice, estimating mutual information from finite data requires making assumptions about the underlying probability distributions. One common approach is to use the empirical probability distributions based on the observed data. Given a dataset with *N* samples, the empirical joint probability distribution $p(x,y)$ is estimated by counting the occurrences of each combination of values (x,y) in the data and dividing by *N*. Similarly, the empirical marginal probability distributions $p(x)$ and $p(y)$ are estimated by counting the occurrences of each value *x* and *y*, respectively, and dividing by *N*[37].

Algorithm 2.4 shows the steps used in MI[38].

Algorithm 2.4 Mutual Information algorithm
Input: 1. ***dataSet***: The dataset containing both the features and the target variable. 2. ***target***: The target variable for which we want to calculate the mutual information. 3. ***features***: The set of features from which we want to select the relevant ones.
Output: 1. ***mutual_Info_Scores*:** selected features, with their MI score
Steps: 1. Initialize an empty dictionary ***mutual_Info_Scores*** to store the mutual information scores for each feature.

2. For each feature in ***features***, do the following steps:
    3. Calculate the mutual information between the feature and the target variable:
        4. Initialize a dictionary ***feature_Counts*** to store the counts of each unique value of the feature.
        5. Initialize a dictionary ***target_Counts*** to store the counts of each unique value of the target variable.
        6. Initialize a dictionary ***joint_Counts*** to store the joint counts of each unique value pair of the feature and target variable.
        7. Initialize the ***mutual_Information*** variable to 0.
        8. For each instance in ***dataSet***, do the following steps:
            9. Increment the count of the current value of the feature in ***feature_Counts***.
            10. Increment the count of the current value of the target variable in ***target_Counts***.
            11. Increment the count of the current value pair of the feature and target variable in ***joint_Counts***.
        12. For each unique value of the feature, do the following steps:
        13. For each unique value of the target variable, do the following steps:
            14. Calculate the probability of the feature value: ***pFeature = feature_Counts[feature_Value] / total_Instances***, where ***total_Instances*** is the total number of instances in the dataset. And
            15. Calculate the probability of the target value: ***pTarget = target_Counts[target_Value] / total_Instances***.
            16. Calculate the joint probability: ***pJoint = joint_Counts[feature_Value][target_Value] / total_Instances***.
            17. If ***pJoint*** is not equal to 0, calculate the mutual information contribution: ***mutual_Information += pJoint * log2(pJoint / (pFeature * pTarget))***.
        18. Store the calculated ***mutual_Information*** in ***mutual_Info_Scores*** with the feature as the key.
19. Sort ***mutual_Info_Scores*** in descending order based on the mutual information scores.
20. Return the sorted list of features from ***mutual_Info_Scores*** along with their respective mutual information scores.

2.5.2 ANOVA

One-way analysis of variance (ANOVA) is a statistical technique commonly used in feature selection to assess the significance of differences among the means of multiple groups or categories. It helps determine if there

are significant variations in a continuous target variable across different levels of a categorical feature. ANOVA can be employed as a feature selection method to identify relevant features that have a significant impact on the target variable[39].

In the context of feature selection, one-way ANOVA compares the means of the target variable across different levels or categories of a categorical feature. The null hypothesis assumes that the means of all groups are equal, indicating no significant relationship between the feature and the target variable. The alternative hypothesis suggests that at least one of the group means is significantly different, implying a relationship between the feature and the target[40].

2.5.3 Univariate linear regression

Univariate linear regression is a straightforward and widely adopted method for feature selection in machine learning. It encompasses the fitting of a linear regression model between a single independent variable (feature) and a dependent variable (target) to evaluate their relationship. By employing univariate linear regression as a feature selection technique, it becomes possible to identify the most relevant features that exert a significant influence on the target variable[41].

The steps involved in using univariate linear regression for feature selection are as follows:

1. **Data Preparation:** Define the independent variables (i.e. features), and dependent variable (i.e. target).
2. **Calculate significant:** For each feature, calculate the correlation coefficient between the selected feature and the target, as shown in equation 2.4.

$$Y = \beta_0 + \beta_1 * X \qquad (2.4)[42]$$

Where: Y: The target variable (dependent variable).X: The feature (independent variable).β_0: The intercept term (y-intercept).β_1: The slope coefficient (regression coefficient) that represents the change in Y for a one-unit increase in X.

3. **Select the features:** Select the features with correlation coefficient above a predefined threshold, or the top *n* number of features.

Algorithm 2.5 shows the steps in the Univariate Linear Regression approach[43].

Algorithm 2.5: Univariate Linear Regression for Feature Selection
Inputs: – *X*: Feature matrix with shape (*n_samples*, *n_features*) – *y*: Target variable with shape (*n_samples*,) – *threshold*: Threshold value for feature selection (e.g., p-value, correlation coefficient)
Outputs: - *selected_features*: List of selected features based on the univariate linear regression
Steps: 1. Initialize an empty list, *selected_features*. 2. Loop through each feature in *X*: a. Fit a univariate linear regression model with the current feature and *y*. b. Compute the p-value or correlation coefficient for the relationship between the current feature and *y*. c. If the p-value (or absolute correlation coefficient) is above the *threshold*: Add the current feature to the *selected_features* list. 3. Return the *selected_features* list.

2.6 Machine Learning

Machine learning is a subfield of artificial intelligence that enables computers to perform intelligent tasks using intelligent software. It has been applied to a wide range of applications, including computer vision and email filtering, where traditional algorithms may be impractical or insufficient. The core objective of machine learning is to develop algorithms that can automatically learn from datasets. These learned models are then capable of

making predictions based on the knowledge acquired during the training process. The underlying concept is that these algorithms generate a statistical inference from the data they are trained on, utilizing this statistical model to predict future values [44].

2.7 Types of Machine Learning

Machine learning has emerged as a transformative field within the realm of artificial intelligence, empowering computers to learn and make predictions from data without being explicitly programmed. Various types of machine learning algorithms have been developed to cater to different problem domains, data characteristics, and learning objectives [44].

1. **Supervised Learning:** Supervised learning is one of the most common and well-understood types of machine learning. It involves training a model on labelled data, where each example is associated with a corresponding target or output variable. Supervised learning algorithms include linear regression, decision trees, random forests, support vector machines, and neural networks [45].
2. **Unsupervised Learning:** In contrast to supervised learning, unsupervised learning deals with unlabelled data, where the algorithm must discover patterns, structures, or relationships within the data without explicit guidance[46].
3. **Deep Learning:** Deep learning is a subset of machine learning that focuses on artificial neural networks with multiple layers, also known as deep neural networks. Deep learning algorithms excel in processing complex data such as images, audio, and text, enabling breakthroughs in computer vision, natural language processing, and speech recognition. Convolutional neural networks (CNNs) are commonly used for image analysis, while recurrent neural networks (RNNs) are effective for sequential data, and transformers have revolutionized natural language processing tasks [47].

4. **Ensemble Learning:** Ensemble learning combines multiple individual models, known as base learners, to form a stronger, more robust model. Each base learner contributes to the final prediction, and different ensemble techniques like bagging (e.g., random forests) and boosting (e.g., AdaBoost, Gradient Boosting) are employed to aggregate their outputs effectively. Ensemble methods reduce the risk of overfitting, enhance generalization, and often produce better results than individual models [48].

These are just a few of the many types of machine learning techniques and approaches. Each type has its strengths and weaknesses, and the choice of technique depends on the problem at hand and the available data.

2.8 Concepts in Machine Learning

In order to fully understand the Machine Learning systems, some of the techniques used in these systems should be explained properly.

2.8.1 Optimization

Optimization is a fundamental aspect of machine learning that aims to find the best set of model parameters that minimize or maximize a given objective function. The objective function is often defined based on a certain loss function that quantifies the model's performance on the task at hand, such as classification accuracy or mean squared error. The process of optimization involves adjusting the model's parameters iteratively to reach the optimal values that lead to the best possible performance[49].

There are various optimization algorithms used in machine learning, each with its advantages and disadvantages. Some of the most popular optimization algorithms are as follows[50], [51]:

1. **Gradient Descent:** Gradient descent is a widely used optimization algorithm that updates the model's parameters in the opposite direction of the gradient of the loss function with respect to those parameters.

2. **Momentum:** Momentum is an extension of gradient descent that introduces a velocity term to accelerate convergence and overcome oscillations in the optimization process.
3. **AdaGrad (Adaptive Gradient Algorithm):** AdaGrad adapts the learning rate for each parameter based on the historical gradients. It gives more weight to parameters that have a smaller historical gradient, which can be advantageous for dealing with sparse data or high-dimensional problems.
4. **Adam (Adaptive Moment Estimation):** Adam is an adaptive optimization algorithm that combines the benefits of both AdaGrad and RMSProp. It maintains separate learning rates for each parameter and adapts them over time, leading to more stable and efficient convergence.

2.8.2 Epoch, Iteration, and Batch Size

In the context of machine learning, epoch, iteration, and batch size are important concepts used during the training process of a model. These concepts are closely related and play a significant role in determining how a model learns from the training data[52], [53].

1. **Epoch:** An epoch refers to a single pass of the entire training dataset through the learning algorithm. In other words, during one epoch, the model has seen and learned from each training sample once. After each epoch, the model's parameters are updated based on the gradients computed during the forward and backward passes of the training process.
2. **Iteration:** An iteration, also known as a training step or update step, is a single update of the model's parameters during the training process. In most cases, one iteration corresponds to processing a single batch

of data through the model and updating the model's parameters based on the loss function's gradients.

3. **Batch Size:** The batch size refers to the number of samples from the training dataset that are processed together in a single forward and backward pass during each iteration.

Figure 2.3 illustrate the difference between the epoch, iteration, and batch size.

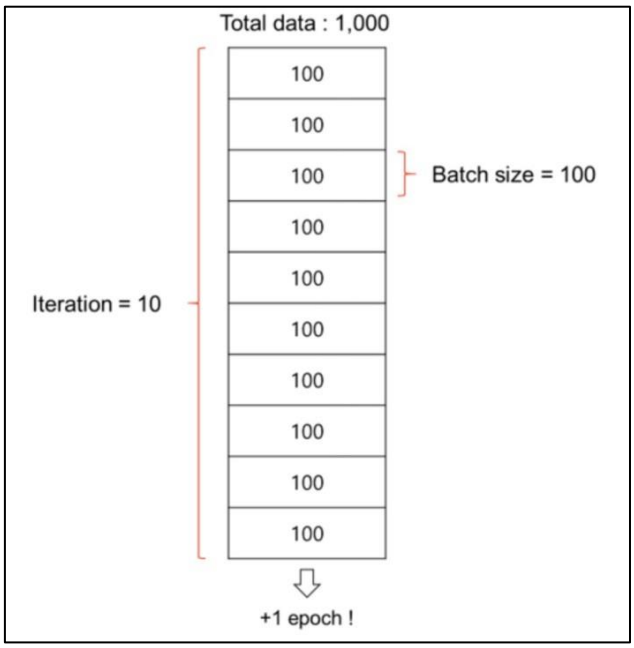

Figure 2.3: Epoch, iteration, and batch size [54]

2.8.3 Activation Function

In machine learning, an activation function is a crucial component of artificial neural networks and other deep learning models. It introduces non-linearity to the model, allowing it to learn complex patterns and make better predictions for a wide range of tasks such as image recognition, natural language processing, and more[55]–[57].

There are several types of activation functions used in deep learning. Some of the most common ones include:

1. **Sigmoid Function:** The sigmoid function is one of the earliest activation functions used in neural networks. It squashes the input values into the range of (0, 1)[55]–[57]. The formula for the sigmoid function is shown in equation 2.5:

$$Sigmoid(x) = \frac{1}{1+e^{-x}} \qquad \ldots(2.5)$$

2. **Rectified Linear Unit (ReLU):** ReLU is one of the most widely used activation functions in deep learning due to its simplicity and ability to mitigate the vanishing gradient problem. It returns the input if it is positive and zero otherwise[55]–[57]. The formula for ReLU is shown in equation 2.6:

$$ReLU(x) = max(0, x) \qquad \ldots(2.6)$$

3. **Softmax:** The softmax function is commonly used in the output layer of a neural network for multi-class classification problems. It converts the raw model outputs (logits) into a probability distribution over multiple classes, ensuring that the sum of the probabilities adds up to 1[55]–[57]. The formula for softmax is shown in equation 2.7:

$$softmax(x_i) = \frac{e^{x_i}}{\sum_{j=1}^{n} e^{x_j}} \qquad \ldots(2.7)$$

where n is the number of classes, and x_i is the raw output for class i.

Figure 2.4 shows a comparison of some of the common Activation Functions.

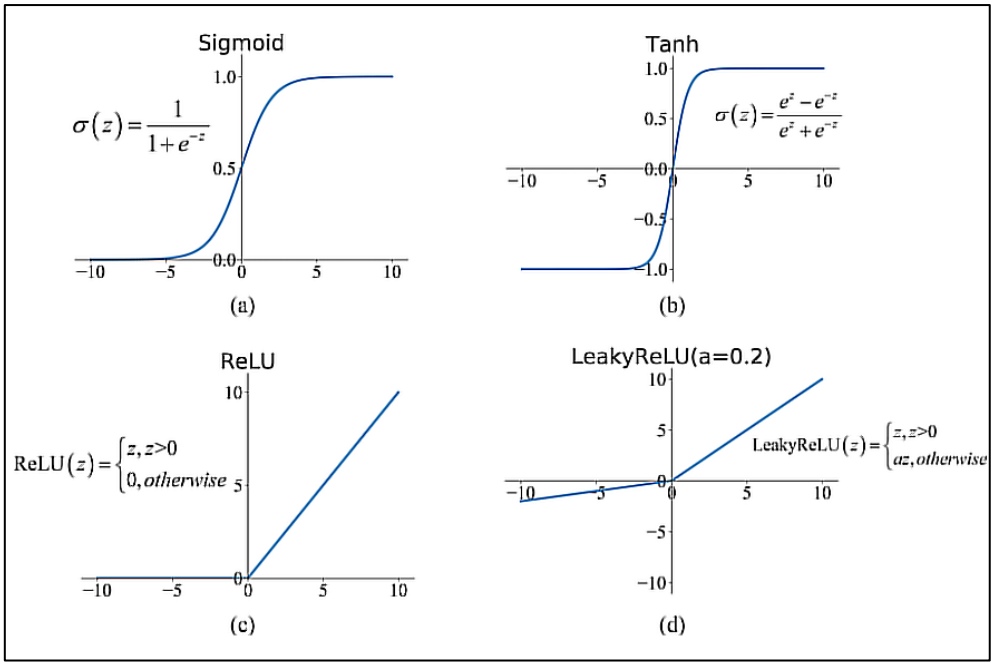

Figure 2.4: comparison of some common Activation Functions [58]

2.8.4 Loss Function

In machine learning, a loss function plays a crucial role in assessing the accuracy of a model's predictions and guiding the learning process during training. It quantifies the discrepancy between the predicted output of a model and the actual target values, helping the model to optimize its parameters to minimize this discrepancy. The ultimate goal is to find the optimal set of parameters that make the model perform well on unseen data[59], [60].

There are various types of loss functions. Some commonly used loss functions include[59], [60]:

1. **Mean Squared Error (MSE):** MSE is one of the most common loss functions used in regression tasks. It calculates the average squared difference between the true target values and the predicted values[59], [60]. Equation 2.8 shows how the MSE is calculated.

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2 \qquad \ldots(2.8)$$

2. **Mean Absolute Error (MAE):** MAE is another loss function used in regression tasks. It calculates the average absolute difference between the true target values and the predicted values [61]. Equation 2.9 shows how MAE is calculated.

$$MAE = \frac{1}{N}\sum_{i=1}^{N} |y_i - \hat{y}_i| \qquad \ldots(2.9)$$

3. **Cross-Entropy Loss (Log Loss):** Cross-entropy loss is commonly used in classification tasks, particularly for problems with multiple classes. It measures the dissimilarity between the predicted probability distribution and the true target labels [62]. Equation 2.10 shows how Cross-Entropy Loss is calculated.

$$h(X, y) = \begin{cases} -\log P(Y = 1|X) & if\ y = 1 \\ -\log P(Y = 0|X) & if\ y = 0 \end{cases} \qquad \ldots(2.10)$$

2.9 Model Building

In order to build and train a predictive model for DM patients based on DNA sequence. The classification techniques will be used.

Classification is an extensively explored task in machine learning, which involves using defined features to determine the class of a predefined target attribute. This process is part of supervised machine learning[63]. In the field of genomics, important concerns revolve around genome classification and sequence annotation.

2.9.1 Support Vector Machine

Support Vector Machine (SVM) is an extensively utilized and powerful supervised machine learning algorithm suitable for both classification and regression tasks. It has gained popularity due to its capacity to handle linear and non-linear data effectively, as well as its ability to tackle high-dimensional datasets. SVMs operate based on the concept of identifying the optimal hyperplane that maximizes the margin, separating various classes of data.[64].

The core idea behind SVM involves transforming the input data into a higher-dimensional feature space using a kernel function (as shown in Figure 2.5). Within this feature space, SVM seeks to locate a hyperplane that maximally separates the data points of different classes. This hyperplane is chosen to maximize the margin, which represents the distance between the hyperplane and the closest data points from each class. These closest data points are known as support vectors, thereby lending the name "Support Vector Machine." [64].

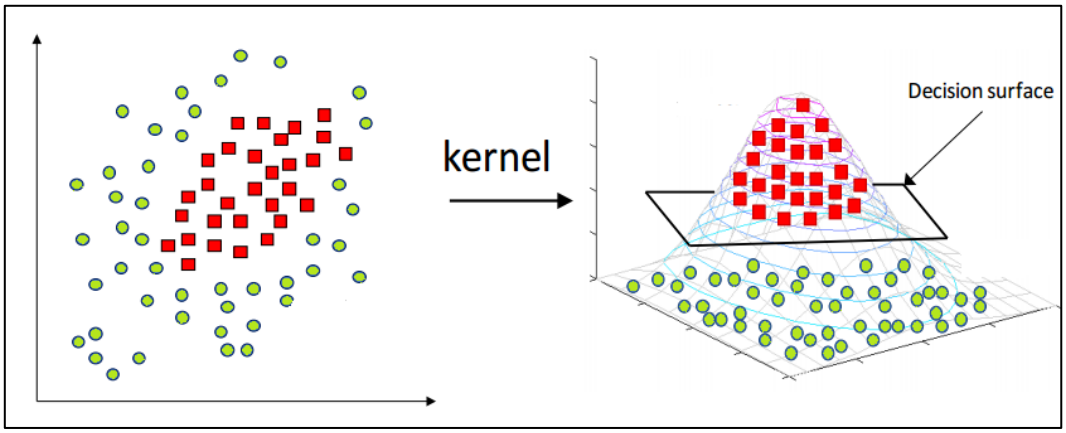

Figure 2.5: Transforming data into a higher-dimensional feature space in SVM

The training process of SVM involves solving an optimization problem to find the hyperplane that maximizes the margin. The resulting hyperplane is then utilized for classifying new, unseen data points[65].

One of the key reasons for SVM's popularity lies in its ability to handle high-dimensional datasets effectively. SVM demonstrates robust performance when the number of features surpasses the number of samples, a common occurrence in real-world applications such as text classification, image recognition, and bioinformatics[65], [66].

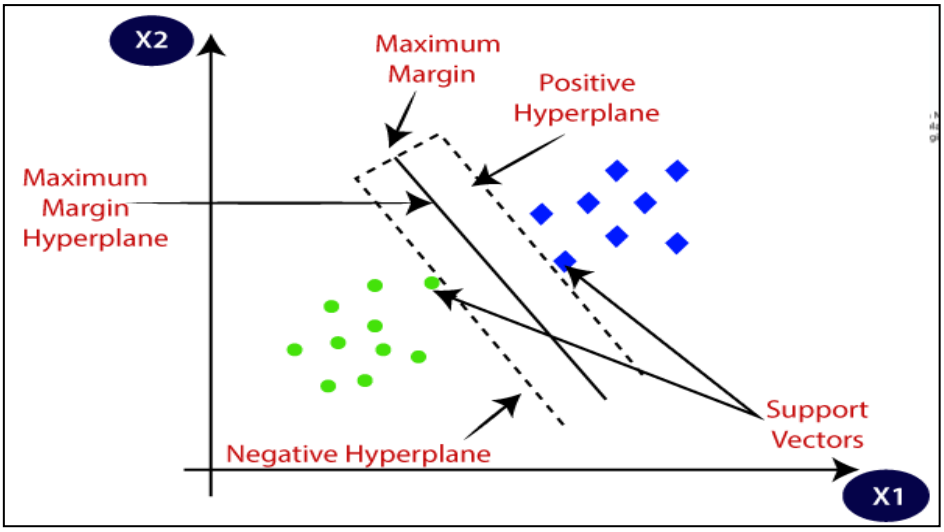

Figure 2.6: Support Vector Machine (SVM) [67]

Algorithm 2.6 shows the SVM steps[68].

Algorithm 2.6: SVM algorithm

Input:

- Training dataset: $X = \{x_1, x_2, ..., xn\}$, where xi represents the input features for the i-th training instance.
- Corresponding labels: $y = \{y_1, y_2, ..., yn\}$, where $yi \in \{-1, +1\}$ represents the class label for the i-th training instance.
- C: a hyperparameter controlling the trade-off between maximizing the margin and minimizing the classification error.

Output:

- The learned SVM model, represented by the support vectors and the separating hyperplane.

Steps:

1. Normalize the training dataset if necessary.
2. Construct the Gram matrix K, where $K(i, j) = \langle xi, xj \rangle$ represents the dot product between the input feature vectors.
3. Initialize the Lagrange multipliers $\alpha i = 0$ for all training instances.
4. Define the optimization problem:
5. Maximize: $W(\alpha) = \Sigma \alpha i - 0.5 \Sigma \Sigma \alpha i \alpha j yiyj K(xi, xj)$
   Subject to: $\Sigma \alpha i yi = 0$ and $0 \leq \alpha i \leq C$ for all i
6. Solve the optimization problem using quadratic programming to find the optimal values of α that maximize W(α).
7. Compute the bias term b using the support vectors:
8. Select a support vector xi with $0 < \alpha i < C$.

9. Compute b as: $b = yi - \Sigma \alpha j\, yj\, K(xi, xj)$ for any support vector xi.
10. Compute the weight vector w as: $w = \Sigma \alpha i\, yi\, xi$.
11. Classify a new test instance x:
12. Compute the predicted label y_pred as: $y\_pred = sign(\Sigma \alpha i\, yi\, K(xi, x) + b)$.
13. The support vectors are the training instances xi with non-zero αi.
14. The separating hyperplane is given by the equation: $\Sigma \alpha i\, yi\, K(xi, x) + b = 0$.

2.9.2 Artificial Neural Network

Artificial Neural Networks (ANNs) are a fundamental concept in the field of artificial intelligence and machine learning. Inspired by the structure and functioning of the human brain, ANNs are computational models composed of interconnected nodes, known as artificial neurons or perceptron's, which work collectively to process and learn from input data[69].

The basic building block of an artificial neural network is the artificial neuron, which mimics the behaviour of a biological neuron. Each artificial neuron receives one or more inputs, applies a mathematical transformation to them, and produces an output. The inputs are typically weighted, representing the strength or importance of each input. The artificial neuron also includes an activation function, which determines the output based on the weighted sum of the inputs[69]. Figure 2.7 shows the work of single neuron.

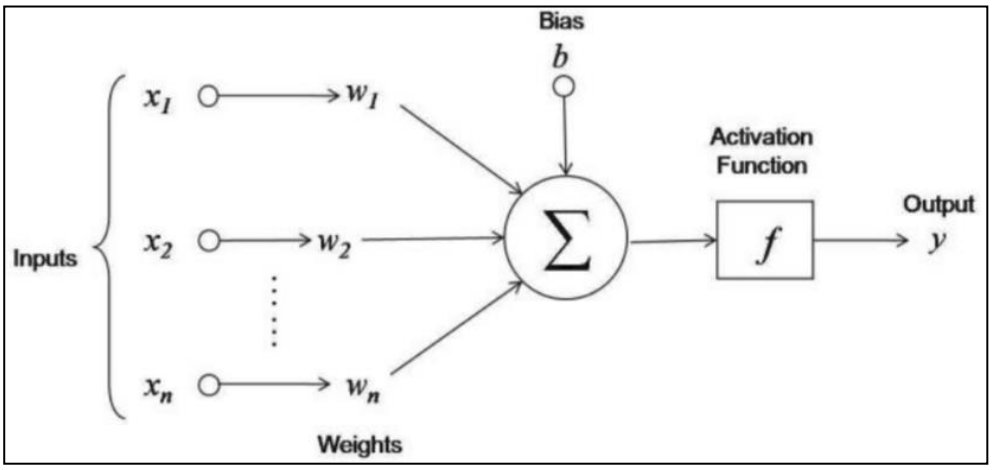

Figure 2.7: Artificial Neuron [70]

The architecture of an artificial neural network consists of multiple layers of interconnected neurons. The input layer receives the initial data, and subsequent layers, known as hidden layers, process and transform the information. Finally, the output layer produces the network's final prediction or output[69].

Training an artificial neural network involves a two-step process: forward propagation and backpropagation. In forward propagation, the input data is passed through the network, layer by layer, and the outputs are calculated. The calculated outputs are then compared to the desired outputs, and the difference, known as the error, is determined[69].

Backpropagation is the process of updating the weights of the artificial neurons to minimize the error. It works by propagating the error backward through the network, adjusting the weights based on the contribution of each neuron to the overall error. This iterative process is repeated until the network's performance reaches a satisfactory level[69].

There are several types of artificial neural networks, each with its own unique characteristics and applications. Feedforward neural networks are the most common type, where information flows in one direction, from the input layer to the output layer. Recurrent neural networks (RNNs) have connections between neurons that form directed cycles, allowing them to process sequential data such as time series or natural language. Convolutional neural networks (CNNs) are specifically designed to process grid-like data, such as images or video frames, using convolutional layers that extract relevant features. These types will be explained in details in the next section[69].

2.9.3 Decision Tree

The Decision Tree (DT) is a well-known machine learning algorithm utilized for classification and regression tasks. It presents a visual representation of a sequence of decisions and their potential outcomes. By learning from

labelled training data, the decision tree algorithm constructs a tree-shaped model where internal nodes correspond to decisions based on specific features, and leaf nodes represent predicted outcomes or class labels[71].

To begin constructing the decision tree, a root node is created, encompassing the entire training dataset. The algorithm then identifies the best feature to split the data, considering criteria like information gain or Gini impurity. Information gain quantifies the decrease in entropy (or increase in information) after the split, while Gini impurity measures the likelihood of misclassifying a randomly chosen element based on the label distribution within the node[72].

After selecting the optimal feature, the dataset is partitioned into subsets according to the possible feature values. This recursive process continues for each subset, generating new internal nodes and branches, until a stopping condition is met. Stopping conditions may involve reaching a maximum depth, having a minimum number of samples in a node, or when further splitting fails to enhance overall performance[72].

The decision tree assigns the predicted outcome or class label at the leaf nodes. In classification, the majority class within the leaf node is chosen as the prediction, while for regression, it may be the mean or median of the target values in that node[72].

Algorithm 2.7 shows the steps used in the building the Decision Tree[73].

Algorithm 2.7: Decision Tree algorithm
Input: – Training dataset: X (input features), y (target variable) – Stopping criteria (e.g., maximum depth, minimum number of samples per leaf) – Splitting criterion (e.g., Gini impurity, information gain)
Output: – Decision Tree model
Steps: 1. Define a node structure for the Decision Tree:

- Attributes:
  - Data subset: X_node (input features), y_node (target variable)
  - Splitting criteria (e.g., feature index, split value)
  - Left child node: left_child
  - Right child node: right_child
  - Class label (for leaf nodes only)

2. Create the root node of the Decision Tree:
   - Assign X_node as the entire training dataset X, and y_node as the corresponding target variable y.
3. Recursively grow the Decision Tree by performing the following steps on each node:
   - If a stopping criterion is met (e.g., maximum depth, minimum samples per leaf), create a leaf node with the majority class label as the predicted class label.
   - Calculate the splitting criterion for each feature in X_node and determine the best feature to split on.
   - Split the data into two subsets based on the best feature and split value:
     - Create left_child node: X_left, y_left
     - Create right_child node: X_right, y_right
   - Recursively apply the above steps to the left_child and right_child nodes.
4. Return the Decision Tree model.

2.9.4 Random Forest

Random Forest (RF) is an ensemble learning algorithm widely utilized in machine learning due to its power and versatility. It excels in both classification and regression tasks, demonstrating remarkable efficacy with high-dimensional datasets. The core principle of Random Forest involves merging the predictions of multiple decision trees to achieve enhanced accuracy and robustness[74].

Random Forest constructs an ensemble of decision trees, with each tree trained on a distinct subset of the training data and features. This introduces randomness to mitigate overfitting and foster diversity among the individual trees within the ensemble. The final prediction of the Random Forest is derived by aggregating the predictions of all the individual trees, accomplished through

voting for classification tasks or averaging for regression tasks[75]. Figure 2.8 shows a representation of Random Forest.

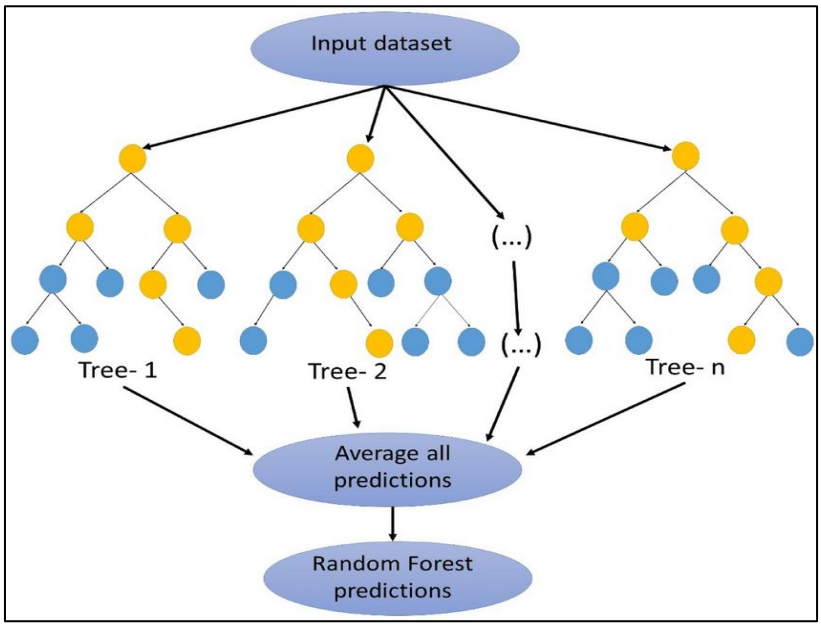

Figure 2.8: Random Forest [76]

The training process of Random Forest involves two levels of randomization. Firstly, a random subset of the original training data is selected using bootstrap sampling or bagging. This technique involves selecting data points from the original dataset with replacement, generating a new subset of data equal in size to the original dataset. Consequently, some data points may be repeated in the new subsets while others may be absent[74][75].

Secondly, for each tree in the Random Forest, a random subset of features is chosen at each split point. This subset typically contains fewer features than the total number available in the dataset. The purpose of this random feature selection is to ensure that the trees make decisions based on different sets of features, thus decorrelating them[74][75].

The decision trees within Random Forest are constructed in a top-down manner by recursively splitting the data based on the selected features. At each split point, the algorithm searches for the optimal feature and split criterion to

optimize a predefined objective function, such as Gini impurity for classification or mean squared error for regression. The process continues until a stopping criterion is met, such as reaching a maximum tree depth or a minimum number of samples per leaf node[74][75].

Algorithm 2.8 shows the steps used for Random Forest algorithm[77].

Algorithm 2.8: Random Forest algorithm
Input: – Training dataset: X (input features), y (target variable) – Number of decision trees: n_trees – Number of features to consider at each split: max_features
Output: – Random Forest model
Steps: 1. Define a list of decision trees: forest = [] 2. For each i from 1 to n_trees, do the following: – Create a bootstrap sample of the training dataset: X_bootstrap, y_bootstrap – Create a decision tree T by training it on the bootstrap sample (X_bootstrap, y_bootstrap) using a specified algorithm (e.g., CART). – Randomly select a subset of features from X_bootstrap with size max_features. – Determine the best split point for T using the selected features. – Split T into two child nodes using the best split point. – Add T to the forest list. 3. Return the Random Forest model containing the list of decision trees.

2.9.5 Gaussian Naïve Bayes

Gaussian Naïve Bayes is a widely used machine learning algorithm for classification tasks. It is a variant of the Naïve Bayes algorithm that assumes the features follow a Gaussian (normal) distribution. This assumption simplifies probability calculations, making the algorithm efficient and easy to implement[78].

The Naïve Bayes algorithm is based on Bayes' theorem, which calculates the posterior probability of a hypothesis using prior knowledge and observed evidence. In classification, it computes the probability of a given class label based on a set of features[78].

The "naïve" aspect of Gaussian Naïve Bayes arises from assuming that the features are conditionally independent given the class label. This means the presence or absence of one feature does not affect the presence or absence of any other feature. Although this assumption is rarely true in practice, Gaussian Naïve Bayes often performs well even when the independence assumption is violated. It is especially useful for high-dimensional data where other algorithms can be computationally expensive or prone to overfitting due to the curse of dimensionality [79].

Algorithm 2.9 shows the steps used in the building the Gaussian Naïve Bayes model[80].

Algorithm 2.9: Gaussian Naïve Bayes algorithm

Input:

- Training dataset: X (input features), y (target variable)

Output:

- Gaussian Naïve Bayes model

Steps:

1. Separate the training dataset into classes based on the target variable.
2. For each class c in the set of unique classes, do the following:
   - Compute the class prior probability: P(c) = count(y = c) / count(y)
   - For each feature X_i in the set of input features, compute the class conditional probability:
     - Calculate the mean (μ) and standard deviation (σ) of X_i for class c.
     - $P(X|Y=c) = \frac{1}{\sqrt{2\pi\sigma_c^2}} \exp(-\frac{(x_i-\mu_y)^2}{2\sigma_y^2})$
3. Return the Gaussian Naïve Bayes model, containing the class prior probabilities and class conditional probabilities.

2.10 Deep Learning

Deep Learning (DL) is a branch of machine learning that concentrates on developing and applying artificial neural networks inspired by the structure and function of the human brain. It is a potent approach for addressing intricate problems and extracting meaningful patterns from extensive datasets. Deep learning has gained immense popularity and achieved remarkable success across diverse domains, including computer vision, natural language processing, speech recognition, and others[81][82].

At the heart of deep learning are artificial neural networks (ANNs), which are computational models comprised of interconnected layers of artificial neurons, also known as nodes or units. These networks are designed to learn and make predictions or decisions based on input data. The term "deep" in deep learning refers to the presence of multiple layers in the network, enabling it to learn hierarchical representations of the data. Figure 2.9 shows a comparison between shallow ANN that is considered as Machine Learning approach, and deep ANN which is considered Deep Learning approach[82]. The figure shows a deeper hidden layer in the Deep Learning model.

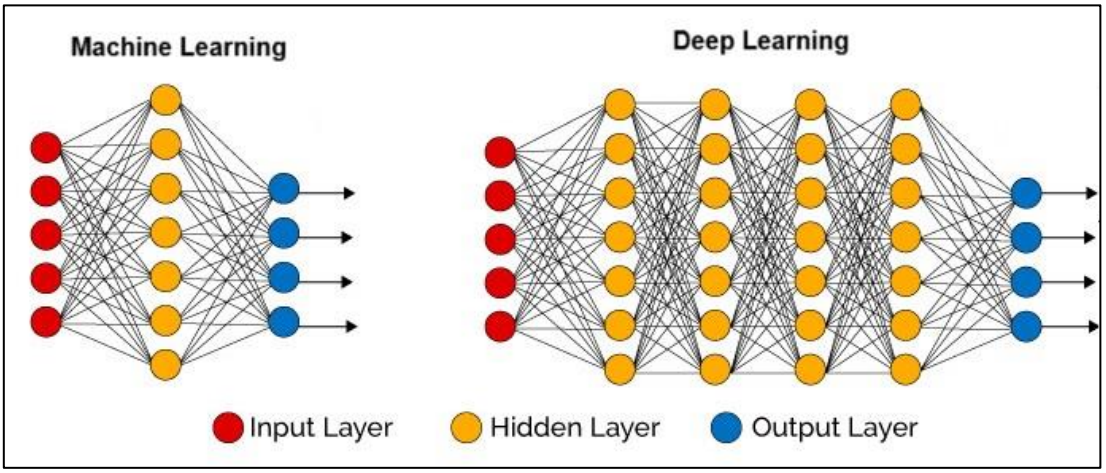

Figure 2.9: a comparison between using ANN in ML and DL [83]

Figure 2.10shows the difference between ML and DL in feature extraction.

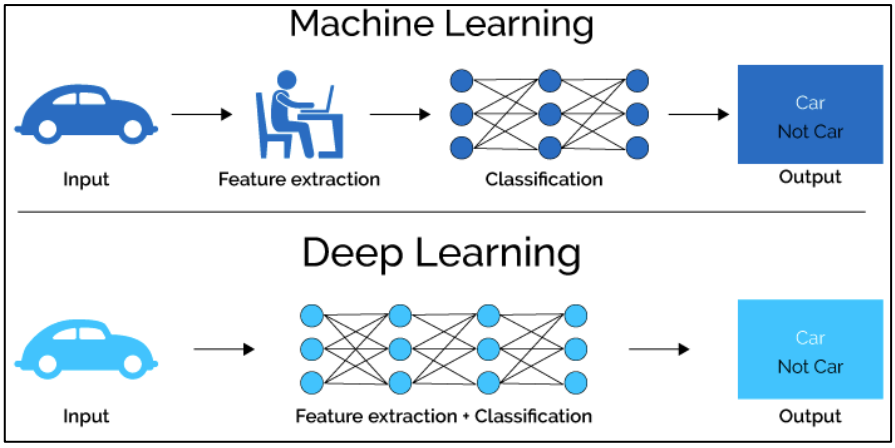

Figure 2.10: Difference between ML and DL in terms of feature extraction [84]

Convolutional Neural Networks (CNNs) are a prominent type of deep learning architecture extensively employed in computer vision tasks such as image classification, object detection, and image segmentation. CNNs exploit spatial correlation in images by employing convolutional layers that apply filters to local regions, allowing the network to capture meaningful patterns and features at different scales[82].

Recurrent Neural Networks (RNNs) are another significant class of deep learning models, commonly utilized in sequential data analysis, such as natural language processing and speech recognition. RNNs possess a unique capability to capture temporal dependencies and handle sequences of varying lengths. They maintain an internal memory or hidden state that enables them to process inputs sequentially and model long-term dependencies[82].

The success of deep learning can be attributed to several factors. First, the availability of large labelled datasets has played a crucial role in effectively training deep learning models. Figure 2.11 shows how DL outperform the ML algorithms giving larger dataset. Additionally, the increased computational power provided by modern GPUs and specialized hardware has accelerated the training and deployment of deep learning models. Furthermore, advancements in optimization algorithms, regularization techniques, and network architectures

have contributed to improving the performance and robustness of deep learning models[82].

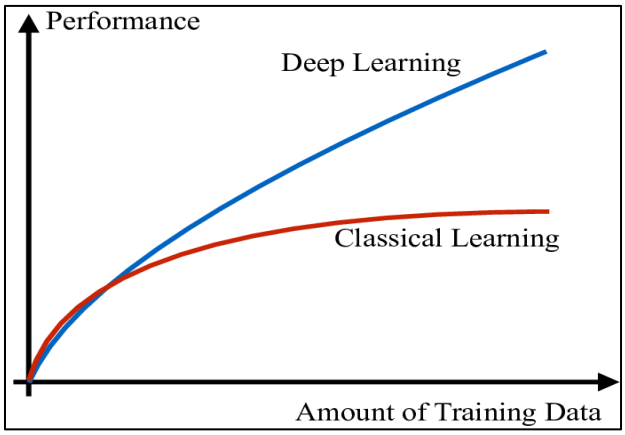

Figure 2.11: comparison between DL and ML performance with amount of data [85]

2.10.1 Convolutional Neural Network

A Convolutional Neural Network (CNN) is a deep learning model developed specifically to process and analyse structured grid-like data, including images, and videos. CNNs have brought about a revolution in computer vision and have achieved impressive success in tasks such as image classification, object detection, and image segmentation[86].

The primary components of a CNN consist of convolutional layers, pooling layers, and fully connected layers (i.e. dense layer).

A. **Convolutional layers:** Convolutional layers serve as the fundamental building blocks of CNNs. They utilize filters (i.e. kernels) that slide across the input data to perform convolution operations. Convolution involves element-wise multiplication of the filter with a local receptive field of the input data, followed by summing the results to generate a feature map[87]. Figure 2.12 shows the convolutional operation between the input image and the kernel.

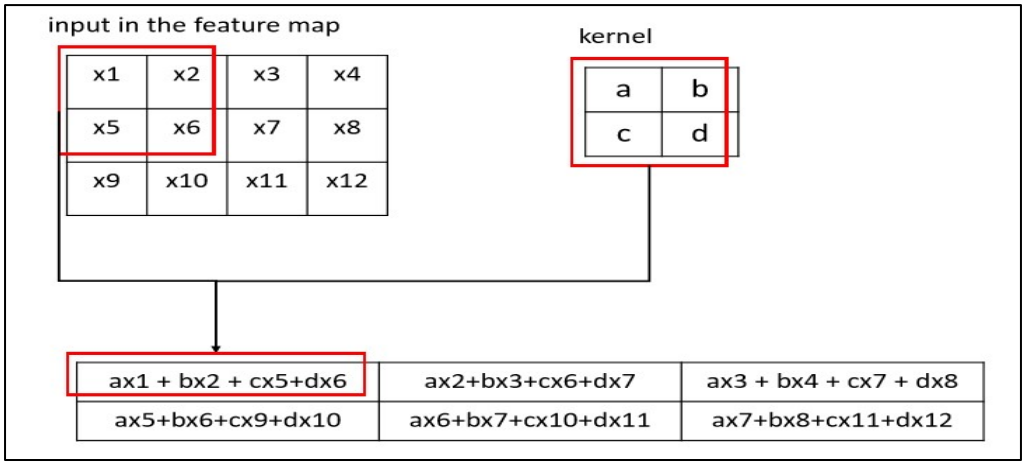

Figure 2.12: Convolution operation[88]

B. **Pooling layers:** Pooling layers are employed to down sample the spatial dimensions of the feature maps while retaining the essential information. The most common pooling operation is max pooling, where the maximum value within a pooling window is selected as the representative value. Pooling helps reduce computational complexity, and control overfitting[89]. Figure 2.13 shows two types of pooling operations. Which are max and average pooling.

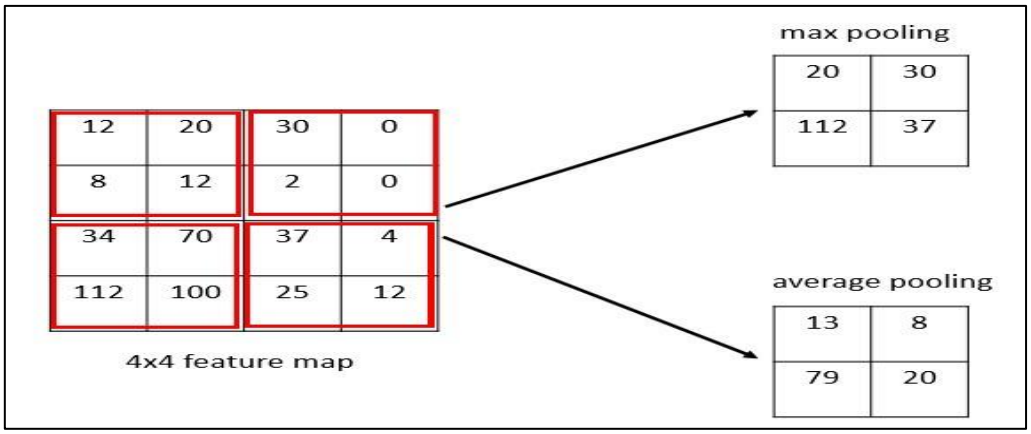

Figure 2.13: Pooling operation[90]

C. **Fully connected layers:** Fully connected layers are typically placed at the end of the CNN architecture. They take the flattened feature maps from previous layers and map them to the target classes or labels[91].

The training process of a CNN involves forward propagation and backpropagation. During forward propagation, the input data passes through the

network layer by layer, and the output is computed. The error between the predicted output and the actual target is measured using a loss function, such as categorical cross-entropy for classification tasks. Backpropagation is then utilized to compute the gradients of the loss with respect to the network parameters, facilitating parameter updates through optimization algorithms. This iterative process continues until the model converges to optimal parameter values[92].

Although CNNs excel in computer vision tasks, their success has extended to other domains such as natural language processing (NLP), speech recognition, and time series data, where data can be represented as grid-like structures[93]–[95].

2.10.2Long Short-Term Memory

Long Short-Term Memory (LSTM) is an architecture of recurrent neural network (RNN) specifically designed to overcome the limitations of traditional RNNs in capturing long-term dependencies within sequential data. LSTM networks have gained substantial popularity across domains such as natural language processing, speech recognition, time series analysis, and more[96].

The gates within an LSTM govern the flow of information into and out of the memory cell. The three primary gates are as follows[96]:

A. **Forget Gate:** This gate determines which information from the previous time step's memory cell state should be discarded or forgotten. It takes the previous hidden state and the current input as inputs and outputs a forget vector that assigns weights to the importance of each memory cell element.
B. **Input Gate:** The input gate decides which new information should be stored in the memory cell. It consists of two components: the input vector and the input modulation vector. The input vector

contains the new candidate values to be added to the memory cell, while the input modulation vector determines the extent to which the candidate values should be added to the memory cell state.

C. **Output Gate:** The output gate controls the amount of information from the current memory cell state that should be exposed to the next hidden state and output. It takes the previous hidden state and the current input, producing an output modulation vector that scales the memory cell state.

Figure 2.14 shows the design of LSTM.

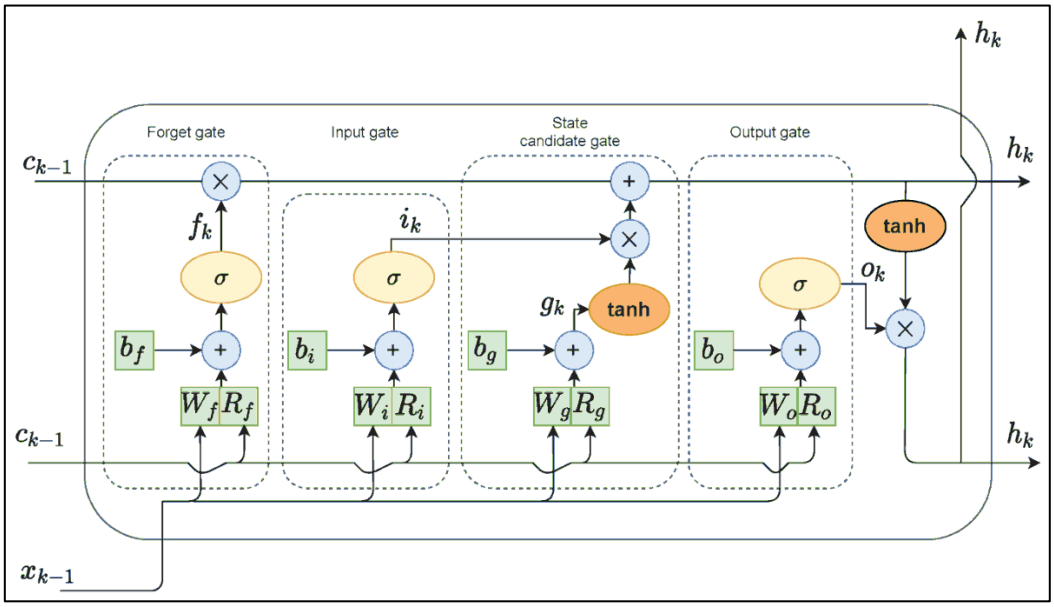

Figure 2.14: LSTM design [97]

2.11 Evaluation

The final model should be evaluated carefully to ensure its performance. The dataset will be divided into training, and testing datasets.

After finishing the training, the testing dataset (the model never trained or validated on the test dataset) will be used to test the model performance, by comparing the prediction of the model on test dataset, with the actual target output. This will reveal the efficiency of the model.

Furthermore, the metrics of *accuracy*, *sensitivity*(i.e. *recall)*, *specificity* and*F1 score* will be used to evaluate the study. These metrics are based on the correctness of the prediction model. If the occurrence of DM is present in a case, it's considered positive. Otherwise its negative. To measure the accuracy of the model, four metrics specified below will be used, which are[98]:

$$accuracy = \frac{True\ Positive + True\ Negative}{True\ Positive + False\ Positive + True\ Negative + False\ Negative} \quad (2.11)$$

$$sensitivity = \frac{True\ Positive}{True\ Positive + False\ Negative} \quad (2.12)$$

$$precision = \frac{True\ Positive}{True\ Positive + False\ Positive} \quad (2.13)$$

$$F1\ score = 2 \cdot \frac{precision \cdot sensitivity}{precision + sensitivity} \quad (2.14)$$

Where the *True* is the correct classification (for both the positive and negative cases). And *False* is for the incorrect classification [99].Figure 2.15 shows how confusion matrix is used to calculate different evaluation metrics.

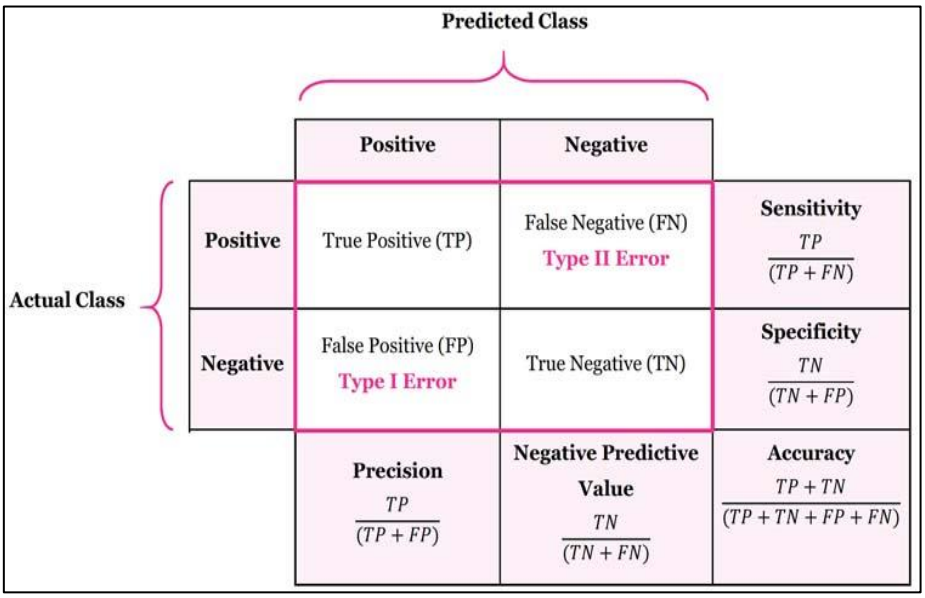

Figure 2.15: Confusion Matrix and the metrics based on it[100]

Chapter Three

Proposed System

3.1 Introduction

This chapter will cover the methods and techniques used for developing the proposed approaches. Including all the pre-processing and model developing steps.

The pre-processing steps includes K-mer, random sampling to solve imbalanced data, ordinal encoder, and Min-Max transformation. After that, feature selection techniques are used. Which are univariate Linear Regression, MI, and ANOVA. And the used dataset is divided into training and testing datasets. These data are fed into different models. In this approach, for the classification task are RF, SVM, DT, and Naïve Bayes. And finally, these models are tested by using testing dataset. Figure 3.1 shows a general diagram of the steps followed to develop this system. And algorithm 3.1 shows the general steps used for the proposed work.

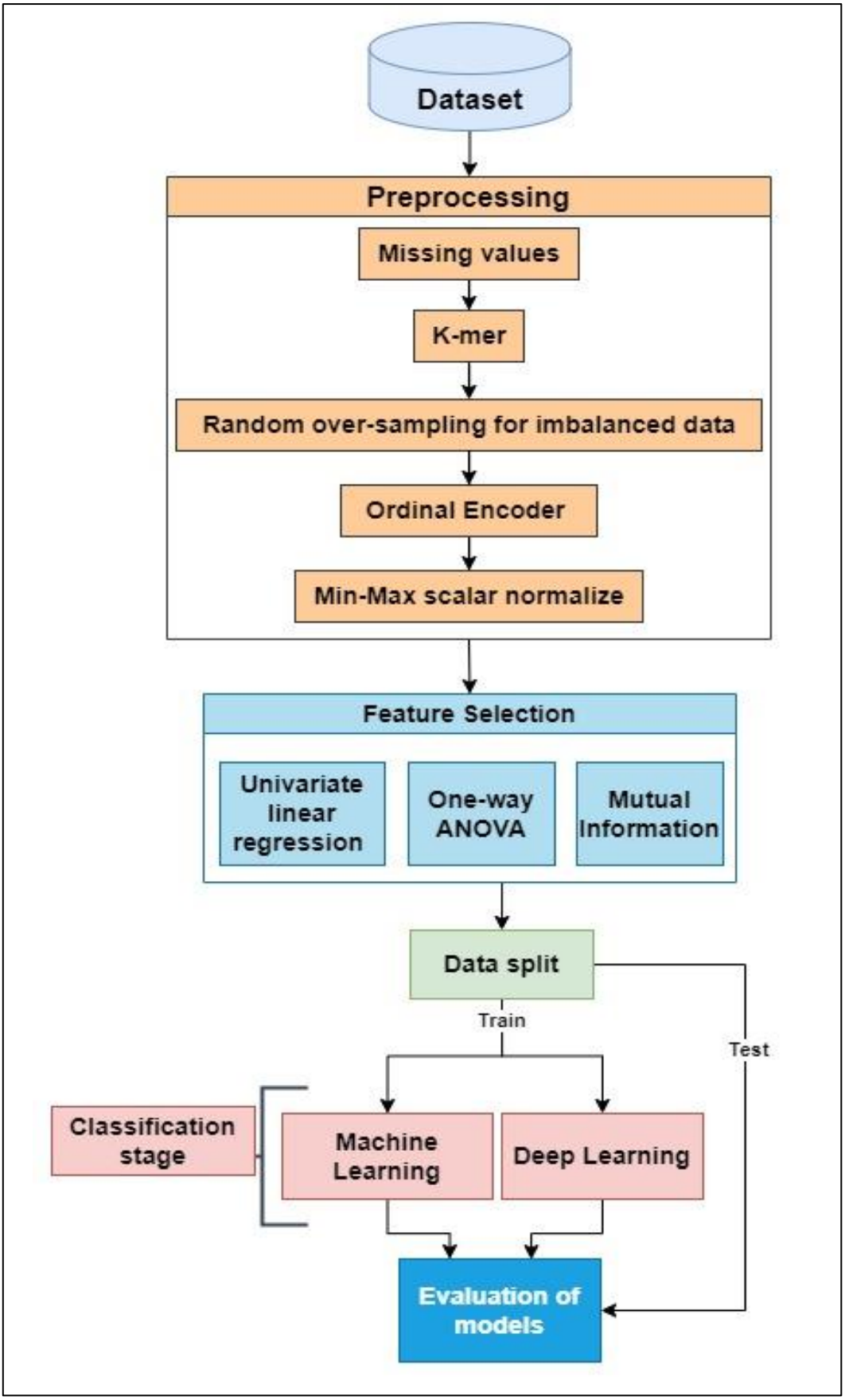

Figure 3.1: general diagram shows the proposed systems

Algorithm 3.1: general approach shows the steps used in building the proposed systems

Input: *dataset*

Output: *trained_model*

Steps:

- Stage 1: Load dataset
- Stage 2: Data pre-processing
    a. Remove missing data
    b. Apply k-mer alignment by using algorithm 2.1
    c. Apply random oversampling by using algorithm 2.2
    d. Apply ordinal encoder by using algorithm 2.3
    e. Apply Min-Max transformation
- Stage 3: Feature selection: apply one of the following approaches
    a. Mutual information by using algorithm 2.4
    b. ANOVA
    c. Univariate Linear Regression by using algorithm 2.5
- Stage 4: Split dataset randomly
    a. 70%-30% training – testing split
    b. 80%-20% training – testing split
- Stage 5: Classification model building by using ML approaches
    c. Call Support Vector Machine algorithm as in algorithm 2.6
    d. Call Random Forest algorithm as in algorithm 2.7
    e. Call Decision Tree algorithm as in algorithm 2.8
    f. Call Gaussian Naïve Bayes algorithm as in algorithm 2.9
- Stage 6: Deep Learning model building
    a. Call CNN-LSTM algorithm as in algorithm 3.2
- Stage 6: Evaluate the proposed models
    a. Calculate *accuracy* by using equation 2.11
    b. Calculate *recall* by using equation 2.12
    c. Calculate *precision* by using equation 2.13
    d. Calculate *F1-score* by using equation 2.14

3.2 Data pre-processing

The pre-processing is important step to make the data more suitable to be used in training the models, to extract features and classify the data correctly.

3.2.1 Missing value

To address the missing values within DNA sequences, the approach involves excluding the entire DNA sequence containing the missing values. This strategy is adopted due to the potential introduction

of noise in the data, which could result in an inaccurate model. Consequently, data imputation is deemed unattainable for DNA sequential data, and instances with missing values are simply omitted. Given that only a singular instance possesses missing values, the act of omitting it does not exert any impact on the overall dataset size.

3.2.2 K-mer

The k-mer is used to select subsets of the input DNA sequences, and create new features. Six values to detect the datasets sizes are used in k-mer which are 3-8. The theory of how the k-mer working is explained in chapter 2. Algorithm 2.1 shows how the k-mer is working.

3.2.3 Random oversampling for imbalanced data

Data balance is crucial step, since having one class with larger instance would lead the classifier to be biased. In order to solve this problem, the number of instances in all classes should be balanced. This goal is achieved by the minor class is filled by picking samples at random with replacement. Before the oversampling operation, there was 10199 instances having diabetes out of 14570. And the remaining 4371 are healthy. After the oversampling, the total number of instances 20398. Increasing the number of healthy people to 10199 instances. Algorithm 2.2 shows how Random over Sampling for imbalanced data is working.

3.2.4 Ordinal Encoder

The features are converted from categorical into integer. This step is necessary since the inputs and output of the Machine Learning models require numeric values. Algorithm 2.3 shows how the ordinal encoder is working.

3.2.5 Min-Max transformation

Machine Learning models perform better when the inputs are scaled to standard range. This step will make the range of the input features between 0 to 1. Equation 2.2 shows the mathematical formula for the Min-Max transformation.

3.3 Feature Selection Stage

The Feature Selection is important step in Machine Learning models. Since it selects only the important features and ignore the unnecessary features that may reduce the performance and accuracy of the model. In this work, three different Feature Selection methods are tested. MI, one-way ANOVA, and Univariate linear regression methods for Feature Selection. These methods are explained in chapter two deeply. And algorithms 2.4, and 2.5 shows how these approaches are working in details respectively. After applying each the feature selection process on the original dataset, 20%, 50%, and 75% percentage of threshold are selected each time.

3.4 Machine Learning Model Building Stage

The process of model building consists of splitting the data, selecting the model, and configure the hyper-parameters.

Data split have two configures. 70% - 30%, and 80% - 20% for each one of the used models. *Sklearn* library in Python is used to split the dataset into two sub-datasets, by using random splitting.

The used models are Random Forest, with 100 trees in the forest. Gaussian Naive Bayes. Support Vector Machine, for the kernel technique is the Radial Basis Function (RBF). And Decision Tree Classifiers, in which Gini impurity is used. These algorithms are deeply explained in chapter two. Figure 3.2 shows the used Machine Learning models.

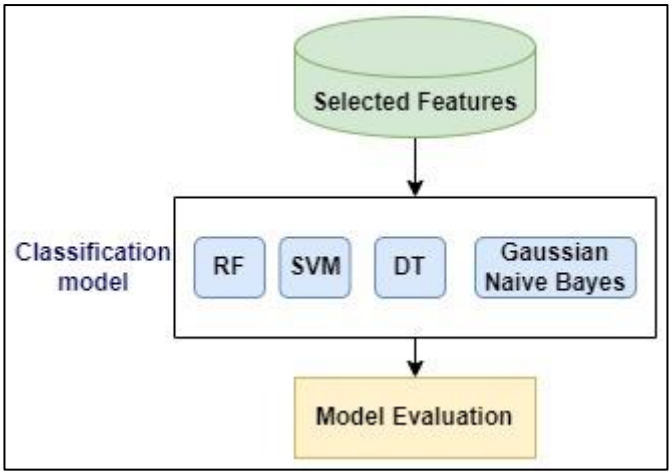

Figure 3.2: Block Diagram of ML Classification Models

3.5 Deep Learning Model Building

The Deep Learning model used for the classification of the DNA gene into diabetes or healthy person. Figure 3.3 shows how the Deep Learning model is incorporated with the pipeline of the model training. The used library is Keras. The data pre-processed as discussed in section 3.2. The proposed model is consisting of 12 layers. Table 3.1 shows the design of the proposed CNN-LSTM model. The proposed CNN-LSTM model is plotted in Figure 3.4.

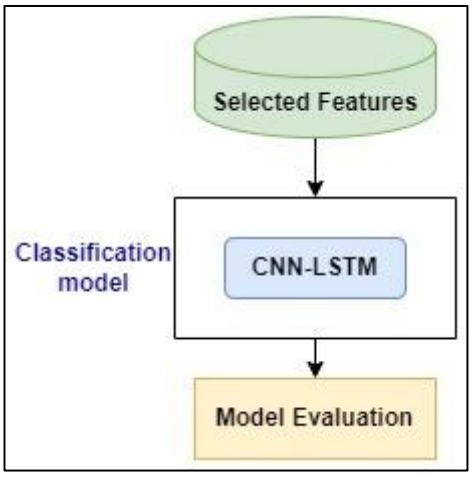

Figure 3.3: Block Diagram of DL Classification Model

Table 3.1: CNN-LSTM proposed system design

Layer	Output shape	# of parameters
Conv1D	None, 26, 16	96
MaxPooling	None, 26, 16	0
Conv1D	None, 22, 32	2592
MaxPooling	None, 22, 32	0
LSTM	None, 22, 32	8320
MaxPooling	None, 22, 32	0
Conv1D	None, 18, 32	5152
MaxPooling	None, 18, 32	0
LSTM	None, 18, 16	3136
MaxPooling	None, 18, 16	0
Flatten	None, 288	0
Dense	None, 2	578
Total trainable parameters		19,874

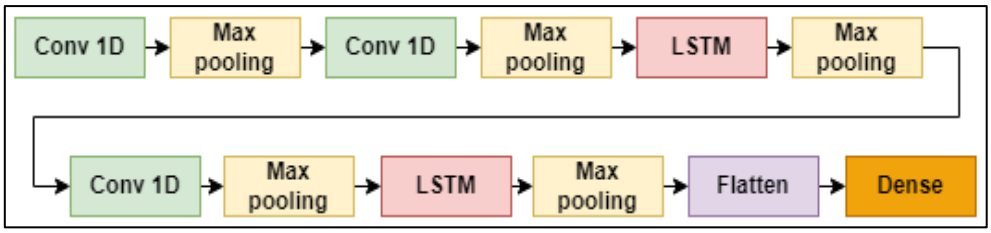

Figure 3.4: CNN-LSTM proposed model

For all the convolutional layers used in the proposed system, the kernel size is 5, stride is 1, and the padding is set to *valid*, means that the filter window always stays inside the input. And the number of filters are 16, 32, 32 for the first, third, and seventh layers respectively. For LSTM layers 5 and 9, the sizes are 32 and 16 respectively. The *return_sequence* is set to true, meaning it will output all the hidden states of each time steps. And for the last layer (Dense layer), it is used for the purpose of classification. With 2 neurons. And Softmax activation function.

The learning process used Adam optimization algorithm. The initial learning rate value is set to 0.001. And the binary cross entropy is used as the loss function. With 300 epochs, and 1024 batch size. Algorithm 3.2 display the steps of the training of the CNN-LSTM model proposed in this work.

Algorithm 3.2: steps of the training of the CNN-LSTM model

Input: *selected_features_dataset*

Output: *CNN-LSTM model*

1. **Model Initialization**
   - Initialize the CNN layers, LSTM layer, and output layer.
   - Set the CNN-LSTM model design, by selecting how many filters are required, kernel sizes, pooling layers, LSTM units, etc.
   - Initialize the optimizer (ADAM optimization), loss function (binary cross entropy), batch size to 1024, epochs to 300, learning rate to 0.001.
2. **Training Loop**
   - Iterate over the dataset for a specified number of epochs.
   - Shuffle the dataset before each epoch to introduce randomness.
   - Divide the dataset into batches of a certain size.
   - Iterate over each batch:
     - Forward Pass:
       - Pass the batch of samples through the CNN layers to extract features.
       - Reshape the output of the CNN layers to match the LSTM input shape.
       - Pass the reshaped features through the LSTM layer.
       - Obtain the output of the LSTM layer.
       - Pass the LSTM output through the output layer to get the final predictions.
     - Loss function is used to calculate the loss value, by using both the true labels and the model prediction.
     - Backward Pass:
       - Calculate the loss gradients in respect to the model weights (i.e. parameters).
       - Use Adam optimizer to update the weights of the model.
       - Clear the gradients for the next iteration.
3. **Evaluation**
   - After training, evaluate the model on a separate test dataset.
   - Pass the test dataset through the trained model to obtain predictions.
   - In order to assess the performance of the model, compute evaluation metrics (accuracy, precision, recall, and F1-score).

Chapter Four

Results and Discussions

4.1 Introduction

This chapter will show and discuss the results of the proposed systems. Notice that the results are divided into sub-groups based on the methods and values of feature selection, Machine Learning, and Deep Learning algorithms.

Each result obtained from each phase and stage will be displayed and discussed. Furthermore, a section for each the software and hardware requirements, and the dataset will be discussed.

4.2 Software and Hardware Requirements

The developed system is built and tested on personal computer (PC). The processor is Intel® Core™ i7-6600U CPU, with 2.60 GHz. The installed Random Access Memory (RAM) is 8 GB. The Operating System (OS) is Windows® 10 Pro, 64-bit, and x64 based processor. The used programming language is Python v 3.11.4. By using Project Jupyter software.

4.3 Dataset

The used dataset is obtained from National Library of Medicine. This library is represented by an official website managed and run by United States government, providing biotechnology information, datasets, medical experiments, researches, and other related materials[101].

The used dataset is DNA Gene type protein coding, about human insulin. This gene encodes insulin, a peptide hormone that plays a vital role in the regulation of carbohydrate and lipid metabolism. After removal of the precursor signal peptide, proinsulin is post-translationally cleaved into three peptides: the B chain and A chain peptides, which are covalently linked via two disulphide bonds to form insulin, and C-peptide. Binding of insulin to the insulin receptor (INSR) stimulates glucose uptake[102].

The dataset contains the DNA sequence of 14571 people. Each DNA sequence length is 301. 10199 DNA sequences are labelled diabetes, and 4371

are labelled non diabetes. Figure 4.1 shows some sample of the DNA sequence of the used dataset.

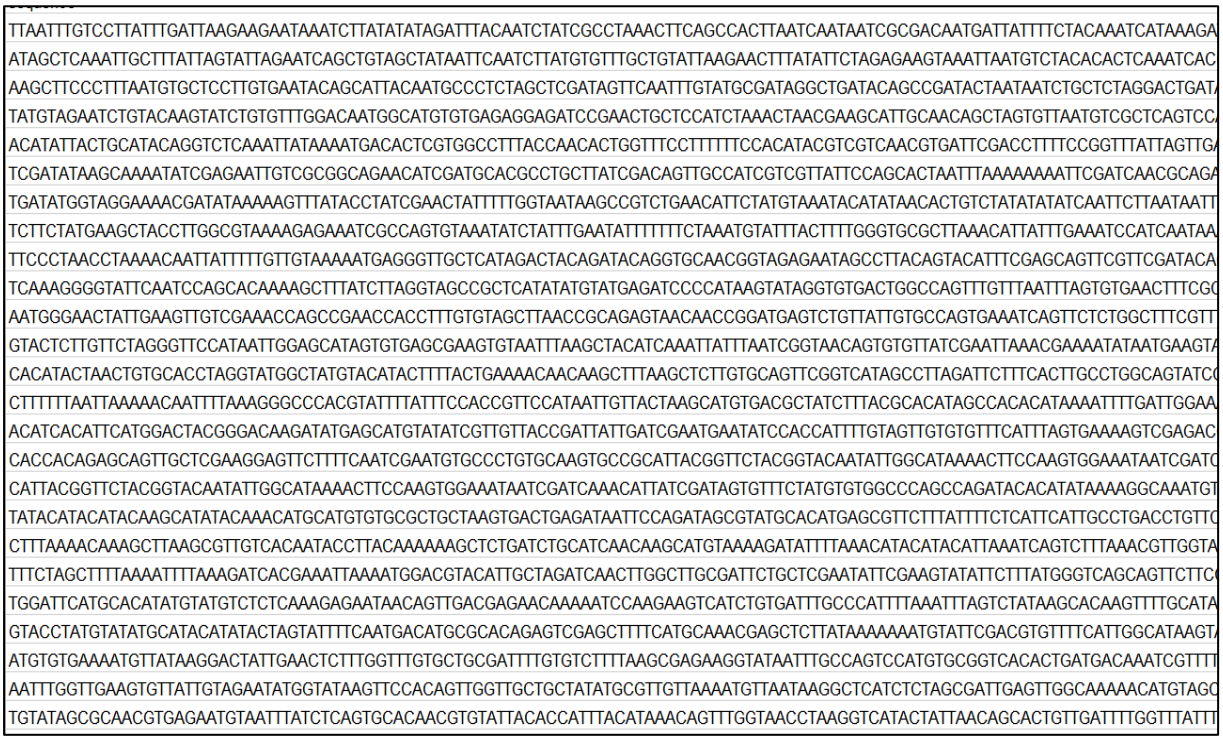

Figure 4.1: sample of the used dataset

4.4 Dataset Pre-Processing Stage Results

The dataset contains the DNA sequence of 14571 people. Each DNA sequence length is 301. 10199 DNA sequences are labelled diabetes, and 4371 are labelled healthy.

In the pre-processing phases, the dataset had many changes.

4.4.1 Missing Values

The missing values found in the DNA sequences are treated by omitting the whole DNA sequence. Since the presence of the missing value will lead to noisy data, and thus inaccurate model. And since the data imputation is not achievable goal with DNA sequential data, the instance with missing values are omitted. And since only one instance have missing values, omitting it will not effect on the dataset size. Figure 4.2 shows the missing value in the dataset.

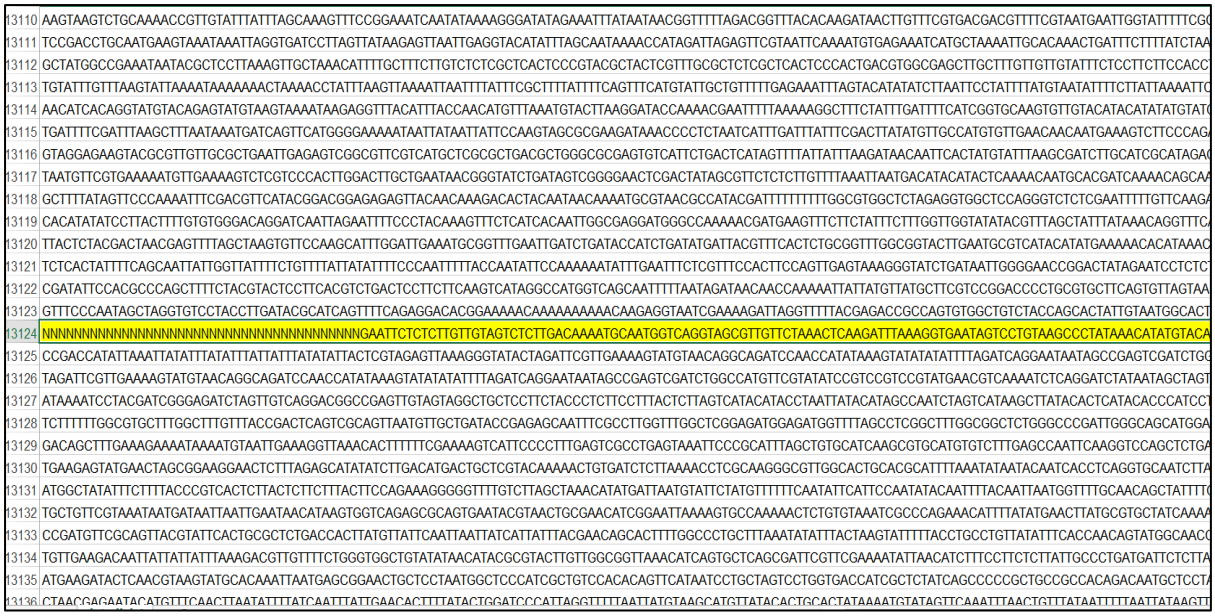

Figure 4.2: the missing value in the dataset

4.4.2 K-mer

K-mer encoding is an approach that involves generating substrings of a specific length, denoted as K, from a biological sequence. These K-mers are composed of nucleotides, namely A, T, G, and C. The utilization of K-mers serves various purposes, including DNA sequence assembly, enhancement of heterologous gene expression, species identification in metagenomic samples, and the development of attenuated vaccines. By employing K-mers, subsets of the input DNA sequences can be selected and transformed into new features .Six sizes are used in k-mer size which are 3, 4, 5, 6, 7, and 8.

Table 4.1 shows a sample of the dataset generated after applying k-mer sampling, with 6 k-mer size. You can notice that each word have the length of 6 characters. And the total number of features is 295.

Table 4.1: sample of the result of words obtained from 3 k-mer size

Class	0	1	2	3	4	…	293	294	295
1	att	ttt	ttg	tgt	gtc	…	tat	att	tta

1	gct	ctc	tca	caa	aaa	…	gga	gaa	aac
1	ctt	ttc	tcc	ccc	cct	…	ata	tat	ata
1	gta	tag	aga	gaa	aat	…	cgc	gca	cat
1	tat	att	tta	tac	act	…	atc	tcg	cgg
…	…	…	…	…	…	…	…	…	…
0	tat	att	ttt	ttt	tta	…	tat	atg	tgg
0	atc	tcc	cca	caa	aaa		gac	aca	cat
0	tcg	cgg	gga	gac	acg	…	aca	cat	ata
0	tac	acc	cct	ctt	ttt	…	tga	gac	aca

4.4.3 Random Oversampling for Imbalanced Data

Before the data balancing, the original dataset contains 14570 instances. 4371 of them of class normal. And 10199 instances of diabetes instances. After applying the random over-sampling, the total features are 20398 instances. Increasing the normal class into 10199 instances.

4.4.4 Ordinal Encoder

The features are converted from categorical into integer. This step is necessary since the inputs and output of the Machine Learning models require numeric values.

Following Figure 4.3shows how the data are encoded.

class	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	60	48	3	15	63	62	59	45	53	23	31	60	51	15	63	62
1	12	50	9	39	29	52	16	0	3	15	62	57	39	31	63	60
1	2	9	39	31	61	53	21	23	31	63	60	48	3	14	59	46
1	51	14	59	44	50	8	32	3	13	55	30	59	44	49	4	16
1	4	19	12	51	15	60	49	7	30	57	36	19	12	49	4	18
1	54	24	35	12	51	12	48	2	9	36	16	0	0	3	12	51
1	56	35	12	51	14	58	43	44	50	10	40	32	0	0	1	6
1	55	31	61	55	28	51	14	56	32	2	9	39	28	49	5	23
1	61	53	21	23	28	48	1	5	23	28	48	0	0	1	4	16
1	52	16	0	2	10	42	42	43	44	51	15	61	52	16	3	13
1	3	14	58	42	40	32	1	7	28	51	15	62	56	32	2	11
1	44	49	7	29	55	31	62	59	47	61	55	28	50	10	42	43
1	17	4	19	12	49	7	28	48	1	7	30	59	46	57	36	17
1	31	63	63	63	63	60	48	3	15	60	48	0	0	0	1	4
1	4	19	13	52	17	4	19	15	61	52	19	14	58	40	33	7
1	17	5	20	17	4	18	8	34	9	36	18	11	47	62	57	39
1	19	15	60	49	6	26	43	47	61	55	28	49	6	26	43	44
1	51	12	49	4	19	12	49	4	19	12	49	4	16	2	9	36
1	31	63	60	48	0	0	1	4	16	0	2	9	39	31	60	48
1	63	61	55	28	50	9	39	31	63	63	60	48	0	0	3	15
1	58	40	35	15	61	52	19	14	57	36	17	4	19	12	51	14
1	44	49	5	23	28	51	14	59	44	51	12	51	14	57	36	19
1	14	59	46	59	46	56	32	0	0	3	14	59	47	60	51	12
1	3	15	63	62	58	43	47	62	56	32	2	11	46	59	47	60
1	59	44	51	12	50	9	38	25	36	16	1	6	27	46	56	34
1	46	56	32	0	0	3	12	49	4	19	13	54	24	35	12	51

Figure 4.3: dataset after encoding

4.4.5 Min-Max transformation

Machine Learning models perform better when the inputs are scaled to standard range. This step will make the range of the input features between 0 to 1. Figure 4.4 shows the data after applying Min-Max transformation.

class	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
1	0.952380952	0.761904762	0.047619048	0.238095238	1	0.984126984	0.936507937	0.714285714	0.841269841	0.365079365	0.492063492	0.952380952	0.80952381	0.238095238	1	0.984126984	0.8888888
1	0.19047619	0.793650794	0.142857143	0.619047619	0.46031746	0.825396825	0.253968254	0	0.047619048	0.238095238	0.984126984	0.904761905	0.619047619	0.492063492	1	0.952380952	0.809523
1	0.031746032	0.142857143	0.619047619	0.492063492	0.968253968	0.841269841	0.333333333	0.365079365	0.492063492	1	0.952380952	0.761904762	0.047619048	0.222222222	0.936507937	0.73015873	0.9047619
1	0.80952381	0.222222222	0.936507937	0.698412698	0.793650794	0.126984127	0.507936508	0.047619048	0.206349206	0.873015873	0.476190476	0.936507937	0.698412698	0.777777778	0.063492063	0.253968254	0.0317460
1	0.063492063	0.301587302	0.19047619	0.80952381	0.238095238	0.952380952	0.777777778	0.111111111	0.476190476	0.904761905	0.571428571	0.301587302	0.19047619	0.777777778	0.063492063	0.285714286	0.1587301
1	0.857142857	0.380952381	0.555555556	0.19047619	0.80952381	0.19047619	0.761904762	0.031746032	0.142857143	0.571428571	0.253968254	0	0	0.047619048	0.19047619	0.80952381	0.2063492
1	0.888888889	0.555555556	0.19047619	0.80952381	0.222222222	0.920634921	0.682539683	0.698412698	0.793650794	0.158730159	0.634920635	0.507936508	0	0	0.015873016	0.095238095	0.3809523
1	0.873015873	0.492063492	0.968253968	0.873015873	0.444444444	0.80952381	0.222222222	0.888888889	0.507936508	0.031746032	0.142857143	0.619047619	0.444444444	0.777777778	0.079365079	0.365079365	0.4920634
1	0.968253968	0.841269841	0.333333333	0.365079365	0.444444444	0.761904762	0.015873016	0.079365079	0.365079365	0.444444444	0.761904762	0	0	0.015873016	0.063492063	0.253968254	0.0476190
1	0.825396825	0.253968254	0	0.031746032	0.158730159	0.666666667	0.666666667	0.682539683	0.698412698	0.80952381	0.238095238	0.968253968	0.825396825	0.253968254	0.047619048	0.206349206	0.8412698
1	0.047619048	0.222222222	0.920634921	0.666666667	0.634920635	0.507936508	0.015873016	0.111111111	0.444444444	0.80952381	0.238095238	0.984126984	0.888888889	0.507936508	0.031746032	0.174603175	0.7460317
1	0.698412698	0.777777778	0.111111111	0.46031746	0.873015873	0.492063492	0.984126984	0.936507937	0.746031746	0.968253968	0.873015873	0.444444444	0.793650794	0.158730159	0.666666667	0.682539683	0.7460317
1	0.26984127	0.063492063	0.301587302	0.19047619	0.777777778	0.111111111	0.444444444	0.761904762	0.015873016	0.111111111	0.476190476	0.936507937	0.73015873	0.904761905	0.571428571	0.26984127	0.0793650
1	0.492063492	1	1	1	1	0.952380952	0.761904762	0.047619048	0.238095238	0.952380952	0.761904762	0	0	0	0.015873016	0.063492063	0.2539682
1	0.063492063	0.301587302	0.206349206	0.825396825	0.26984127	0.063492063	0.301587302	0.238095238	0.968253968	0.825396825	0.301587302	0.222222222	0.920634921	0.634920635	0.523809524	0.111111111	0.4444444
1	0.26984127	0.079365079	0.317460317	0.26984127	0.063492063	0.285714286	0.126984127	0.53968254	0.142857143	0.571428571	0.285714286	0.174603175	0.746031746	0.984126984	0.904761905	0.619047619	0.460317
1	0.301587302	0.238095238	0.952380952	0.777777778	0.095238095	0.412698413	0.682539683	0.746031746	0.968253968	0.873015873	0.444444444	0.777777778	0.095238095	0.412698413	0.682539683	0.698412698	0.7777777
1	0.80952381	0.19047619	0.777777778	0.063492063	0.301587302	0.19047619	0.777777778	0.063492063	0.301587302	0.19047619	0.777777778	0.063492063	0.253968254	0.031746032	0.142857143	0.571428571	0.3015873
1	0.492063492	1	0.952380952	0.761904762	0	0	0.015873016	0.063492063	0.253968254	0	0.031746032	0.142857143	0.619047619	0.492063492	0.952380952	0.761904762	0.0317460
1	1	0.968253968	0.873015873	0.444444444	0.793650794	0.142857143	0.619047619	0.492063492	1	1	0.952380952	0.761904762	0	0	0.047619048	0.238095238	
1	0.920634921	0.634920635	0.555555556	0.238095238	0.968253968	0.825396825	0.301587302	0.222222222	0.904761905	0.571428571	0.26984127	0.063492063	0.301587302	0.19047619	0.80952381	0.222222222	0.9365079
1	0.698412698	0.777777778	0.079365079	0.365079365	0.444444444	0.80952381	0.222222222	0.936507937	0.698412698	0.80952381	0.19047619	0.80952381	0.222222222	0.904761905	0.571428571	0.301587302	0.190476
1	0.222222222	0.936507937	0.73015873	0.936507937	0.73015873	0.888888889	0.507936508	0	0	0.047619048	0.222222222	0.936507937	0.746031746	0.952380952	0.80952381	0.19047619	0.7619047
1	0.047619048	0.238095238	1	0.984126984	0.920634921	0.682539683	0.746031746	0.984126984	0.888888889	0.507936508	0.031746032	0.174603175	0.73015873	0.936507937	0.746031746	0.952380952	0.809523
1	0.936507937	0.698412698	0.80952381	0.19047619	0.793650794	0.142857143	0.603174603	0.396825397	0.571428571	0.253968254	0.015873016	0.095238095	0.428571429	0.73015873	0.888888889	0.53968254	0.1269841
1	0.73015873	0.888888889	0.507936508	0	0	0.047619048	0.19047619	0.777777778	0.063492063	0.301587302	0.206349206	0.857142857	0.380952381	0.555555556	0.19047619	0.80952381	0.2380952

Figure 4.4: dataset after transformation

4.5 Feature Selection Results

The Feature Selection is important step in Machine Learning models. Since it select only the important features and ignore the unnecessary feature that may reduce the performance and accuracy of the model. In this work, three different Feature Selection methods are tested. MI, one way ANOVA, and Univariate linear regression methods for Feature Selection.

The number of selected features are based on percentage threshold values. Which are 20%, 50%, and 75%.Table 4.2 shows the number of words and features for each value of the selected k-mer and feature selection.

Table 4.2: number of words and features for each k-mer size

K-mer size	Number of words	Number of features 20%	Number of features 50%	Number of features 75%
3	299	60	149	224

4	298	60	149	223
5	297	60	148	222
6	296	59	148	222
7	295	59	147	221
8	294	59	147	220

4.6 Machine Learning Results

This section will show the results of the Machine Learning approaches only. The next section will be dedicated into the Deep Learning approaches.

4.6.1 First Data Split

In this subsection the results of the dataset split into 80% training, and 20% testing will be explained. Noticed that each table is consisting of sub-tables. Each sub-table is responsible for showing the accuracy, precession, recall, and F1 score of each of the four proposed Machine Learning models. Moreover, the whole table consists of different sub-tables. Each sub-table have different k-mer size. And each table will show a single percentage of selected features.

Table 4.3: results of Machine Learning models with 20% features selected, and 80%-20% dataset split

	Feature Selection with N=20, K=8											
	Regression				**ANOVA**				**MI**			
Methods	accuracy	Precision	recall	f1-score	Accuracy	precision	recall	f1-score	accuracy	precision	recall	f1-score
RandomForest	**0.88**	**0.88**	**0.88**	**0.88**	**0.87**	**0.87**	**0.87**	**0.87**	**0.88**	**0.88**	**0.88**	**0.88**
GaussianNB	0.69	0.69	0.69	0.69	0.66	0.66	0.66	0.66	0.7	0.7	0.7	0.7
SVM	0.77	0.77	0.77	0.77	0.75	0.75	0.75	0.75	0.77	0.77	0.77	0.77
DecisionTree	0.81	0.82	0.81	0.81	0.82	0.83	0.82	0.82	0.81	0.83	0.81	0.82
	Feature Selection with N=20, K=7											
	Regression				**ANOVA**				**MI**			
	accuracy	precision	recall	f1-score	Accuracy	precision	recall	f1-score	accuracy	precision	recall	f1-score
RandomForest	**0.87**	**0.87**	**0.87**	**0.87**	**0.88**	**0.88**	**0.88**	**0.88**	**0.88**	**0.89**	**0.88**	**0.88**
GaussianNB	0.69	0.69	0.69	0.69	0.66	0.66	0.66	0.66	0.71	0.71	0.71	0.71
SVM	0.77	0.77	0.77	0.77	0.76	0.76	0.76	0.76	0.79	0.79	0.79	0.79
DecisionTree	0.82	0.83	0.82	0.82	0.82	0.83	0.82	0.82	0.82	0.83	0.82	0.82
	Feature Selection with N=20, K=6											
	Regression				**ANOVA**				**MI**			
	accuracy	precision	recall	f1-score	Accuracy	precision	recall	f1-score	accuracy	precision	recall	f1-score
RandomForest	**0.88**	**0.88**	**0.88**	**0.88**	**0.87**	**0.88**	**0.87**	**0.87**	**0.87**	**0.88**	**0.87**	**0.87**
GaussianNB	0.68	0.69	0.68	0.68	0.69	0.69	0.69	0.69	0.69	0.69	0.69	0.69
SVM	0.77	0.77	0.77	0.77	0.77	0.77	0.77	0.77	0.78	0.78	0.78	0.78
DecisionTree	0.82	0.84	0.82	0.82	0.81	0.82	0.81	0.81	0.82	0.83	0.82	0.82
	Feature Selection with N=20, K=5											
	Regression				**ANOVA**				**MI**			
	accuracy	precision	recall	f1-score	Accuracy	precision	recall	f1-score	accuracy	precision	recall	f1-score
RandomForest	**0.89**	**0.89**	**0.89**	**0.89**	**0.88**	**0.88**	**0.88**	**0.88**	**0.88**	**0.88**	**0.88**	**0.88**
GaussianNB	0.7	0.7	0.7	0.7	0.69	0.69	0.69	0.69	0.7	0.7	0.7	0.7

SVM	0.78	0.78	0.78	0.78	0.78	0.78	0.78	0.78	0.79	0.79	0.79	0.79
DecisionTree	0.82	0.84	0.82	0.82	0.81	0.83	0.81	0.81	0.82	0.84	0.82	0.82
	Feature Selection with N=20, K=4											
	Regression				**ANOVA**				**MI**			
	accuracy	precision	recall	f1-score	Accuracy	precision	recall	f1-score	accuracy	precision	recall	f1-score
RandomForest	**0.88**	**0.89**	**0.88**	**0.88**	**0.88**	**0.89**	**0.88**	**0.88**	**0.87**	**0.87**	**0.87**	**0.87**
GaussianNB	0.69	0.69	0.69	0.69	0.67	0.68	0.67	0.67	0.69	0.7	0.69	0.69
SVM	0.78	0.78	0.78	0.78	0.78	0.78	0.78	0.78	0.78	0.78	0.78	0.78
DecisionTree	0.82	0.84	0.82	0.82	0.81	0.83	0.81	0.81	0.81	0.83	0.81	0.81
	Feature Selection with N=20, K=3											
	Regression				**ANOVA**				**MI**			
	accuracy	precision	recall	f1-score	Accuracy	precision	recall	f1-score	accuracy	precision	recall	f1-score
RandomForest	**0.88**	**0.88**	**0.88**	**0.88**	**0.88**	**0.88**	**0.88**	**0.88**	**0.88**	**0.88**	**0.88**	**0.88**
GaussianNB	0.69	0.69	0.69	0.69	0.68	0.69	0.68	0.68	0.69	0.69	0.69	0.69
SVM	0.79	0.79	0.79	0.79	0.78	0.78	0.78	0.78	0.78	0.78	0.78	0.78
DecisionTree	0.8	0.82	0.8	0.8	0.81	0.82	0.81	0.81	0.82	0.83	0.82	0.82

Table 4.3 shows 6 sub-tables. Each sub-table consist three feature selection approaches (Regression, ANOVA, and Mutual Information). For each feature selection method, the accuracy, precision, recall, and f1-score are calculated for the four Machine Learning models. These models are RandomForest (RF), Gaussian Naïve Bayes, Support Vector Machine (SVM), and Decision Tree (DT).

Each sub-table have a different configuration in terms of number of k-mer selected. The used k-mer values are denoted by the variable *k*. which the value of it starts from 8 down to 3. Notice that the number of features selected in this table are all set to 20.

You can notice that Random Forrest achieved the highest classification accuracy metrices in all the tested configurations. The highest value for accuracy metric is 89%, by using Random Forrest with Regression feature selection method, and 5 k-mer. And the lowest accuracy value for accuracy metric is 87%. Concluding that there are not big differences between the tested configurations.

Table 4.4: results of Machine Learning models with 75% features selected, and 80%-20% dataset split

	Feature Selection with N=75 , K=8											
	Regression				**ANOVA**				**MI**			
Methods	accuracy	precision	recall	f1-score	Accuracy	precision	recall	f1-score	accuracy	precision	recall	f1-score
RandomForest	**0.89**	**0.89**	**0.89**	**0.89**	**0.88**	**0.88**	**0.88**	**0.88**	**0.90**	**0.90**	**0.90**	**0.90**
GaussianNB	0.71	0.71	0.71	0.71	0.71	0.71	0.71	0.71	0.7	0.7	0.7	0.7
SVM	0.83	0.83	0.83	0.83	0.82	0.82	0.82	0.82	0.83	0.83	0.83	0.83
DecisionTree	0.82	0.83	0.82	0.82	0.82	0.83	0.82	0.82	0.82	0.83	0.82	0.82
	Feature Selection with N=75 , K=7											
	Regression				**ANOVA**				**MI**			
	accuracy	precision	recall	f1-score	Accuracy	precision	recall	f1-score	accuracy	precision	recall	f1-score
RandomForest	**0.89**	**0.89**	**0.89**	**0.89**	**0.88**	**0.88**	**0.88**	**0.88**	**0.89**	**0.89**	**0.89**	**0.89**
GaussianNB	0.72	0.72	0.72	0.72	0.71	0.71	0.71	0.71	0.71	0.71	0.71	0.71
SVM	0.83	0.83	0.83	0.83	0.82	0.82	0.82	0.82	0.83	0.83	0.83	0.83
DecisionTree	0.82	0.84	0.82	0.82	0.82	0.83	0.82	0.82	0.81	0.82	0.81	0.81
	Feature Selection with N=75 , K=6											
	Regression				**ANOVA**				**MI**			
	accuracy	precision	recall	f1-score	Accuracy	precision	recall	f1-score	accuracy	precision	recall	f1-score
RandomForest	**0.89**	**0.89**	**0.89**	**0.89**	**0.89**	**0.89**	**0.89**	**0.89**	**0.88**	**0.88**	**0.88**	**0.88**
GaussianNB	0.7	0.7	0.7	0.7	0.71	0.71	0.71	0.71	0.71	0.71	0.71	0.71
SVM	0.82	0.82	0.82	0.82	0.84	0.84	0.84	0.84	0.83	0.83	0.83	0.83
DecisionTree	0.81	0.83	0.81	0.82	0.80	0.82	0.80	0.80	0.81	0.82	0.81	0.81
	Feature Selection with N=75 , K=5											
	Regression				**ANOVA**				**MI**			
	accuracy	precision	recall	f1-score	Accuracy	precision	recall	f1-score	accuracy	precision	recall	f1-score
RandomForest	**0.90**	**0.90**	**0.90**	**0.90**	**0.89**	**0.89**	**0.89**	**0.89**	**0.89**	**0.89**	**0.89**	**0.89**
GaussianNB	0.72	0.72	0.72	0.72	0.71	0.71	0.71	0.71	0.71	0.71	0.71	0.71

SVM	0.83	0.83	0.83	0.83	0.83	0.83	0.83	0.83	0.83	0.83	0.83	0.83
DecisionTree	0.81	0.83	0.81	0.82	0.81	0.83	0.81	0.82	0.81	0.83	0.81	0.81
	Feature Selection with N=75 , K=4											
	Regression				**ANOVA**				**MI**			
	accuracy	precision	recall	f1-score	Accuracy	precision	recall	f1-score	accuracy	precision	recall	f1-score
RandomForest	**0.89**	**0.89**	**0.89**	**0.89**	**0.9**	**0.9**	**0.9**	**0.9**	**0.89**	**0.89**	**0.89**	**0.89**
GaussianNB	0.71	0.71	0.71	0.71	0.7	0.7	0.7	0.7	0.71	0.71	0.71	0.71
SVM	0.83	0.83	0.83	0.83	0.83	0.83	0.83	0.83	0.83	0.83	0.83	0.83
DecisionTree	0.82	0.83	0.82	0.82	0.81	0.82	0.81	0.81	0.81	0.83	0.81	0.81
	Feature Selection with N=75 , K=3											
	Regression				**ANOVA**				**MI**			
	accuracy	precision	recall	f1-score	Accuracy	precision	recall	f1-score	accuracy	precision	recall	f1-score
RandomForest	**0.9**	**0.9**	**0.9**	**0.9**	**0.88**	**0.88**	**0.88**	**0.88**	**0.89**	**0.89**	**0.89**	**0.89**
GaussianNB	0.72	0.72	0.72	0.72	0.69	0.69	0.69	0.69	0.71	0.71	0.71	0.71
SVM	0.85	0.85	0.85	0.85	0.82	0.82	0.82	0.82	0.82	0.82	0.82	0.82
DecisionTree	0.83	0.84	0.83	0.83	0.8	0.82	0.8	0.8	0.82	0.83	0.82	0.82

Table 4.4 shows the results just like in Table 4.3. But for Table 4.4 it shows the results with 75% features selected. For this test, the highest accuracy is 90%, and the lowest accuracy is 88%. Which is again not that big difference. However, it showed 1% higher accuracy than the previous table.

Table 4.5: results of Machine Learning models with 50% features selected, and 80%-20% dataset split

	Feature Selection with N=50 , K=8											
	Regression				**ANOVA**				**MI**			
Methods	accuracy	precision	recall	f1-score	Accuracy	precision	recall	f1-score	accuracy	precision	recall	f1-score
RandomForest	**0.88**	**0.88**	**0.88**	**0.88**	**0.89**	**0.89**	**0.89**	**0.89**	**0.89**	**0.89**	**0.89**	**0.89**
GaussianNB	0.71	0.71	0.71	0.71	0.7	0.7	0.7	0.7	0.71	0.71	0.71	0.71
SVM	0.82	0.82	0.82	0.82	0.82	0.82	0.82	0.82	0.82	0.82	0.82	0.82
DecisionTree	0.82	0.83	0.82	0.82	0.81	0.83	0.81	0.82	0.83	0.84	0.83	0.83
	Feature Selection with N=50 , K=7											
	Regression				**ANOVA**				**MI**			
	accuracy	precision	recall	f1-score	Accuracy	precision	recall	f1-score	accuracy	precision	recall	f1-score
RandomForest	**0.89**	**0.89**	**0.89**	**0.89**	**0.88**	**0.88**	**0.88**	**0.88**	**0.89**	**0.89**	**0.89**	**0.89**
GaussianNB	0.71	0.71	0.71	0.71	0.71	0.71	0.71	0.71	0.7	0.7	0.7	0.7
SVM	0.81	0.81	0.81	0.81	0.82	0.82	0.82	0.82	0.83	0.83	0.83	0.83
DecisionTree	0.82	0.83	0.82	0.82	0.82	0.83	0.82	0.82	0.82	0.83	0.82	0.82
	Feature Selection with N=50 , K=6											
	Regression				**ANOVA**				**MI**			
	accuracy	precision	recall	f1-score	Accuracy	precision	recall	f1-score	accuracy	precision	recall	f1-score
RandomForest	**0.88**	**0.88**	**0.88**	**0.88**	**0.89**	**0.89**	**0.89**	**0.89**	**0.88**	**0.89**	**0.88**	**0.88**
GaussianNB	0.71	0.71	0.71	0.71	0.72	0.72	0.72	0.72	0.71	0.71	0.71	0.71
SVM	0.81	0.81	0.81	0.81	0.83	0.83	0.83	0.82	0.82	0.82	0.82	0.82
DecisionTree	0.81	0.82	0.81	0.81	0.82	0.83	0.82	0.82	0.82	0.84	0.82	0.82
	Feature Selection with N=50 , K=5											
	Regression				**ANOVA**				**MI**			
	accuracy	precision	recall	f1-score	Accuracy	precision	recall	f1-score	accuracy	precision	recall	f1-score
RandomForest	**0.89**	**0.89**	**0.89**	**0.89**	**0.89**	**0.89**	**0.89**	**0.89**	**0.88**	**0.88**	**0.88**	**0.88**

GaussianNB	0.7	0.7	0.7	0.7	0.69	0.69	0.69	0.69	0.7	0.7	0.7	0.7
SVM	0.82	0.82	0.82	0.82	0.82	0.82	0.82	0.82	0.82	0.82	0.82	0.82
DecisionTree	0.83	0.84	0.83	0.83	0.81	0.82	0.81	0.81	0.82	0.83	0.82	0.82
	Feature Selection with N=50 , K=4											
	Regression				**ANOVA**				**MI**			
	accuracy	precision	recall	f1-score	Accuracy	precision	recall	f1-score	accuracy	precision	recall	f1-score
RandomForest	**0.9**	**0.9**	**0.9**	**0.9**	**0.89**	**0.89**	**0.89**	**0.89**	**0.89**	**0.89**	**0.89**	**0.89**
GaussianNB	0.71	0.71	0.71	0.71	0.7	0.7	0.7	0.7	0.71	0.71	0.71	0.71
SVM	0.82	0.82	0.82	0.82	0.82	0.82	0.82	0.82	0.82	0.82	0.82	0.82
DecisionTree	0.82	0.84	0.82	0.82	0.82	0.83	0.82	0.82	0.82	0.83	0.82	0.82
	Feature Selection with N=50 , K=3											
	Regression				**ANOVA**				**MI**			
	accuracy	precision	recall	f1-score	Accuracy	precision	recall	f1-score	accuracy	precision	recall	f1-score
RandomForest	**0.89**	**0.89**	**0.89**	**0.89**	**0.9**	**0.9**	**0.9**	**0.9**	**0.88**	**0.88**	**0.88**	**0.88**
GaussianNB	0.69	0.69	0.69	0.69	0.7	0.7	0.7	0.7	0.71	0.71	0.71	0.71
SVM	0.81	0.81	0.81	0.81	0.82	0.82	0.82	0.82	0.82	0.82	0.82	0.82
DecisionTree	0.8	0.81	0.8	0.8	0.81	0.83	0.81	0.81	0.81	0.83	0.81	0.81

Table 4.5 shows the Machine Learning models with 50% features selected, and 3 to 8 k-mer size. Generally, its performance is not big different from the previous two tables. Random Forrest obtained the highest values for all accuracy metrics. The highest accuracy obtained is 90%, and the lowest is 88%.

From the previous testing accuracies, we can conclude that the best performing algorithm is obtained by setting the k-mer size to 4, and number of features to 75%. The Regression feature selection approach achieved 90% accuracy, 91% precession, 89% recall, and 90% F1-score. While the ANOVA feature selection approach achieved 90% accuracy, 89% precision, 91% recall, and 90% F1-score.

Notice that, with the 20% features selected, the best accuracy obtained is 89%. However, when the features selected are 50% and 75%, the best accuracy obtained is 90%. Indicating that selecting only 20% of features is not sufficient for good classification.

Furthermore, with the 50% and 75% features selected, there are multiple k-mer sizes obtained the 90% accuracy. In the 50% features selected, k-mer sizes of 3 and 4 obtained 90% accuracy. On the other hand, the 75% features selected showed the k-mer sizes of 3,4,5, and 8. Indicating that more features selected, leads to more high-performance approaches.

Another note is for both the 50% and 75% features selected, the k-mer sizes of 3 and 4 obtained the highest accuracy in both. This is because the sizes of 3 and 4 will produce the highest numbers of words. Larger number of words leads to more flexible pattern recognition of the DNA sequence.

4.6.2 Second Data Split

In this subsection the results of the dataset split into 70% training, and 30% testing will be explained.

Table 4.6: results of Machine Learning models with 20% features selected, and 70%-30% dataset split

	Feature Selection with N=20 , K=8											
	Regression				**ANOVA**				**MI**			
Methods	Accuracy	precision	Recall	f1-score	Accuracy	precision	recall	f1-score	accuracy	precision	recall	f1-score
RandomForest	**0.87**	**0.87**	**0.87**	**0.87**	**0.85**	**0.86**	**0.85**	**0.85**	**0.86**	**0.86**	**0.86**	**0.86**
GaussianNB	0.69	0.69	0.69	0.69	0.67	0.67	0.67	0.67	0.70	0.70	0.70	0.70
SVM	0.77	0.77	0.77	0.77	0.75	0.75	0.75	0.75	0.76	0.76	0.76	0.76
DecisionTree	0.80	0.82	0.80	0.81	0.79	0.81	0.79	0.79	0.80	0.81	0.80	0.80
	Feature Selection with N=20 , K=7											
	Regression				**ANOVA**				**MI**			
	accuracy	precision	Recall	f1-score	Accuracy	precision	recall	f1-score	accuracy	precision	recall	f1-score
RandomForest	**0.85**	**0.86**	**0.85**	**0.85**	**0.86**	**0.86**	**0.86**	**0.86**	**0.86**	**0.86**	**0.86**	**0.86**
GaussianNB	0.68	0.69	0.68	0.68	0.68	0.69	0.68	0.68	0.68	0.68	0.68	0.68
SVM	0.77	0.77	0.77	0.77	0.77	0.77	0.77	0.77	0.76	0.76	0.76	0.76
DecisionTree	0.80	0.81	0.80	0.80	0.80	0.82	0.80	0.80	0.80	0.81	0.80	0.80
	Feature Selection with N=20 , K=6											
	Regression				**ANOVA**				**MI**			
	accuracy	precision	Recall	f1-score	Accuracy	precision	recall	f1-score	accuracy	precision	recall	f1-score
RandomForest	**0.87**	**0.87**	**0.87**	**0.87**	**0.86**	**0.87**	**0.86**	**0.86**	**0.87**	**0.87**	**0.87**	**0.87**
GaussianNB	0.70	0.70	0.70	0.70	0.68	0.68	0.68	0.68	0.70	0.70	0.70	0.70
SVM	0.78	0.78	0.78	0.78	0.76	0.76	0.76	0.76	0.77	0.77	0.77	0.77
DecisionTree	0.80	0.82	0.80	0.80	0.79	0.81	0.79	0.79	0.80	0.81	0.80	0.80
	Feature Selection with N=20 , K=5											
	Regression				**ANOVA**				**MI**			

	accuracy	precision	Recall	f1-score	Accuracy	precision	recall	f1-score	accuracy	precision	recall	f1-score
RandomForest	**0.86**	**0.86**	**0.86**	**0.86**	**0.86**	**0.86**	**0.86**	**0.86**	**0.87**	**0.87**	**0.87**	**0.87**
GaussianNB	0.70	0.70	0.70	0.70	0.67	0.68	0.67	0.68	0.69	0.70	0.69	0.69
SVM	0.77	0.77	0.77	0.77	0.75	0.75	0.75	0.75	0.78	0.78	0.78	0.78
DecisionTree	0.80	0.81	0.80	0.80	0.79	0.80	0.79	0.79	0.80	0.81	0.80	0.80
	Feature Selection with N=20 , K=4											
	Regression				**ANOVA**				**MI**			
	accuracy	precision	Recall	f1-score	Accuracy	precision	recall	f1-score	accuracy	precision	recall	f1-score
RandomForest	**0.87**	**0.87**	**0.87**	**0.87**	**0.87**	**0.87**	**0.87**	**0.87**	**0.87**	**0.87**	**0.87**	**0.87**
GaussianNB	0.69	0.69	0.69	0.69	0.68	0.68	0.68	0.68	0.69	0.69	0.69	0.69
SVM	0.78	0.78	0.78	0.78	0.77	0.77	0.77	0.77	0.77	0.77	0.77	0.77
DecisionTree	0.80	0.81	0.80	0.80	0.79	0.81	0.79	0.80	0.80	0.81	0.80	0.80
	Feature Selection with N=20 , K=3											
	Regression				**ANOVA**				**MI**			
	accuracy	precision	Recall	f1-score	Accuracy	precision	recall	f1-score	accuracy	precision	recall	f1-score
RandomForest	**0.87**	**0.87**	**0.87**	**0.87**	**0.87**	**0.87**	**0.87**	**0.87**	**0.87**	0.87	0.87	0.87
GaussianNB	0.69	0.69	0.69	0.69	0.69	0.69	0.69	0.69	0.70	0.70	0.70	0.70
SVM	0.77	0.77	0.77	0.77	0.77	0.77	0.77	0.77	0.77	0.77	0.77	0.77
DecisionTree	0.80	0.81	0.80	0.80	0.79	0.81	0.79	0.79	0.81	0.82	0.81	0.81

Table 4.6 shows the Machine Learning models with 20%features selected, and 3 to 8 k-mer size. Random Forest still dominating the lead with the highest accuracy. However, it showed less accuracy level in general. The highest accuracy level is 87%, and the lowest is 85%.

Table 4.7: results of Machine Learning models with 75% features selected, and 70%-30% dataset split

	Feature Selection with N=75 , K=8											
	FR				**FOA**				**MI**			
Methods	accuracy	precision	recall	f1-score	Accuracy	precision	recall	f1-score	accuracy	precision	recall	f1-score
RandomForest	**0.88**	**0.88**	**0.88**	**0.88**	**0.87**	**0.87**	**0.87**	**0.87**	**0.87**	**0.87**	**0.87**	**0.87**
GaussianNB	0.71	0.72	0.71	0.71	0.71	0.71	0.71	0.71	0.71	0.71	0.71	0.71
SVM	0.82	0.82	0.82	0.82	0.82	0.82	0.82	0.82	0.82	0.82	0.82	0.82
DecisionTree	0.80	0.82	0.80	0.81	0.80	0.82	0.80	0.80	0.80	0.81	0.80	0.80
	Feature Selection with N=75 , K=7											
	FR				**FOA**				**MI**			
	accuracy	precision	recall	f1-score	Accuracy	precision	recall	f1-score	accuracy	precision	recall	f1-score
RandomForest	**0.87**	**0.87**	**0.87**	**0.87**	**0.86**	**0.86**	**0.86**	**0.86**	**0.87**	**0.87**	**0.87**	**0.87**
GaussianNB	0.71	0.71	0.71	0.71	0.71	0.71	0.71	0.71	0.71	0.71	0.71	0.71
SVM	0.82	0.82	0.82	0.82	0.81	0.81	0.81	0.81	0.81	0.81	0.81	0.81
DecisionTree	0.79	0.80	0.79	0.80	0.79	0.80	0.79	0.79	0.79	0.81	0.79	0.80
	Feature Selection with N=75 , K=6											
	FR				**FOA**				**MI**			
	accuracy	precision	recall	f1-score	Accuracy	precision	recall	f1-score	accuracy	precision	recall	f1-score
RandomForest	**0.89**	**0.89**	**0.89**	**0.89**	**0.87**	**0.87**	**0.87**	**0.87**	**0.87**	**0.87**	**0.87**	**0.87**
GaussianNB	0.71	0.71	0.71	0.71	0.71	0.71	0.71	0.71	0.72	0.72	0.72	0.72
SVM	0.82	0.82	0.82	0.82	0.82	0.82	0.82	0.82	0.82	0.82	0.82	0.82
DecisionTree	0.80	0.81	0.80	0.80	0.80	0.81	0.80	0.80	0.80	0.81	0.80	0.80
	Feature Selection with N=75 , K=5											
	FR				**FOA**				**MI**			

	accuracy	precision	recall	f1-score	Accuracy	precision	recall	f1-score	accuracy	precision	recall	f1-score
RandomForest	**0.87**	**0.87**	**0.87**	**0.87**	**0.88**	**0.88**	**0.88**	**0.88**	**0.87**	**0.87**	**0.87**	**0.87**
GaussianNB	0.71	0.71	0.71	0.71	0.70	0.70	0.70	0.70	0.71	0.71	0.71	0.71
SVM	0.81	0.81	0.81	0.81	0.81	0.81	0.81	0.81	0.81	0.81	0.81	0.81
DecisionTree	0.79	0.80	0.79	0.79	0.80	0.81	0.80	0.80	0.81	0.83	0.81	0.81
	Feature Selection with N=75 , K=4											
	FR				**FOA**				**MI**			
	accuracy	precision	recall	f1-score	Accuracy	precision	recall	f1-score	accuracy	precision	recall	f1-score
RandomForest	**0.88**	**0.88**	**0.88**	**0.88**	**0.88**	**0.88**	**0.88**	**0.88**	**0.87**	**0.87**	**0.87**	**0.87**
GaussianNB	0.71	0.71	0.71	0.71	0.71	0.71	0.71	0.71	0.71	0.71	0.71	0.71
SVM	0.82	0.82	0.82	0.82	0.82	0.82	0.82	0.82	0.82	0.82	0.82	0.82
DecisionTree	0.80	0.81	0.80	0.80	0.78	0.80	0.78	0.79	0.80	0.81	0.80	0.80
	Feature Selection with N=75 , K=3											
	FR				**FOA**				**MI**			
	accuracy	precision	recall	f1-score	Accuracy	precision	recall	f1-score	accuracy	precision	recall	f1-score
RandomForest	**0.88**	**0.88**	**0.88**	**0.88**	**0.87**	**0.87**	**0.87**	**0.87**	**0.88**	**0.88**	**0.88**	**0.88**
GaussianNB	0.72	0.72	0.72	0.72	0.70	0.70	0.70	0.70	0.72	0.72	0.72	0.72
SVM	0.82	0.82	0.82	0.82	0.81	0.81	0.81	0.81	0.82	0.82	0.82	0.82
DecisionTree	0.80	0.82	0.80	0.80	0.80	0.81	0.80	0.80	0.80	0.81	0.80	0.80

Similarly, Table 4.7 display the results of the four models by using the three feature selection approaches, by using 75% feature selection. And k-mer sizes from 3-8. The results show the superiority of Random Forest algorithm comparing to other Machine Learning models. The highest accuracy value obtained by Random Forest is 89%. While the lowest accuracy value is 87%.

Table 4.8: results of Machine Learning models with 50% features selected, and 70%-30% dataset split

	Feature Selection with N=50 , K=8											
	FR				**FOA**				**MI**			
Methods	accuracy	precision	recall	f1-score	accuracy	precision	recall	f1-score	Accuracy	precision	recall	f1-score
RandomForest	**0.87**	**0.87**	**0.87**	**0.87**	**0.86**	**0.87**	**0.86**	**0.86**	**0.86**	**0.86**	**0.86**	**0.86**
GaussianNB	0.71	0.71	0.71	0.71	0.70	0.70	0.70	0.70	0.71	0.71	0.71	0.71
SVM	0.81	0.81	0.81	0.81	0.81	0.81	0.81	0.81	0.81	0.81	0.81	0.81
DecisionTree	0.78	0.79	0.78	0.78	0.80	0.82	0.80	0.81	0.80	0.82	0.80	0.80
	Feature Selection with N=50 , K=7											
	FR				**FOA**				**MI**			
	accuracy	precision	recall	f1-score	accuracy	precision	recall	f1-score	Accuracy	precision	recall	f1-score
RandomForest	**0.87**	**0.87**	**0.87**	**0.87**	**0.88**	**0.88**	**0.88**	**0.88**	**0.86**	**0.87**	**0.86**	**0.86**
GaussianNB	0.71	0.71	0.71	0.71	0.70	0.70	0.70	0.70	0.71	0.71	0.71	0.71
SVM	0.81	0.81	0.81	0.81	0.80	0.80	0.80	0.80	0.82	0.82	0.82	0.82
DecisionTree	0.80	0.81	0.80	0.80	0.80	0.82	0.80	0.80	0.80	0.82	0.80	0.80
	Feature Selection with N=50 , K=6											
	FR				**FOA**				**MI**			
	accuracy	precision	recall	f1-score	accuracy	precision	recall	f1-score	Accuracy	precision	recall	f1-score
RandomForest	**0.87**	**0.87**	**0.87**	**0.87**	**0.86**	**0.87**	**0.86**	**0.86**	**0.87**	**0.87**	**0.87**	**0.87**
GaussianNB	0.71	0.71	0.71	0.71	0.70	0.70	0.70	0.70	0.70	0.70	0.70	0.70
SVM	0.80	0.80	0.80	0.80	0.79	0.79	0.79	0.79	0.80	0.80	0.80	0.80
DecisionTree	0.80	0.81	0.80	0.80	0.79	0.80	0.79	0.79	0.79	0.80	0.79	0.79
	Feature Selection with N=50 , K=5											
	FR				**FOA**				**MI**			

	accuracy	precision	recall	f1-score	accuracy	precision	recall	f1-score	Accuracy	precision	recall	f1-score
RandomForest	**0.86**	**0.87**	**0.86**	**0.86**	**0.87**	**0.87**	**0.87**	**0.87**	**0.88**	**0.88**	**0.88**	**0.88**
GaussianNB	0.70	0.70	0.70	0.70	0.71	0.71	0.71	0.71	0.71	0.71	0.71	0.71
SVM	0.80	0.80	0.80	0.80	0.80	0.80	0.80	0.80	0.82	0.82	0.82	0.82
DecisionTree	0.80	0.82	0.80	0.80	0.79	0.80	0.79	0.79	0.80	0.81	0.80	0.80
	Feature Selection with N=50 , K=4											
	FR				**FOA**				**MI**			
	accuracy	precision	recall	f1-score	accuracy	precision	recall	f1-score	Accuracy	precision	recall	f1-score
RandomForest	**0.87**	**0.87**	**0.87**	**0.87**	**0.87**	**0.87**	**0.87**	**0.87**	**0.86**	**0.86**	**0.86**	**0.86**
GaussianNB	0.71	0.71	0.71	0.71	0.70	0.70	0.70	0.70	0.70	0.70	0.70	0.70
SVM	0.81	0.81	0.81	0.81	0.80	0.80	0.80	0.80	0.80	0.80	0.80	0.80
DecisionTree	0.80	0.81	0.80	0.80	0.79	0.81	0.79	0.79	0.79	0.81	0.79	0.79
	Feature Selection with N=50 , K=3											
	FR				**FOA**				**MI**			
	accuracy	precision	recall	f1-score	accuracy	precision	recall	f1-score	Accuracy	precision	recall	f1-score
RandomForest	**0.87**	**0.87**	**0.87**	**0.87**	**0.87**	**0.87**	**0.87**	**0.87**	**0.87**	**0.87**	**0.87**	**0.87**
GaussianNB	0.70	0.70	0.70	0.70	0.69	0.70	0.69	0.69	0.72	0.72	0.72	0.72
SVM	0.81	0.81	0.81	0.81	0.81	0.81	0.81	0.81	0.80	0.80	0.80	0.80
DecisionTree	0.80	0.81	0.80	0.80	0.80	0.81	0.80	0.80	0.80	0.81	0.80	0.80

Table 4.8 display the results of the four models by using the three feature selection approaches, by using 50% feature selection. And k-mer sizes from 3-8. Similar to the previous results, the Random Forest shows higher performance comparing with the other used models. Higher results obtained 88%, and the lowest is 86%.

Notice that the performance of 70%-30% dataset split performed worse than the 80%-20% dataset split. This is because of the insufficient amount of data used in the training process in the 70%-30% dataset split. Leading to poor classification performance in the traditional Machine Learning algorithms.

Another observation from the obtained results is that the three used feature selection approaches (MI, regression, and ANOVA) gives similar accuracies when using their outputs to similar algorithms.

4.7 Deep Learning Results

The Deep Learning model used in this work is consisting of CNN and LSTM. This is due to the sequence nature of the used dataset.

The Deep Learning model is trained only by using 70% training – 30% testing data split. And the percentage of the selected features are 20% and 50%. And the k-mer sizes are from 3-8.

Table 4.9: results of Deep Learning models with 20% features selected, and 70%-30% dataset split

Feature selection with N=20 and K=3				
Feature method	**Accuracy**	**Precision**	**Recall**	**F1-Score**
One-way ANOVA	0.99	0.99	0.99	0.99
F-regressor	0.99	0.99	0.99	0.99
Mutual Information	0.94	0.94	0.94	0.94
Feature selection with N=20 and K=4				
Feature method	**Accuracy**	**Precision**	**Recall**	**F1-Score**
One-way ANOVA	0.98	0.98	0.98	0.98
F-regressor	0.96	0.96	0.96	0.96
Mutual Information	0.97	0.97	0.97	0.97
Feature selection with N=20 and K=5				

Feature method	Accuracy	Precision	Recall	F1-Score
One-way ANOVA	0.96	0.96	0.96	0.96
F-regressor	0.92	0.92	0.92	0.92
Mutual Information	0.99	0.99	0.99	0.99
Feature selection with N=20 and K=6				
Feature method	**Accuracy**	**Precision**	**Recall**	**F1-Score**
One-way ANOVA	0.97	0.97	0.97	0.97
F-regressor	1.00	1.00	1.00	1.00
Mutual Information	0.98	0.98	0.98	0.98
Feature selection with N=20 and K=7				
Feature method	**Accuracy**	**Precision**	**Recall**	**F1-Score**
One-way ANOVA	0.99	0.99	0.99	0.99
F-regressor	1.00	1.00	1.00	1.00
Mutual Information	0.91	0.91	0.91	0.91
Feature selection with N=20 and K=8				
Feature method	**Accuracy**	**Precision**	**Recall**	**F1-Score**
One-way ANOVA	1.00	1.00	1.00	1.00
F-regressor	0.99	0.99	0.99	0.99
Mutual Information	0.99	0.99	0.99	0.99

Table 4.9 shows the results obtained by Deep Learning model. The percentage of used feature selection is 20%. You can notice how the performance is much higher comparing with the Machine Learning model. The highest performance obtained in this test, is 100% for all accuracy, precision, recall, and F1-score. These results achieved by using k-mer size of 6 and 7 with Regression feature selection. And 8 k-mer size, with One Way Anova feature selection approach.

Figures from 4.5 to 4.10 shows a comparison between the results obtained from Deep Learning model, with various feature selection approaches.

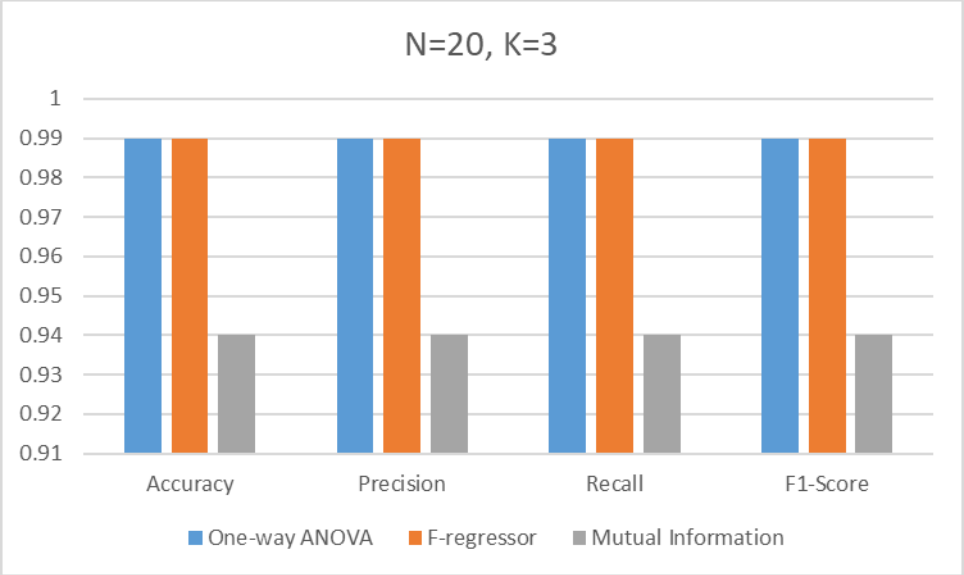

Figure 4.5: comparison between the results obtained from Deep Learning model using various Feature Selection approaches, with N=20 and K=3.

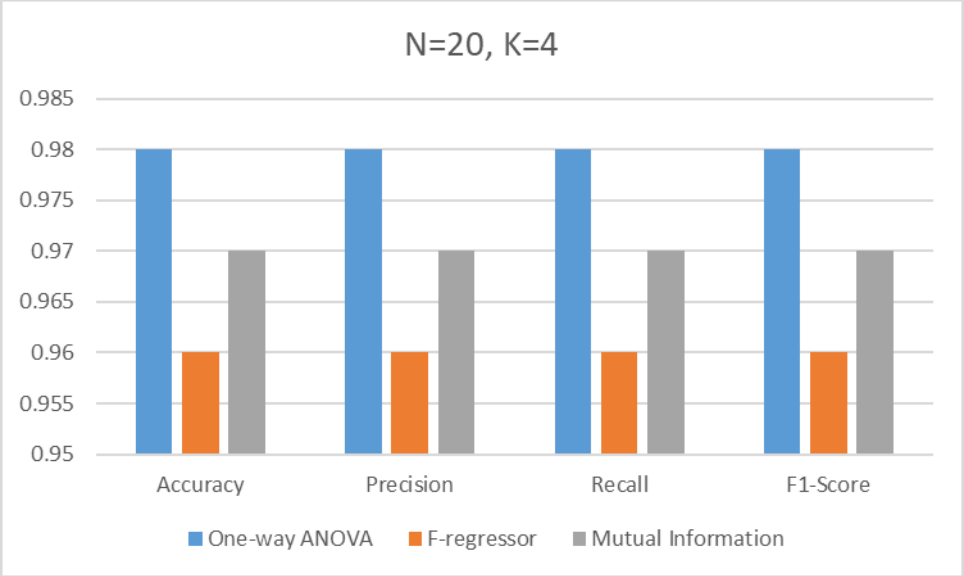

Figure 4.6: comparison between the results obtained from Deep Learning model using various Feature Selection approaches, with N=20 and K=4.

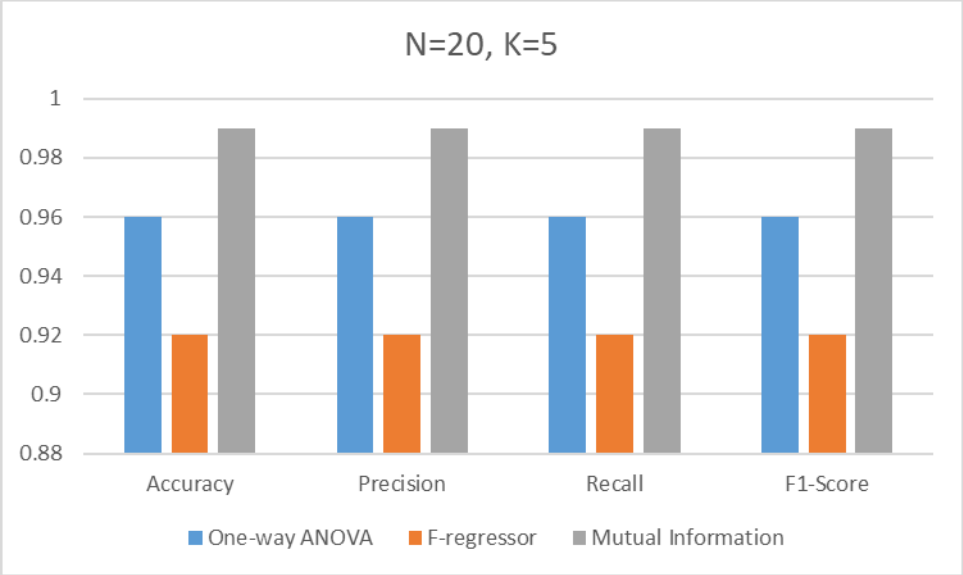

Figure 4.7: comparison between the results obtained from Deep Learning model using various Feature Selection approaches, with N=20 and K=5.

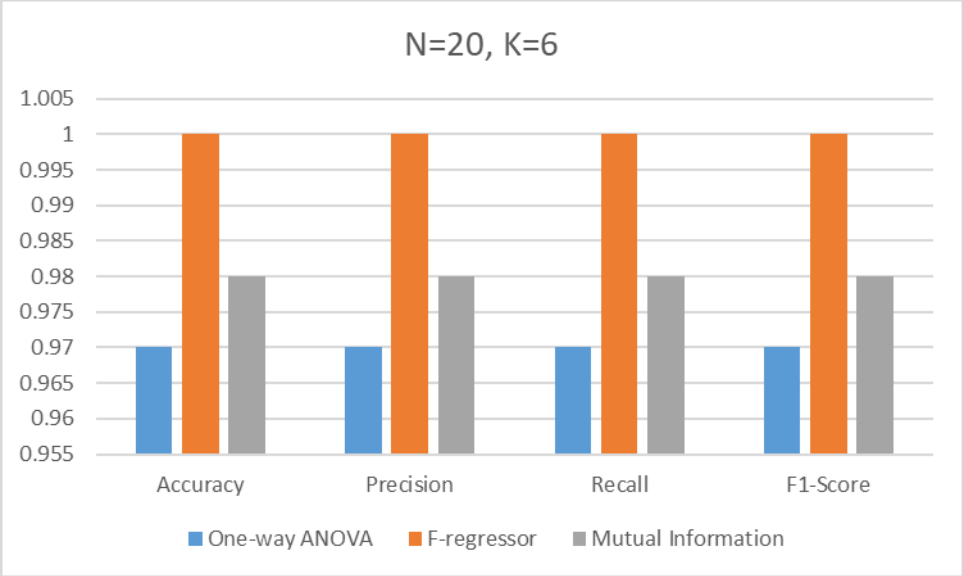

Figure 4.8: comparison between the results obtained from Deep Learning model using various Feature Selection approaches, with N=20 and K=6.

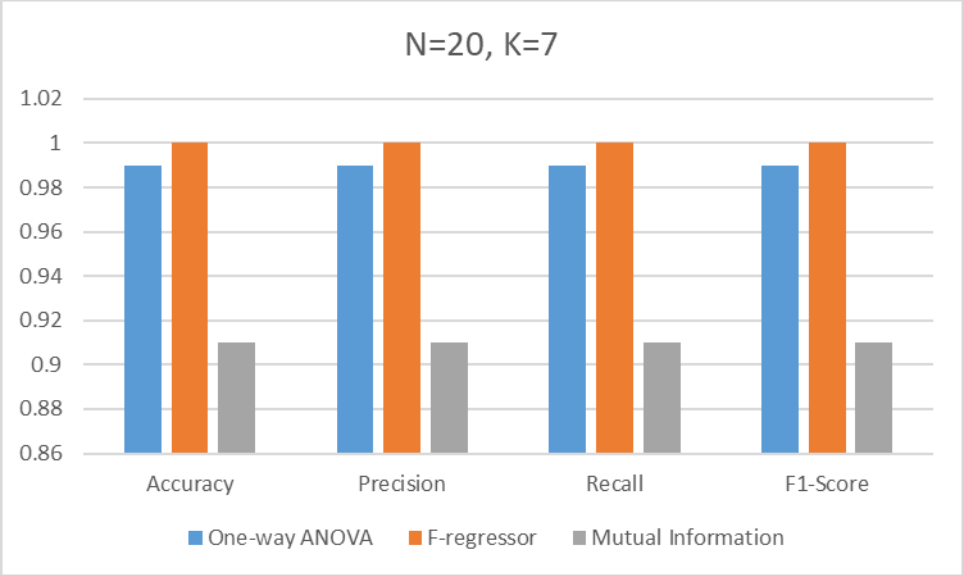

Figure 4.9: comparison between the results obtained from Deep Learning model using various Feature Selection approaches, with N=20 and K=7.

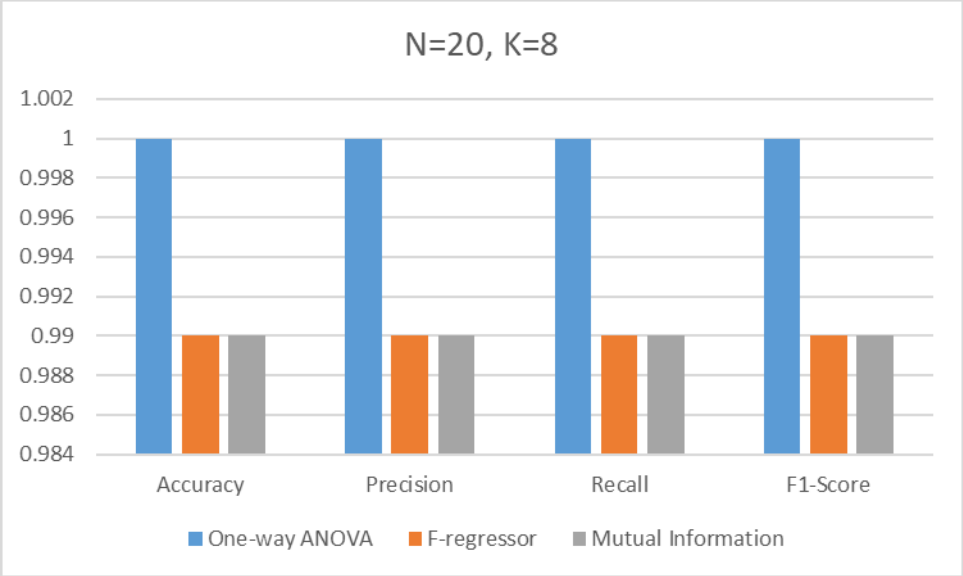

Figure 4.10: comparison between the results obtained from Deep Learning model using various Feature Selection approaches, with N=20 and K=8.

Table 4.10: results of Deep Learning models with 50% features selected, and 70%-30% dataset split

Feature selection with N=50 and K=3				
Feature method	Accuracy	Precision	Recall	F1-Score

One-way ANOVA	1.00	1.00	1.00	1.00
F-regressor	1.00	1.00	1.00	1.00
Mutual Information	1.00	1.00	1.00	1.00
Feature selection with N=50 and K=4				
Feature method	Accuracy	Precision	Recall	F1-Score
One-way ANOVA	1.00	1.00	1.00	1.00
F-regressor	1.00	1.00	1.00	1.00
Mutual Information	1.00	1.00	1.00	1.00
Feature selection with N=50 and K=5				
Feature method	Accuracy	Precision	Recall	F1-Score
One-way ANOVA	1.00	1.00	1.00	1.00
F-regressor	1.00	1.00	1.00	1.00
Mutual Information	1.00	1.00	1.00	1.00
Feature selection with N=50 and K=6				
Feature method	Accuracy	Precision	Recall	F1-Score
One-way ANOVA	1.00	1.00	1.00	1.00
F-regressor	1.00	1.00	1.00	1.00
Mutual Information	1.00	1.00	1.00	1.00
Feature selection with N=50 and K=7				
Feature method	Accuracy	Precision	Recall	F1-Score
One-way ANOVA	1.00	1.00	1.00	1.00
F-regressor	1.00	1.00	1.00	1.00
Mutual Information	1.00	1.00	1.00	1.00
Feature selection with N=50 and K=8				
Feature method	Accuracy	Precision	Recall	F1-Score
One-way ANOVA	1.00	1.00	1.00	1.00
F-regressor	1.00	1.00	1.00	1.00
Mutual Information	1.00	1.00	1.00	1.00

Table 4.10 shows the results obtained by Deep Learning model. The percentage of used feature selection is 50%. With this approach all the configurations gained 100% accuracy, precision, recall, and F1-score.

Deep Learning performance is much higher than traditional Machine Learning algorithms, since the number of trainable parameters (i.e. weights) in Deep Learning is much higher. Leading it to better recognizing more complex patterns, comparing to traditional Machine Learning algorithms.

Furthermore, the time consumed by the DL model to predict one sample of data is 0.328 milliseconds (note that 1 seconds = 1000 milliseconds) by using the hardware and software specified in section 4.2. which is considered very fast performance in terms of time complexity.

Overall, the whole testing dataset (6120 samples) took the model 2008 milliseconds to predict. With 100% accuracy.

4.8 Result Comparison

Table 4.11 shows comparison between the results of the proposed work and other related works. Notice that no other publication worked on the dataset used in this thesis. Therefore, the comparison will be based on other works that targeted DM patients, with DNA sequence data.

Table 4.11: Results comparison with other similar works

Work	Year	Dataset	Proposed approach	Results
[13]	2022	DNA sequence. 96 patients with T2DM, and 71 healthy person (control group)	Random Forests, SVM, Ridge Logistic Regression and Decision Trees	AUC 0.927
[15]	2022	DNA sequence. 300 human insulin sequence are selected from GenBank website.	CNN-LSTM	Accuracy is 99%, 97.5%, and 95% for LSTM-CNN, CNN, and LSTM, respectively.
[15]	2022	RNA sequencing data from the GEO (Transcript Expression Omnibus) database	Monte Carlo feature selection, support vector machine, and repeated incremental pruning	Identified T2D-associated genes (MTND4P24, MTND2P28, and LOC100128906)
[9]	2018	Gene Expression. Database of Genotypes and Phenotypes (dbGaP)	Random Forest	AUC: 73.96%, Sensitivity: 68.42%, and specificity: 78.67%
[10]	2021	Genetic Expression. 177 diabetics persons with 173 control group get	Multilayer Perceptron (MLP), K-star (K*)	MLP: 75% accuracy. K*: 73% accuracy.

		their data genome data.		
[11]	2020	Gene Expression. Total of 164 people. 82 diabetics and 82 healthy people.	Feed-forward neural network	Accuracy: 88%
Proposed work	2023	DNA sequence. From National Library of Medicine. Contains the DNA sequence of 14571 people. 10199 DNA sequences are labelled diabetes, and 4371 are labelled non diabetes.	CNN-LSTM	Accuracy 100%.

Table 4.11 shows that the proposed work achieved higher accuracy than other proposed system. However notice that the used dataset is used only in this work.

Given the experiments conducted in this thesis, with the results from other researches as shown in the previous table, the final conclusion would be the best performing model is the proposed CNN-LSTM model, with 70%-30% training-testing dataset split. With any k-mer size and any feature selection approach.

Chapter Five

Conclusions and Future Works

5.1 Conclusion

In conclusion, the research and system development steps and techniques of this thesis could be summarized into the following:

1. The pre-processing and feature selection steps are crucial and vital in determination of the efficient of ML and DL models.
2. Deep Learning showed higher performance comparing with other Machine Learning approaches.
3. This thesis bridges the gap between the fields of computer science and genetics, providing novel insights that may change the face of genetic illness prediction in the future. The insights and pipeline created have the potential to advance precision medicine approaches and have ramifications for other hereditary illnesses.
4. This thesis uses a comprehensive evaluation framework to compare several machine learning and deep learning models for DM prediction accuracy.

5.2 Future works

The future works could be summarized in the following points:

1. Expand the dataset by using additional DNA sequences for healthy and diabetes patients.
2. Try combined feature selection techniques with the Machine Learning models, instead of the single feature selection method each time.
3. Use the models to classify further diseases. So, the classification will be able to detect the disease based on the DNA sequence. For example, cancer, blood pressure, and others.
4. Investigate the ensemble method, by using combined ML algorithms together.
5. Investigate using LSTM only to learn the pattern of DM in DNA sequence.

References

References

[1] N. G. Forouhi and N. J. Wareham, "Epidemiology of diabetes," *Medicine*, vol. 47, no. 1, pp. 22–27, Jan. 2019, doi: 10.1016/j.mpmed.2018.10.004.

[2] M. V. C. Greenberg and D. Bourc'his, "The diverse roles of DNA methylation in mammalian development and disease," *Nat Rev Mol Cell Biol*, vol. 20, no. 10, pp. 590–607, Oct. 2019, doi: 10.1038/s41580-019-0159-6.

[3] P. Rorsman and F. M. Ashcroft, "Pancreatic β-Cell Electrical Activity and Insulin Secretion: Of Mice and Men," *Physiol Rev*, vol. 98, no. 1, pp. 117–214, Jan. 2018, doi: 10.1152/physrev.00008.2017.

[4] J. Kierstein, "Introduction to Genetics," in *Nutrition Management of Inherited Metabolic Diseases*, Cham: Springer International Publishing, 2022, pp. 3–22. doi: 10.1007/978-3-030-94510-7_1.

[5] M. Alehegn, R. Joshi, and P. Mulay, "Analysis and Prediction of Diabetes Mellitus using Machine Learning Algorithm," *International Journal of Pure and Applied Mathematics*, vol. 118, no. 9, pp. 871–877, 2018.

[6] B. Abdulaimma *et al.*, "Improving Type 2 Diabetes Phenotypic Classification by Combining Genetics and Conventional Risk Factors," in *2018 IEEE Congress on Evolutionary Computation (CEC)*, IEEE, Jul. 2018, pp. 1–7. doi: 10.1109/CEC.2018.8477647.

[7] H. Zhou, R. Myrzashova, and R. Zheng, "Diabetes prediction model based on an enhanced deep neural network," *EURASIP J Wirel Commun Netw*, vol. 2020, no. 1, p. 148, Dec. 2020, doi: 10.1186/s13638-020-01765-7.

[8] M. M. Hatmal *et al.*, "Artificial Neural Networks Model for Predicting Type 2 Diabetes Mellitus Based on VDR Gene FokI Polymorphism, Lipid Profile and Demographic Data," *Biology (Basel)*, vol. 9, no. 8, p. 222, Aug. 2020, doi: 10.3390/biology9080222.

[9] M. M. Hatmal *et al.*, "Investigating the association of CD36 gene polymorphisms (rs1761667 and rs1527483) with T2DM and dyslipidemia: Statistical analysis, machine learning based prediction, and meta-analysis," *PLoS One*, vol. 16, no. 10, p. e0257857, Oct. 2021, doi: 10.1371/journal.pone.0257857.

[10] M. Muneeb and A. Henschel, "Eye-color and Type-2 diabetes phenotype prediction from genotype data using deep learning methods," *BMC*

Bioinformatics, vol. 22, no. 1, p. 198, Dec. 2021, doi: 10.1186/s12859-021-04077-9.

[11] H. Gunasekaran, K. Ramalakshmi, A. Rex Macedo Arokiaraj, S. Deepa Kanmani, C. Venkatesan, and C. Suresh Gnana Dhas, "Analysis of DNA Sequence Classification Using CNN and Hybrid Models," *Comput Math Methods Med*, vol. 2021, pp. 1–12, Jul. 2021, doi: 10.1155/2021/1835056.

[12] Y. Jian, M. Pasquier, A. Sagahyroon, and F. Aloul, "A Machine Learning Approach to Predicting Diabetes Complications," *Healthcare*, vol. 9, no. 12, p. 1712, Dec. 2021, doi: 10.3390/healthcare9121712.

[13] M. Karaglani *et al.*, "Liquid Biopsy in Type 2 Diabetes Mellitus Management: Building Specific Biosignatures via Machine Learning," *J Clin Med*, vol. 11, no. 4, p. 1045, Feb. 2022, doi: 10.3390/jcm11041045.

[14] J. Li, J. Ding, D. U. Zhi, K. Gu, and H. Wang, "Identification of Type 2 Diabetes Based on a Ten-Gene Biomarker Prediction Model Constructed Using a Support Vector Machine Algorithm," *Biomed Res Int*, vol. 2022, pp. 1–15, Mar. 2022, doi: 10.1155/2022/1230761.

[15] N. E. El-Attar, B. M. Moustafa, and W. A. Awad, "Deep Learning Model to Detect Diabetes Mellitus Based on DNA Sequence," *Intelligent Automation & Soft Computing*, vol. 31, no. 1, pp. 325–338, 2022, doi: 10.32604/iasc.2022.019970.

[16] Z. Li, X. Pan, and Y.-D. Cai, "Identification of Type 2 Diabetes Biomarkers From Mixed Single-Cell Sequencing Data With Feature Selection Methods," *Front Bioeng Biotechnol*, vol. 10, Jun. 2022, doi: 10.3389/fbioe.2022.890901.

[17] U. Alam, O. Asghar, S. Azmi, and R. A. Malik, "General aspects of diabetes mellitus," 2014, pp. 211–222. doi: 10.1016/B978-0-444-53480-4.00015-1.

[18] K. Kaul, J. M. Tarr, S. I. Ahmad, E. M. Kohner, and R. Chibber, "Introduction to Diabetes Mellitus," 2013, pp. 1–11. doi: 10.1007/978-1-4614-5441-0_1.

[19] H. Feng, P. Jin, and H. Wu, "Disease prediction by cell-free DNA methylation," *Brief Bioinform*, vol. 20, no. 2, pp. 585–597, Mar. 2019, doi: 10.1093/bib/bby029.

[20] D. Lee *et al.*, “A method to predict the impact of regulatory variants from DNA sequence,” *Nat Genet*, vol. 47, no. 8, pp. 955–961, Aug. 2015, doi: 10.1038/ng.3331.

[21] C. Ao, L. Yu, and Q. Zou, “Prediction of bio-sequence modifications and the associations with diseases,” *Brief Funct Genomics*, vol. 20, no. 1, pp. 1–18, Mar. 2021, doi: 10.1093/bfgp/elaa023.

[22] R. J. A. Little and D. B. Rubin, “Introduction,” in *Statistical Analysis with Missing Data*, 2014, pp. 1–23. doi: 10.1002/9781119013563.ch1.

[23] S. C. Manekar and S. R. Sathe, “A benchmark study of k-mer counting methods for high-throughput sequencing,” *Gigascience*, Oct. 2018, doi: 10.1093/gigascience/giy125.

[24] H. Gunasekaran, K. Ramalakshmi, A. Rex Macedo Arokiaraj, S. Deepa Kanmani, C. Venkatesan, and C. Suresh Gnana Dhas, “Analysis of DNA Sequence Classification Using CNN and Hybrid Models,” *Comput Math Methods Med*, vol. 2021, pp. 1–12, Jul. 2021, doi: 10.1155/2021/1835056.

[25] S. C. Manekar and S. R. Sathe, “A benchmark study of k-mer counting methods for high-throughput sequencing,” *Gigascience*, Oct. 2018, doi: 10.1093/gigascience/giy125.

[26] N. Pérez, M. Gutierrez, and N. Vera, “Computational Performance Assessment of k-mer Counting Algorithms,” *Journal of Computational Biology*, vol. 23, no. 4, pp. 248–255, Apr. 2016, doi: 10.1089/cmb.2015.0199.

[27] P. Kumar, R. Bhatnagar, K. Gaur, and A. Bhatnagar, “Classification of Imbalanced Data:Review of Methods and Applications,” *IOP Conf Ser Mater Sci Eng*, vol. 1099, no. 1, p. 012077, Mar. 2021, doi: 10.1088/1757-899X/1099/1/012077.

[28] R. Mohammed, J. Rawashdeh, and M. Abdullah, “Machine Learning with Oversampling and Undersampling Techniques: Overview Study and Experimental Results,” in *2020 11th International Conference on Information and Communication Systems (ICICS)*, IEEE, Apr. 2020, pp. 243–248. doi: 10.1109/ICICS49469.2020.239556.

[29] W. Xia *et al.*, “High-Resolution Remote Sensing Imagery Classification of Imbalanced Data Using Multistage Sampling Method and Deep Neural Networks,” *Remote Sens (Basel)*, vol. 11, no. 21, p. 2523, Oct. 2019, doi: 10.3390/rs11212523.

[30] A. C. H. Choong and N. K. Lee, “Evaluation of convolutionary neural networks modeling of DNA sequences using ordinal versus one-hot encoding method,” in *2017 International Conference on Computer and Drone Applications (IConDA)*, IEEE, Nov. 2017, pp. 60–65. doi: 10.1109/ICONDA.2017.8270400.

[31] K. Kunanbayev, I. Temirbek, and A. Zollanvari, “Complex Encoding,” in *2021 International Joint Conference on Neural Networks (IJCNN)*, IEEE, Jul. 2021, pp. 1–6. doi: 10.1109/IJCNN52387.2021.9534094.

[32] K. Kunanbayev, I. Temirbek, and A. Zollanvari, “Complex Encoding,” in *2021 International Joint Conference on Neural Networks (IJCNN)*, IEEE, Jul. 2021, pp. 1–6. doi: 10.1109/IJCNN52387.2021.9534094.

[33] V. Gajera, Shubham, R. Gupta, and P. K. Jana, “An effective Multi-Objective task scheduling algorithm using Min-Max normalization in cloud computing,” in *2016 2nd International Conference on Applied and Theoretical Computing and Communication Technology (iCATccT)*, IEEE, 2016, pp. 812–816. doi: 10.1109/ICATCCT.2016.7912111.

[34] S. Salcedo-Sanz, L. Cornejo-Bueno, L. Prieto, D. Paredes, and R. García-Herrera, “Feature selection in machine learning prediction systems for renewable energy applications,” *Renewable and Sustainable Energy Reviews*, vol. 90, pp. 728–741, Jul. 2018, doi: 10.1016/j.rser.2018.04.008.

[35] E. Hancer, B. Xue, and M. Zhang, “A survey on feature selection approaches for clustering,” *Artif Intell Rev*, vol. 53, no. 6, pp. 4519–4545, Aug. 2020, doi: 10.1007/s10462-019-09800-w.

[36] J. R. Vergara and P. A. Estévez, “A review of feature selection methods based on mutual information,” *Neural Comput Appl*, vol. 24, no. 1, pp. 175–186, Jan. 2014, doi: 10.1007/s00521-013-1368-0.

[37] M. A. Sulaiman and J. Labadin, “Feature selection based on mutual information,” in *2015 9th International Conference on IT in Asia (CITA)*, IEEE, Aug. 2015, pp. 1–6. doi: 10.1109/CITA.2015.7349827.

[38] A.-H. Jiang, X.-C. Huang, Z.-H. Zhang, J. Li, Z.-Y. Zhang, and H.-X. Hua, “Mutual information algorithms,” *Mech Syst Signal Process*, vol. 24, no. 8, pp. 2947–2960, Nov. 2010, doi: 10.1016/j.ymssp.2010.05.015.

[39] H. Ding, P.-M. Feng, W. Chen, and H. Lin, “Identification of bacteriophage virion proteins by the ANOVA feature selection and analysis,” *Mol. BioSyst.*, vol. 10, no. 8, pp. 2229–2235, 2014, doi: 10.1039/C4MB00316K.

[40] M. Bejani, D. Gharavian, and N. M. Charkari, “Audiovisual emotion recognition using ANOVA feature selection method and multi-classifier neural networks,” *Neural Comput Appl*, vol. 24, no. 2, pp. 399–412, Feb. 2014, doi: 10.1007/s00521-012-1228-3.

[41] A. Forte, G. Garcia-Donato, and M. Steel, “Methods and Tools for Bayesian Variable Selection and Model Averaging in Univariate Linear Regression,” Dec. 2016.

[42] J. C. F. de Winter, S. D. Gosling, and J. Potter, “Comparing the Pearson and Spearman correlation coefficients across distributions and sample sizes: A tutorial using simulations and empirical data.,” *Psychol Methods*, vol. 21, no. 3, pp. 273–290, Sep. 2016, doi: 10.1037/met0000079.

[43] Md. A. Hasan, Md. K. Hasan, and M. A. Mottalib, “Linear regression-based feature selection for microarray data classification,” *Int J Data Min Bioinform*, vol. 11, no. 2, p. 167, 2015, doi: 10.1504/IJDMB.2015.066776.

[44] E. Alpaydin, *Introduction to Machine Learning*, 3rd ed. MIT Press, 2014.

[45] T. Jiang, J. L. Gradus, and A. J. Rosellini, “Supervised Machine Learning: A Brief Primer,” *Behav Ther*, vol. 51, no. 5, pp. 675–687, Sep. 2020, doi: 10.1016/j.beth.2020.05.002.

[46] M. Alloghani, D. Al-Jumeily, J. Mustafina, A. Hussain, and A. J. Aljaaf, “A Systematic Review on Supervised and Unsupervised Machine Learning Algorithms for Data Science,” 2020, pp. 3–21. doi: 10.1007/978-3-030-22475-2_1.

[47] S. Dong, P. Wang, and K. Abbas, “A survey on deep learning and its applications,” *Comput Sci Rev*, vol. 40, p. 100379, May 2021, doi: 10.1016/j.cosrev.2021.100379.

[48] X. Dong, Z. Yu, W. Cao, Y. Shi, and Q. Ma, “A survey on ensemble learning,” *Front Comput Sci*, vol. 14, no. 2, pp. 241–258, Apr. 2020, doi: 10.1007/s11704-019-8208-z.

[49] S. Sun, Z. Cao, H. Zhu, and J. Zhao, “A Survey of Optimization Methods From a Machine Learning Perspective,” *IEEE Trans Cybern*, vol. 50, no. 8, pp. 3668–3681, Aug. 2020, doi: 10.1109/TCYB.2019.2950779.

[50] R.-Y. Sun, “Optimization for Deep Learning: An Overview,” *Journal of the Operations Research Society of China*, vol. 8, no. 2, pp. 249–294, Jun. 2020, doi: 10.1007/s40305-020-00309-6.

[51] L. Bottou, F. E. Curtis, and J. Nocedal, “Optimization Methods for Large-Scale Machine Learning,” *SIAM Review*, vol. 60, no. 2, pp. 223–311, Jan. 2018, doi: 10.1137/16M1080173.

[52] François Fleuret, *The little book of Deep Learning*. Alanna Maldonado, 2023.

[53] Y. You, Z. Zhang, C.-J. Hsieh, J. Demmel, and K. Keutzer, “ImageNet Training in Minutes,” Sep. 2017.

[54] Sudip Kundu, “Confusion killer : Epoch Vs Batch size Vs Iteration,” LinkedIn. Accessed: Apr. 21, 2023. [Online]. Available: https://www.linkedin.com/pulse/confusion-killer-epoch-vs-batch-size-iteration-sudip-kundu/

[55] S. KILIÇARSLAN, K. ADEM, and M. ÇELİK, “An overview of the activation functions used in deep learning algorithms,” *Journal of New Results in Science*, vol. 10, no. 3, pp. 75–88, Dec. 2021, doi: 10.54187/jnrs.1011739.

[56] M. Gustineli, “A survey on recently proposed activation functions for Deep Learning,” Apr. 2022.

[57] S. R. Dubey, S. K. Singh, and B. B. Chaudhuri, “Activation functions in deep learning: A comprehensive survey and benchmark,” *Neurocomputing*, vol. 503, pp. 92–108, Sep. 2022, doi: 10.1016/j.neucom.2022.06.111.

[58] J. Feng, X. He, Q. Teng, C. Ren, H. Chen, and Y. Li, “Reconstruction of porous media from extremely limited information using conditional generative adversarial networks,” *Phys Rev E*, vol. 100, no. 3, p. 033308, Sep. 2019, doi: 10.1103/PhysRevE.100.033308.

[59] F. Nie, Z. Hu, and X. Li, “An investigation for loss functions widely used in machine learning,” *Commun Inf Syst*, vol. 18, no. 1, pp. 37–52, 2018, doi: 10.4310/CIS.2018.v18.n1.a2.

[60] Q. Wang, Y. Ma, K. Zhao, and Y. Tian, “A Comprehensive Survey of Loss Functions in Machine Learning,” *Annals of Data Science*, vol. 9, no. 2, pp. 187–212, Apr. 2022, doi: 10.1007/s40745-020-00253-5.

[61] W. Wang and Y. Lu, “Analysis of the Mean Absolute Error (MAE) and the Root Mean Square Error (RMSE) in Assessing Rounding Model,” *IOP Conf Ser Mater Sci Eng*, vol. 324, p. 012049, Mar. 2018, doi: 10.1088/1757-899X/324/1/012049.

[62] D. Ramos, J. Franco-Pedroso, A. Lozano-Diez, and J. Gonzalez-Rodriguez, “Deconstructing Cross-Entropy for Probabilistic Binary Classifiers,” *Entropy*, vol. 20, no. 3, p. 208, Mar. 2018, doi: 10.3390/e20030208.

[63] D. A. Omondiagbe, S. Veeramani, and A. S. Sidhu, “Machine Learning Classification Techniques for Breast Cancer Diagnosis,” *IOP Conf Ser Mater Sci Eng*, vol. 495, p. 012033, Jun. 2019, doi: 10.1088/1757-899X/495/1/012033.

[64] S. Suthaharan, “Support Vector Machine,” 2016, pp. 207–235. doi: 10.1007/978-1-4899-7641-3_9.

[65] A. Tharwat, “Parameter investigation of support vector machine classifier with kernel functions,” *Knowl Inf Syst*, vol. 61, no. 3, pp. 1269–1302, Dec. 2019, doi: 10.1007/s10115-019-01335-4.

[66] B. Ghaddar and J. Naoum-Sawaya, “High dimensional data classification and feature selection using support vector machines,” *Eur J Oper Res*, vol. 265, no. 3, pp. 993–1004, Mar. 2018, doi: 10.1016/j.ejor.2017.08.040.

[67] S. Sampath, “Support Vector Machine (SVM) Algorithm - Javatpoint.”

[68] I. Syarif, A. Prugel-Bennett, and G. Wills, “SVM Parameter Optimization using Grid Search and Genetic Algorithm to Improve Classification Performance,” *TELKOMNIKA (Telecommunication Computing Electronics and Control)*, vol. 14, no. 4, p. 1502, Dec. 2016, doi: 10.12928/telkomnika.v14i4.3956.

[69] C. C. Aggarwal, “An Introduction to Neural Networks,” in *Neural Networks and Deep Learning*, Cham: Springer International Publishing, 2018, pp. 1–52. doi: 10.1007/978-3-319-94463-0_1.

[70] J. B. Ahire, “The Artificial Neural Networks Handbook: Part 4.” Accessed: Jul. 27, 2020. [Online]. Available: https://dzone.com/articles/the-artificial-neural-networks-handbook-part-4

[71] S. Suthaharan, “Decision Tree Learning,” 2016, pp. 237–269. doi: 10.1007/978-1-4899-7641-3_10.

[72] B. Charbuty and A. Abdulazeez, “Classification Based on Decision Tree Algorithm for Machine Learning,” *Journal of Applied Science and Technology Trends*, vol. 2, no. 01, pp. 20–28, Mar. 2021, doi: 10.38094/jastt20165.

[73] B. Charbuty and A. Abdulazeez, “Classification Based on Decision Tree Algorithm for Machine Learning,” *Journal of Applied Science and Technology Trends*, vol. 2, no. 01, pp. 20–28, Mar. 2021, doi: 10.38094/jastt20165.

[74] Y. Qi, “Random Forest for Bioinformatics,” in *Ensemble Machine Learning*, New York, NY: Springer New York, 2012, pp. 307–323. doi: 10.1007/978-1-4419-9326-7_11.

[75] G. Biau and E. Scornet, “A random forest guided tour,” *TEST*, vol. 25, no. 2, pp. 197–227, Jun. 2016, doi: 10.1007/s11749-016-0481-7.

[76] H. Sahour, V. Gholami, J. Torkaman, M. Vazifedan, and S. Saeedi, “Random forest and extreme gradient boosting algorithms for streamflow modeling using vessel features and tree-rings,” *Environ Earth Sci*, vol. 80, no. 22, p. 747, Nov. 2021, doi: 10.1007/s12665-021-10054-5.

[77] G. Biau and E. Scornet, “A random forest guided tour,” *TEST*, vol. 25, no. 2, pp. 197–227, Jun. 2016, doi: 10.1007/s11749-016-0481-7.

[78] A. Kelly and M. A. Johnson, “Investigating the Statistical Assumptions of Naïve Bayes Classifiers,” in *2021 55th Annual Conference on Information Sciences and Systems (CISS)*, IEEE, Mar. 2021, pp. 1–6. doi: 10.1109/CISS50987.2021.9400215.

[79] H. Kamel, D. Abdulah, and J. M. Al-Tuwaijari, “Cancer Classification Using Gaussian Naive Bayes Algorithm,” in *2019 International Engineering Conference (IEC)*, IEEE, Jun. 2019, pp. 165–170. doi: 10.1109/IEC47844.2019.8950650.

[80] M. Ontivero-Ortega, A. Lage-Castellanos, G. Valente, R. Goebel, and M. Valdes-Sosa, “Fast Gaussian Naïve Bayes for searchlight classification analysis,” *Neuroimage*, vol. 163, pp. 471–479, Dec. 2017, doi: 10.1016/j.neuroimage.2017.09.001.

[81] F. Shaheen, B. Verma, and Md. Asafuddoula, “Impact of Automatic Feature Extraction in Deep Learning Architecture,” in *2016 International Conference on Digital Image Computing: Techniques and Applications (DICTA)*, IEEE, Nov. 2016, pp. 1–8. doi: 10.1109/DICTA.2016.7797053.

[82] Aston Zhang, Zack C. Lipton, Mu Li, and Alex J. Smola, *Dive into Deep Learning*. Cambridge University Press, 2023.

[83] A. A. Abdul lateef, S. T. F. Al-Janabi, and B. Al-Khateeb, “Survey on intrusion detection systems based on deep learning,” *Periodicals of*

Engineering and Natural Sciences (PEN), vol. 7, no. 3, p. 1074, Aug. 2019, doi: 10.21533/pen.v7i3.635.

[84] U. Odi and T. Nguyen, "Geological Facies Prediction Using Computed Tomography in a Machine Learning and Deep Learning Environment," in *Proceedings of the 6th Unconventional Resources Technology Conference*, Tulsa, OK, USA: American Association of Petroleum Geologists, 2018. doi: 10.15530/urtec-2018-2901881.

[85] A. Zappone, M. Di Renzo, and M. Debbah, "Wireless Networks Design in the Era of Deep Learning: Model-Based, AI-Based, or Both?," *IEEE Transactions on Communications*, vol. 67, no. 10, pp. 7331–7376, Oct. 2019, doi: 10.1109/TCOMM.2019.2924010.

[86] R. Chauhan, K. K. Ghanshala, and R. C. Joshi, "Convolutional Neural Network (CNN) for Image Detection and Recognition," in *2018 First International Conference on Secure Cyber Computing and Communication (ICSCCC)*, IEEE, Dec. 2018, pp. 278–282. doi: 10.1109/ICSCCC.2018.8703316.

[87] I. Goodfellow, Y. Bengio, and A. Courville, "The Convolution Operation," in *Deep Learning*, MIT Press, 2015, pp. 322–330.

[88] Yi Zhou, "Sentiment classifcation with deep neural networks," Tampere University, 2019.

[89] I. Goodfellow, Y. Bengio, and A. Courville, "Pooling," in *Deep Learning*, MIT Press, 2016, pp. 330–334.

[90] S. Bhatnagar, L. Gill, and B. Ghosh, "Drone Image Segmentation Using Machine and Deep Learning for Mapping Raised Bog Vegetation Communities," *Remote Sens (Basel)*, vol. 12, no. 16, p. 2602, Aug. 2020, doi: 10.3390/rs12162602.

[91] C.-L. Zhang, J.-H. Luo, X.-S. Wei, and J. Wu, "In Defense of Fully Connected Layers in Visual Representation Transfer," in *Advances in Multimedia Information Processing – PCM 2017*, 2018, pp. 807–817. doi: 10.1007/978-3-319-77383-4_79.

[92] S. Rehman, S. Tu, O. Rehman, Y. Huang, C. Magurawalage, and C.-C. Chang, "Optimization of CNN through Novel Training Strategy for Visual Classification Problems," *Entropy*, vol. 20, no. 4, p. 290, Apr. 2018, doi: 10.3390/e20040290.

[93] P. Li, J. Li, and G. Wang, “Application of Convolutional Neural Network in Natural Language Processing,” in *2018 15th International Computer Conference on Wavelet Active Media Technology and Information Processing (ICCWAMTIP)*, IEEE, Dec. 2018, pp. 120–122. doi: 10.1109/ICCWAMTIP.2018.8632576.

[94] W. Wang and J. Gang, “Application of Convolutional Neural Network in Natural Language Processing,” in *2018 International Conference on Information Systems and Computer Aided Education (ICISCAE)*, IEEE, Jul. 2018, pp. 64–70. doi: 10.1109/ICISCAE.2018.8666928.

[95] B. Zhao, H. Lu, S. Chen, J. Liu, and D. Wu, “Convolutional neural networks for time series classification,” *Journal of Systems Engineering and Electronics*, vol. 28, no. 1, pp. 162–169, Feb. 2017, doi: 10.21629/JSEE.2017.01.18.

[96] G. Van Houdt, C. Mosquera, and G. Nápoles, “A review on the long short-term memory model,” *Artif Intell Rev*, vol. 53, no. 8, pp. 5929–5955, Dec. 2020, doi: 10.1007/s10462-020-09838-1.

[97] K. Zarzycki and M. Ławryńczuk, “LSTM and GRU Neural Networks as Models of Dynamical Processes Used in Predictive Control: A Comparison of Models Developed for Two Chemical Reactors,” *Sensors*, vol. 21, no. 16, p. 5625, Aug. 2021, doi: 10.3390/s21165625.

[98] B. Ghojogh and M. Crowley, “The Theory Behind Overfitting, Cross Validation, Regularization, Bagging, and Boosting: Tutorial,” May 2019.

[99] J. Balayla, “Prevalence threshold (ϕe) and the geometry of screening curves,” *PLoS One*, vol. 15, no. 10, p. e0240215, Oct. 2020, doi: 10.1371/journal.pone.0240215.

[100] Baishalini Sahu, “What evaluation approaches would you work to deal with the effectiveness of a machine learning model,” *LinkedIn*. Jul. 22, 2021.

[101] “National Library of Medicine.” Accessed: May 14, 2023. [Online]. Available: https://www.ncbi.nlm.nih.gov/

[102] NCBI, “INS insulin [Homo sapiens (human)].” Accessed: Jan. 16, 2023. [Online]. Available: https://www.ncbi.nlm.nih.gov/gene?Db=gene&Cmd=DetailsSearch&Term=3630

المستخلص

داء السكري (DM) هو اضطراب أيضي مزمن يتميز بارتفاع مستويات السكر في الدم. ويحدث ذلك عندما لا ينتج الجسم ما يكفي من الأنسولين أو يفشل في استخدام الأنسولين الذي ينتجه بشكل فعال. يمكن تصنيف مرض السكري إلى ثلاثة أنواع رئيسية: النوع الأول، والنوع الثاني، وسكري الحمل. إذا ترك مرض السكري دون السيطرة عليه، فإنه يمكن أن يؤدي إلى مضاعفات صحية خطيرة ويصبح خطيرا. يمكن أن يؤدي ارتفاع مستويات السكر في الدم لفترة طويلة إلى إتلاف أعضاء وأنظمة مختلفة، بما في ذلك القلب والأوعية الدموية والكلى والعينين والأعصاب. يزيد مرض السكري من خطر الإصابة بأمراض القلب والأوعية الدموية والسكتة الدماغية وأمراض الكلى وفقدان البصر وبتر الأطراف السفلية.

تحليل الحمض النووي، المعروف أيضًا باسم اختبار الحمض النووي أو الاختبار الجيني، هو أسلوب علمي يستخدم لفحص وتحليل الحمض النووي للفرد (الحمض النووي الريبي منقوص الأكسجين). وهو يتضمن دراسة تسلسلات معينة من الحمض النووي أو الجينات أو الكروموسومات للحصول على نظرة ثاقبة للجوانب المختلفة لعلم الوراثة البشرية وتحديد الاختلافات أو الطفرات الجينية المحتملة. من خلال فتح المعلومات الجينية المشفرة في حمضنا النووي، قدم تحليل الحمض النووي فهمًا أعمق لتركيبتنا الجينية وساهم بشكل كبير في التقدم في الطب الشخصي والبحث الجيني.

في هذه الأطروحة، يتم استخدام تحليل الحمض النووي لتصنيف مرض السكري والتنبؤ به لدى المرضى. مما يؤدي إلى الكشف المبكر عن مرض السكري والوقاية من مضاعفاته.

تحتوي مجموعة البيانات على الحمض النووي لـ ١٤٥٧١ شخصًا. تتم معالجة البيانات مسبقًا للتخلص من البيانات المفقودة. وتطبيق تقنية k-mer بأحجام مختلفة. بعد ذلك الإفراط في أخذ العينات العشوائية لعدم التوازن في مجموعة البيانات. تطبيق الترميز الترتيبي، وأخيراً تحويل.Min-Max

بعد ذلك يتم تطبيق ثلاث تقنيات لاختيار الميزات بشكل منفصل. وهي المعلومات المتبادلة، ANOVA، والانحدار الخطي أحادي المتغير.

وأخيرًا، تم تطوير نماذج التعلم الآلي والتعلم العميق لتصنيف حالة مرض السكري بناءً على تسلسل الحمض النووي. نماذج التعلم الآلي هيRandom Forest ، وSupport Vector Machine، وDecision Tree، و .Gaussian Naïve Bayesنموذج التعلم العميق هو مزيج من-CNN LSTM.

تم بناء النظام المطور واختباره على جهاز الكمبيوتر. المعالج هو Intel® Core™ i7-6600U CPU، بسرعة ٢.٦٠ جيجاهرتز. ذاكرة الوصول العشوائي المثبتة هي ٨ جيجابايت. أفضل النتائج التي تم الحصول عليها باستخدام أسلوب التعلم الآلي هي باستخدامRandom Forest ، بدقة تصل إلى ٩٠%. وأفضل نتيجة تم الحصول عليها باستخدام منهج التعلم العميق هي باستخدام نموذج-CNN LSTM، بدقة ودقة واستدعاء ودرجة F1 بنسبة ١٠٠.%

جمهورية العراق

وزارة التعليم العالي و البحث العلمي

جامعة بابل – كلية العلوم للبنات

قسم علوم الحاسوب

تحديد مرض السكر بناء على بيانات DNA وتقنيات تعلم الآلة

رسالة مقدمة الى

مجلس كلية العلوم للبنات-جامعة بابل

وهي جزء من متطلبات نيل درجة الماجستير في علوم الحاسبات

من قبل

لينا عبدالرحيم حمزه

بإشراف

أ.د. حسين عطية الخالدي

أ.م.د. سرى زكي ناجي الراشد

١٤٤٥ ٢٠٢٣