UTM UNIVERSITI TEKNOLOGI MALAYSIA
INTERNATIONAL JOURNAL OF INNOVATIVE COMPUTING
ISSN 2180-4370
Journal Homepage : https://ijic.utm.my/

Text Classification on Diabetes Mellitus Symptom and Treatment Documents Using Machine Learning Approaches

Phang Cheng Yi
Faculty of Computing
Universiti Teknologi Malaysia
Skudai, Johor, Malaysia
phangyi@graduate.utm.my

***Abstract*—This study aims to develop a classification model for the symptoms and treatments of Diabetes Mellitus (DM) using machine learning approaches based on the abstracts of DM-related journal articles. A systematic research methodology is designed to achieve the aim of research. The dataset is collected from PubMed website using Entrez in Biopython and pre-processed by splitting abstracts into sentences and performing text-cleaning techniques, labelling and imbalance data handling. TF-IDF method is used for feature extraction and vectorizes the pre-processed data into numerical data for fitting in the machine learning models such as SVM, Logistic Regression, Decision Tree, KNN, and Random Forest. The performance of these models is evaluated with several performance metrics based on confusion matrix, such as accuracy, precision, and recall. The performance of applying different hyperparameters and different sets of training and testing data sets will be compared. Most of the classifiers obtained the best performance when using 80:20 splits. The SVM model outperforms the other classifiers in terms of performance metrics and confusion matrix. The research findings can provide a solution for identifying and classifying the symptoms and treatments of DM from multiple biomedical journal articles more efficiently and facilitate the development of more effective interventions for DM management.**

Keywords — Text Classification, Diabetes Mellitus, Symptom and Treatment, Machine Learning.

I. INTRODUCTION

Diabetes mellitus (DM) is a metabolic disorder characterised by excessively increased blood glucose levels with numerous subtypes including type 1 diabetes, type 2 diabetes, maturity-onset diabetes of the young (MODY), gestational diabetes, neonatal diabetes, and steroid-induced diabetes [1]. Hyperglycemia is caused by abnormalities in either insulin secretion or insulin action, or both. The severe and varied symptoms of hyperglycemia include abnormalities in the metabolism of carbohydrates, fats, and proteins [2]. These issues impact body organs, including the eyes, kidneys, heart, and nerves, causing organ damage, dysfunction, and eventually organ failure. Therefore, early detection and efficient management is required but it's also challenging.

Due to the increasing number of people with diabetes, there are a lot of publishments regarding to diabetes for early diagnosis, treatment, management, and prevention. Inevitably, biomedical fields experts continuously publish their research outcome in several different sources of biomedical text database such as PubMed, MEDLINE, EMBASE and Google Scholar have been generating an enormous number of relevant documents. This phenomenon has increased the challenge for information seekers to discover the main points from a vast number of documents that stored in the database and may face difficulty classifying the important information from these documents as well as identify their relationships for better understanding.

A notable example of a disease that could profit from the creation of actual evidence using Natural Language Processing (NLP) is DM. Text classifications is one of the NLP unstructured text analysis techniques used in extracting information from textual resources like journal articles, newspaper, email, and image. A predetermined label or tag will be given to each document in the dataset by the classifier [3]. There are several types of commonly used machine learning text classifiers such as Support Vector Machine (SVM), Decision Tree, Random Forest, Logistic Regression, and K-Nearest-Neighbors (KNN).

This research study is aimed to develop and evaluate a machine learning-based text classification model for DM symptom and treatment documents. The objectives of the research include identifying the significant features that are relevant to Diabetes Mellitus (DM) symptoms and treatment in multiple documents, performing text classification for a collection of DM documents dataset using machine learning methods and evaluating the performance of machine learning models.

II. Literature Review

A. Text Classification

Text classification is the automated assignment of text documents into predefined groups according to the content of the text itself by using various technologies and algorithms [4]. Due to the vast amount of information available from numerous sources, such as Internet pages, news pages, e-mail messages, texts shared through social media platforms, reports, and journal articles, dedicated research on the classification of texts has increased significantly. In recent years, text classification issues have received extensive study and have been tackled through numerous real-world applications [5].

Text classification systems can be break down into four stages which includes feature extraction, dimension reductions, classifier selection, and evaluations. Initially, the raw text data must be cleaned and pre-processed. Furthermore, text enrichment method is needed for certain text data to expand user's query to improve keywords matching. When a classifier is built using mathematical modelling, unstructured text sequences must be transformed into a structured feature space to make sure feature extraction can work smoothly. The utilization of dimensionality reduction can efficiently resolve the problems of hefty memory requirements and time complexity in text pre-processing to scale down the feature.

B. Feature Extraction

A text can be converted into a keyword schedule using feature extraction, which makes it feasible for supervised learning to parse the text with ease [6]. Vectorizing text data is necessary before utilizing it as input for a machine learning system since the input must be numerical [7]. Term Frequency-Inverse Document Frequency (TF-IDF) is a simple and effective term weighting vectorizer. The TF-IDF emphasizes a word's importance in a text in relation to the entire corpus [7]. The basic concept is that a term that frequently appears in a document but infrequently in the whole corpus is more informative than a word that regularly appears in both corpus and document. Only words with higher TF-IDF values will be considered as input. There are two steps involved in TF-IDF, calculating Term Frequency (TF) and Inverse Document Frequency (IDF).

C. Multilabel Classification

Multi-label Classification (MLC) is one of the primary categories of classification algorithms in classification. MLC can solve the problem that involves several fields such as text classification, tag recommendation, music categorization and image labelling [8]. The concept of MLC contrasts with Multiclass Classification where each instance in a dataset can belong to multiple labels in multi-label cases [9]. It is the extension from single-label classification. For example, in text classification, a news article that discusses politics and sports. This article could be concurrently assigned to the "politics" and "sports" labels in a multi-label classification scenario, implying that it covers issues relevant to both categories.

D. Machine Learning Approaches

Machine learning (ML) is frequently referred to as the newest technologies that are most in demand during the fourth industrial revolution (4IR or Industry 4.0). ML is a discipline of artificial intelligence (AI) that enables computers to perform tasks and learn from experience regardless of whether they are being specifically programmed [11]. In Industry 4.0, the employment of smart technologies like machine learning become trend in the automation of conventional industrial practices and manufacturing for exploratory data analysis and real-world applications development in their corresponding fields [12]. According to [13], there are four main types of machine learning algorithms: supervised, unsupervised, semi-supervised, and reinforcement learning. Supervised learning and unsupervised learning are among the most popularly used methods in text classification. Support Vector Machine (SVM), Naïve Bayes Classifier, K-Nearest Neighbor (KNN), Decision Tree and Random Forest are some of the examples of supervised machine learning algorithms.

E. Related Work

In the research conducted by [14]., they performed text classification of Urdu language using three well known classifiers which are Decision Tree, SVM and KNN. The evaluation was done on a collection of Urdu texts that included approximately 16,768 documents. They applied WEKA filter for features selection and feature extraction followed by employing the weighting scheme TF-IDF for scoring the features. The effectiveness of the classification models was tested by varying the size of top-ranked features. According to their result by comparing the three classifiers, SVM classifier shown the highest accuracy, 68.73% as compared to Decision Tree and KNN with accuracy of 62.37% and 55.41% respectively when selecting top 300 features with 80% training data.

Moreover, the recent study conducted by [7], is regarding the classification of textual data obtained from web resources like Google Scholar and Research Gate. Based on their results, they discovered that the SVM model with the utilization of TF-IDF vectorization outshines the other alternatives, especially the Decision Tree method by obtaining accuracy of 88%, which are much higher than the accuracy of Decision Tree model with TF-IDF. Meanwhile, the Naive Bayes and KNN methods do equivalently well with the accuracy of 86% and 83% respectively.

Singh and Chand [15] had conducted a research study regarding the detection of toxicity in social media comments using machine learning models such as MNB, Logistic Regression and SVM with TF-IDF. The dataset they utilized for multilabel classification is the Jigsaw dataset obtained from Google with six classes which are toxic, severe_toxic, obscene, threat, insult, and identity_hate. The results show that Logistic Regression outperformed other models with the highest F1-score of 0.9763 and the lowest hamming loss of 0.0257.

There is also another research conducted by [16] using the same dataset on classifying toxic comments for online safety. They proposed six machine learning algorithms, including Random Forest, Decision Tree, Logistic Regression, Naïve Bayes, and KNN. According to their results, the Logistic Regression model achieved highest accuracy of 89.46% with lowest hamming loss of 2.43 followed by SVM classifier with 88.69% accuracy and 2.76 hamming loss. Random Forest model has the lowest accuracy and highest hamming loss but with lowest log loss among the six machine learning models.

III. Methodology

The research methodology of this project is separated into four phases. The first phase is literature review, second phase is data collection and data preparation, third phase is design and implementation, and fourth phase is results and evaluation. The flowchart of research methodology is shown in Figure 3.1.

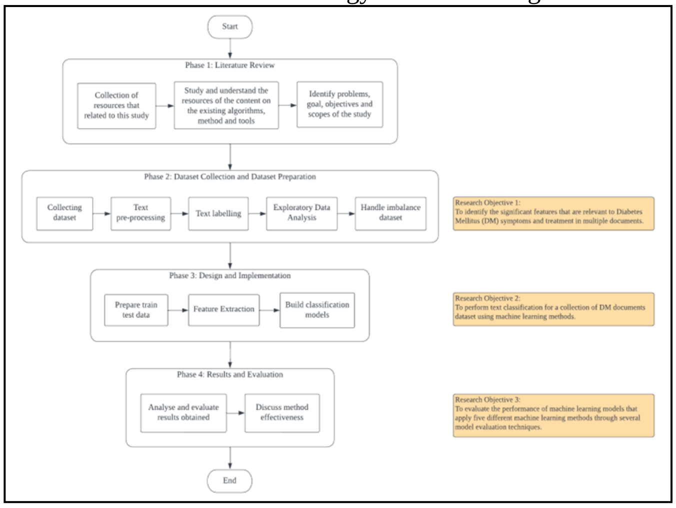

Start
Phase 1: Literature Review
Collection of resources that related to this study
Study and understand the resources of the content on the existing algorithms, method and tools
Identify problems, goal, objectives and scopes of the study
Phase 2: Dataset Collection and Dataset Preparation
Collecting dataset
Text pre-processing
Text labelling
Exploratory Data Analysis
Handle imbalance dataset
Research Objective 1: To identify the significant features that are relevant to Diabetes Mellitus (DM) symptoms and treatment in multiple documents.
Phase 3: Design and Implementation
Prepare train test data
Feature Extraction
Build classification models
Research Objective 2: To perform text classification for a collection of DM documents dataset using machine learning methods.
Phase 4: Results and Evaluation
Analyse and evaluate results obtained
Discuss method effectiveness
Research Objective 3: To evaluate the performance of machine learning models that apply five different machine learning methods through several model evaluation techniques.
End

Fig. 1. Workflow of Research Methodology

A. Data Preparation

The dataset used in this study is the textual data for DM that was scrapped from PubMed website. There are around 1020 articles PubMed articles are selected using the scrapper, Entrez, a global query cross-database search engine that enables users to search for biomedical publications through Biopython. After collecting the dataset, pre-processing is carried out to clean the text for classification. Before text cleaning, the abstract from raw dataset will be tokenized into sentences and saved in a new csv file. Then, this sentences dataset will be used to perform text pre-processing. The cleaned dataset will be used to perform text labelling using self-defined symptom and treatment keywords that are obtained from authorized websites such as Mayo Clinic, Diabetes UK, WebMD and World Health Organization (WHO) and research papers. Furthermore, the labelled dataset will be handled to solve imbalance problems that might cause bias during model training. Finally, the well-prepared dataset will be used to develop the text classification algorithms that are proposed.

B. Model Building

Firstly, the balanced dataset that is saved as a csv file is loaded. To prepare the training and testing data, the dataset is divided into 'X' and 'y' variables, where 'X' represents the sentences column while 'y' represents the labels column. Then, the dataset will be split into three sets of ratios: 70% for training data and 30% for testing data, 80% for training data and 20% for testing data, and 90% for training data and 10% for testing data.

Before fitting the text data into the model, it needs to be transformed into numerical values that can be understood by the classifiers. The 'TfidfVectorizer' class is imported to convert the text data to a matrix of TF-IDF features. The 'ngram_range' parameter is used to specify a range of n-values for different n-grams to be extracted. In this research, 'ngram_range' of (1, 5) means the creation of TF-IDF matrix will include unigrams which are single words, bigrams which are two-word combinations, and up to 5-grams which are five-word combinations. The training data will be fitted into the vectorizer to learn the vocabulary and the inverse document frequency (IDF) followed by the transformation of training and testing data into a sparse matrix of TF-IDF features.

There are five algorithms utilized for classification, including Support Vector Machine (SVM), Logistic Regression, Decision Tree, K-Nearest Neighbor (KNN) and Random Forest. The dataset will be trained by tuning the hyperparameters according to each classifier to identify the best parameter to be used for classification. All the results obtained, and its performance measurement will be further analyzed, compared and discussed.

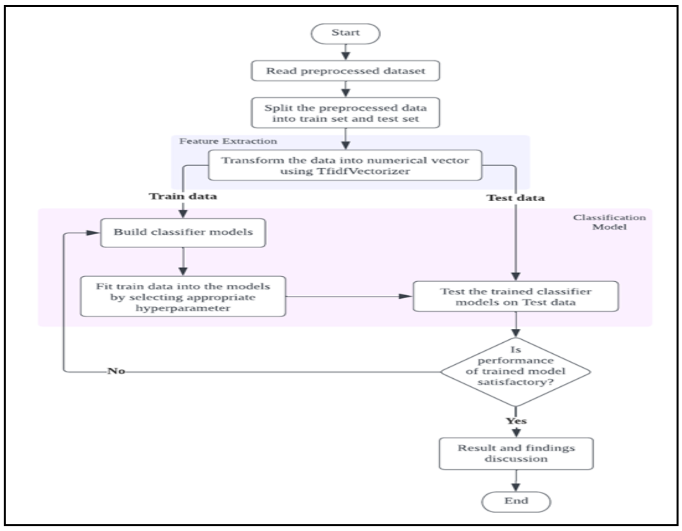

Start
Read preprocessed dataset
Split the preprocessed data into train set and test set
Feature Extraction
Transform the data into numerical vector using TfidfVectorizer
Train data
Test data
Classification Model
Build classifier models
Fit train data into the models by selecting appropriate hyperparameter
Test the trained classifier models on Test data
Is performance of trained model satisfactory?
No
Yes
Result and findings discussion
End

Fig. 2. Experimental Design of Classification Model

C. Performance Evaluation

The measurements such as accuracy, precision, recall and F1-score are based on the confusion matrix. The confusion matrix comprises of four evaluation outputs which are true positives (TP), false positives (FP), true negatives (TN) and false negatives (FN). TP denotes a positive class that is correctly predicted by the model, TN denotes a negative class that is correctly predicted by the model, FP denotes a positive class that is incorrectly predicted by the model and FN denotes a negative class that is incorrectly predicted by the model.

Accuracy is defined as the ratio of the correctly classified results to the total number of predictions. It can be calculated by using the following formula:

$$\text{Accuracy} = \frac{TP+TN}{TP+TN+FP+FN} \tag{1}$$

Precision is the proportion of correctly identified outcomes that were positively identified. It can be calculated by using the following formula:

$$\text{Precision} = \frac{TP}{TP+FP} \tag{2}$$

Recall indicates the percentage of data that was projected to be positive but turned out to be negative. It is the percentage of TP in the collection of all genuinely positive data.

$$\text{Recall} = \frac{TP}{TP+FN} \tag{3}$$

F1-score is calculated from the combination of Precision and Recall by getting their harmonic mean.

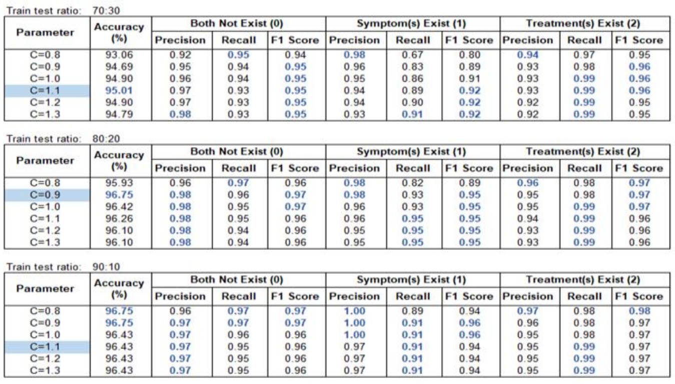
$$\text{F1-score} = 2 \times \frac{Precision \times Recall}{(Precision+Recall)} \tag{4}$$

IV. Results

In this section, the results of the five machine learning methods using performance metrics based on confusion matrix will be displayed and analyzed.

A. Performance Measure of Support Vector Machine

Fig. 3 shows the performance of SVM model trained with different regularization parameters (C) and evaluated on three different train-test splits (70:30, 80:20, 90:10). Based on Fig. 3, the parameter that gives the best result in train test split of 70:30 and 90:10 is C=1.1 while in train test split of 80:20 is C=0.9. For train test ratio of 70:30, the accuracy increases as C increases up to 1.1, peaking at 95.01%. Generally, there is no large difference in precision, recall and F1-score over all classes for C from 0.9 to 1.1 in average, and hence C=1.1 is selected to be the best results in train test ratio of 70:30 since it has the highest accuracy. For the train test ratio of 80:20, the accuracy peaks at C=0.9 with 96.75% and remains stable around 96.10% for higher values of C. In average of precision, recall and F1-score for all classes, C=0.9 give the best performance with highest accuracy in train test ratio of 80:20. For the train test ratio of 90:10, the accuracy peaks at C=0.9 with 96.75% and remains constant around 96.43% starting from C=1.0. Although the overall results for parameter C=0.9 is the highest, its precision value indicates that it might happen overfitting for class 1. Therefore, C=1.1 is considered as the best parameter with accuracy of 96.43% in train test ratio of 90:10 due to its precision, recall and F1-score are averagely high as others without overfitting in class 1. For optimal performance, a C value around 0.9 to 1.1 seems to provide the best balance between precision, recall and F1-score across all classes and train test splits. In SVM model, the best performance is given by C value of 0.9 with train test split ratio of 80:20.


Train test ratio: 70:30

Parameter	Accuracy (%)	Both Not Exist (0)			Symptom(s) Exist (1)			Treatment(s) Exist (2)		
		Precision	Recall	F1 Score	Precision	Recall	F1 Score	Precision	Recall	F1 Score
C=0.8	93.06	0.92	0.95	0.94	0.98	0.67	0.80	0.94	0.97	0.95
C=0.9	94.69	0.95	0.94	0.95	0.96	0.83	0.89	0.93	0.98	0.96
C=1.0	94.90	0.96	0.94	0.95	0.95	0.86	0.91	0.93	0.99	0.96
C=1.1	95.01	0.97	0.93	0.95	0.94	0.89	0.92	0.93	0.99	0.96
C=1.2	94.90	0.97	0.93	0.95	0.94	0.90	0.92	0.92	0.99	0.95
C=1.3	94.79	0.98	0.93	0.95	0.93	0.91	0.92	0.92	0.99	0.95

Train test ratio: 80:20

Parameter	Accuracy (%)	Both Not Exist (0)			Symptom(s) Exist (1)			Treatment(s) Exist (2)		
		Precision	Recall	F1 Score	Precision	Recall	F1 Score	Precision	Recall	F1 Score
C=0.8	95.93	0.96	0.97	0.96	0.98	0.82	0.89	0.96	0.98	0.97
C=0.9	96.75	0.98	0.96	0.97	0.98	0.93	0.95	0.95	0.98	0.97
C=1.0	96.42	0.98	0.95	0.97	0.96	0.93	0.95	0.95	0.99	0.97
C=1.1	96.26	0.98	0.95	0.96	0.96	0.95	0.95	0.94	0.99	0.96
C=1.2	96.10	0.98	0.94	0.96	0.95	0.95	0.95	0.93	0.99	0.96
C=1.3	96.10	0.98	0.94	0.96	0.95	0.95	0.95	0.93	0.99	0.96

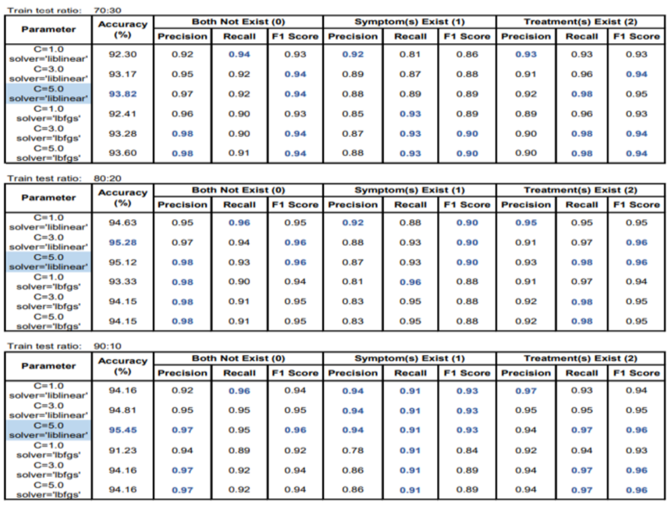
Train test ratio: 90:10

Parameter	Accuracy (%)	Both Not Exist (0)			Symptom(s) Exist (1)			Treatment(s) Exist (2)		
		Precision	Recall	F1 Score	Precision	Recall	F1 Score	Precision	Recall	F1 Score
C=0.8	96.75	0.96	0.97	0.97	1.00	0.89	0.94	0.97	0.98	0.98
C=0.9	96.75	0.97	0.97	0.97	1.00	0.91	0.96	0.96	0.98	0.97
C=1.0	96.43	0.97	0.96	0.96	1.00	0.91	0.96	0.95	0.98	0.97
C=1.1	96.43	0.97	0.95	0.96	0.97	0.91	0.94	0.95	0.99	0.97
C=1.2	96.43	0.97	0.95	0.96	0.97	0.91	0.94	0.95	0.99	0.97
C=1.3	96.43	0.97	0.95	0.96	0.97	0.91	0.94	0.95	0.99	0.97

Fig. 3. Results of Support Vector Machine

B. Performance Measure of Logistic Regression

Fig. 4 shows the performance of LR model trained with balanced class weight, different regularization parameters (C) and solvers, evaluated on three different train-test splits (70:30, 80:20, 90:10). Based on Fig. 4, the parameters that give the best result in all train-test splits are C=5.0 and solver='liblinear'. For train test ratio of 70:30, the highest accuracy is 93.82% with precision, recall and F1-score of 0.97, 0.92 and 0.94 for class 0, 0.88, 0.89 and 0.89 for class 1 and 0.92, 0.98, 0.95 for class 2. For train test ratio of 80:20, the highest accuracy is 95.28% with C=3.0 and solver='liblinear'. Although the hyperparameters with C=5.0 and solver='liblinear' has a lower accuracy than with C=3.0, it is selected as the best parameter for LR model with 80:20 train test split since its precision, recall and F1-score are higher in average across all classes. For the train test ratio of 90:10, the highest accuracy is 95.45% with the value of precision, recall and F1-score that are averagely the highest across all classes. According to the analysis results, higher values of C generally improve performance, especially for recall in class 1. The 'liblinear' solver often performs slightly better than 'lbfgs', especially with higher C values. Furthermore, increasing the training size generally enhances the model's performance, indicating the model benefits from more training data. For overall best performance, the configuration with C=5.0 and solver='liblinear' is recommended.


Train test ratio: 70:30

Parameter	Accuracy (%)	Both Not Exist (0)			Symptom(s) Exist (1)			Treatment(s) Exist (2)		
		Precision	Recall	F1 Score	Precision	Recall	F1 Score	Precision	Recall	F1 Score
C=1.0 solver='liblinear'	92.30	0.92	0.94	0.93	0.92	0.81	0.86	0.93	0.93	0.93
C=3.0 solver='liblinear'	93.17	0.95	0.92	0.94	0.89	0.87	0.88	0.91	0.96	0.94
C=5.0 solver='liblinear'	93.82	0.97	0.92	0.94	0.88	0.89	0.89	0.92	0.98	0.95
C=1.0 solver='lbfgs'	92.41	0.96	0.90	0.93	0.85	0.93	0.89	0.89	0.96	0.93
C=3.0 solver='lbfgs'	93.28	0.98	0.90	0.94	0.87	0.93	0.90	0.90	0.98	0.94
C=5.0 solver='lbfgs'	93.60	0.98	0.91	0.94	0.88	0.93	0.90	0.90	0.98	0.94

Train test ratio: 80:20

Parameter	Accuracy (%)	Both Not Exist (0)			Symptom(s) Exist (1)			Treatment(s) Exist (2)		
		Precision	Recall	F1 Score	Precision	Recall	F1 Score	Precision	Recall	F1 Score
C=1.0 solver='liblinear'	94.63	0.95	0.96	0.95	0.92	0.88	0.90	0.95	0.95	0.95
C=3.0 solver='liblinear'	95.28	0.97	0.94	0.96	0.88	0.93	0.90	0.91	0.97	0.96
C=5.0 solver='liblinear'	95.12	0.98	0.93	0.96	0.87	0.93	0.90	0.93	0.98	0.96
C=1.0 solver='lbfgs'	93.33	0.98	0.90	0.94	0.81	0.96	0.88	0.91	0.97	0.94
C=3.0 solver='lbfgs'	94.15	0.98	0.91	0.95	0.83	0.95	0.88	0.92	0.98	0.95
C=5.0 solver='lbfgs'	94.15	0.98	0.91	0.95	0.83	0.95	0.88	0.92	0.98	0.95

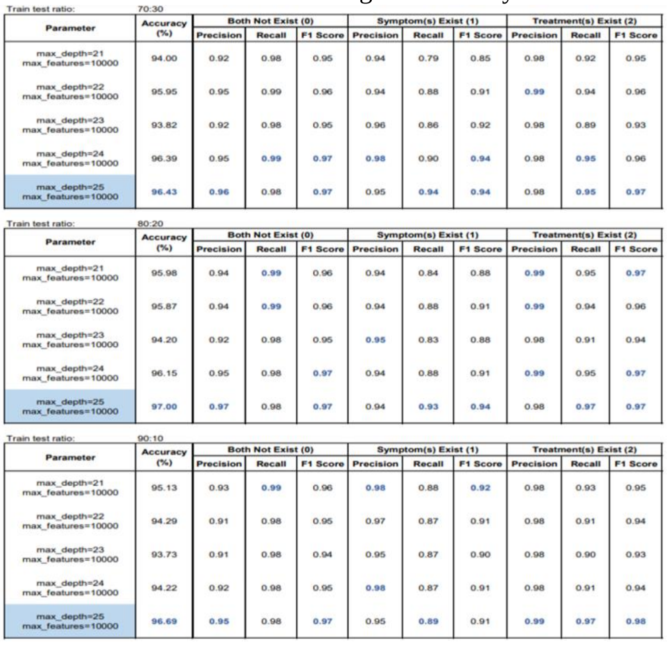
Train test ratio: 90:10

Parameter	Accuracy (%)	Both Not Exist (0)			Symptom(s) Exist (1)			Treatment(s) Exist (2)		
		Precision	Recall	F1 Score	Precision	Recall	F1 Score	Precision	Recall	F1 Score
C=1.0 solver='liblinear'	94.16	0.92	0.96	0.94	0.94	0.91	0.93	0.97	0.93	0.94
C=3.0 solver='liblinear'	94.81	0.95	0.95	0.95	0.94	0.91	0.93	0.95	0.95	0.95
C=5.0 solver='liblinear'	95.45	0.97	0.95	0.96	0.94	0.91	0.93	0.94	0.97	0.96
C=1.0 solver='lbfgs'	91.23	0.94	0.89	0.92	0.78	0.91	0.84	0.92	0.94	0.93
C=3.0 solver='lbfgs'	94.16	0.97	0.92	0.94	0.86	0.91	0.89	0.94	0.97	0.96
C=5.0 solver='lbfgs'	94.16	0.97	0.92	0.94	0.86	0.91	0.89	0.94	0.97	0.96

Fig. 4. Results of Logistic Regression

C. Performance Measure of Decision Tree

Fig. 5 shows the performance of Decision Tree model trained with 10,000 features and different maximum depth of the trees, evaluated on three different train-test splits (70:30, 80:20, 90:10). Based on Fig. 5, the parameters that give the best result in all train-test splits are max_depth=25 and max_features=10000. The accuracy for train test ratio of 70:30 peaks at 96.43% with the highest F1-score across all classes. For the train test ratio of 80:20, the highest accuracy is 97.00%. The F1-score in 80:20 is the same as in 70:30 but with slightly higher precision of 0.97 for class 0 and recall of 0.97 for class 2. For the train test ratio of 90:10, the accuracy peaks at 96.69%. The precision and F1-score of max_depth=25 is the highest for all classes in average. The recall for class 0 is same as the other two splits, stable around 0.98 and 0.99 over the five different maximum depths of trees. In conclusion, max_depth=25 shows the highest accuracy across all train-test splits. This maximum depth with 10,000 features provides a good balance between all metrics, especially for the "Symptom(s) Exist" and "Treatment(s) Exist" classes. In the Decision Tree model, the best performed results are given by train test ratio of 80:20 with the highest accuracy of 97.00%.



Train test ratio: 70:30

Parameter	Accuracy (%)	Both Not Exist (0)			Symptom(s) Exist (1)			Treatment(s) Exist (2)		
		Precision	Recall	F1 Score	Precision	Recall	F1 Score	Precision	Recall	F1 Score
max_depth=21 max_features=10000	94.00	0.92	0.98	0.95	0.94	0.79	0.85	0.98	0.92	0.95
max_depth=22 max_features=10000	95.95	0.95	0.99	0.96	0.94	0.88	0.91	0.99	0.94	0.96
max_depth=23 max_features=10000	93.82	0.92	0.98	0.95	0.96	0.86	0.92	0.98	0.89	0.93
max_depth=24 max_features=10000	96.39	0.95	0.99	0.97	0.98	0.90	0.94	0.98	0.95	0.96
max_depth=25 max_features=10000	96.43	0.96	0.98	0.97	0.95	0.94	0.94	0.98	0.95	0.97

Train test ratio: 80:20

Parameter	Accuracy (%)	Both Not Exist (0)			Symptom(s) Exist (1)			Treatment(s) Exist (2)		
		Precision	Recall	F1 Score	Precision	Recall	F1 Score	Precision	Recall	F1 Score
max_depth=21 max_features=10000	95.98	0.94	0.99	0.96	0.94	0.84	0.88	0.99	0.95	0.97
max_depth=22 max_features=10000	95.87	0.94	0.99	0.96	0.94	0.88	0.91	0.99	0.94	0.96
max_depth=23 max_features=10000	94.20	0.92	0.98	0.95	0.95	0.83	0.88	0.98	0.91	0.94
max_depth=24 max_features=10000	96.15	0.95	0.98	0.97	0.94	0.88	0.91	0.99	0.95	0.97
max_depth=25 max_features=10000	97.00	0.97	0.98	0.97	0.94	0.93	0.94	0.98	0.97	0.97

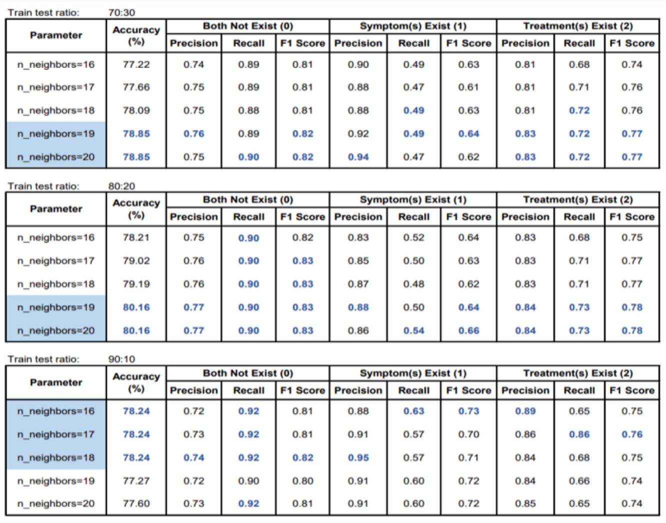
Train test ratio: 90:10

Parameter	Accuracy (%)	Both Not Exist (0)			Symptom(s) Exist (1)			Treatment(s) Exist (2)		
		Precision	Recall	F1 Score	Precision	Recall	F1 Score	Precision	Recall	F1 Score
max_depth=21 max_features=10000	95.13	0.93	0.99	0.96	0.98	0.88	0.92	0.98	0.93	0.95
max_depth=22 max_features=10000	94.29	0.91	0.98	0.95	0.97	0.87	0.91	0.98	0.91	0.94
max_depth=23 max_features=10000	93.73	0.91	0.98	0.94	0.95	0.87	0.90	0.98	0.90	0.93
max_depth=24 max_features=10000	94.22	0.92	0.98	0.95	0.98	0.87	0.91	0.98	0.91	0.94
max_depth=25 max_features=10000	96.69	0.95	0.98	0.97	0.95	0.89	0.91	0.99	0.97	0.98

Fig. 5. Results of Decision Tree

D. Performance Measure of K-Nearest Neighbor

Fig. 6 shows the performance of KNN model trained with different number of neighbors, evaluated on three different train-test splits (70:30, 80:20, 90:10). Based on Fig. 6, the parameter that gives the best result in train test split of 70:30 and 80:20 is n_neighbors within the range of 19 to 20 while in train test split of 90:10 is n_neighbors within the range of 16 to 18. For train test ratio of 70:30, the accuracy increases with the number of neighbors, peaking at 78.85% for 'n_neighbors=19' and 'n_neighbors=20'. The precision for class 1 is high with a peak of 0.94 but recall is quite low, which is less than 0.50. Same goes for train test ratio of 80:20, the accuracy increases with the number of neighbors, peaking at 80.16% for 'n_neighbors=19' and 'n_neighbors=20'. There is not much difference between the precision, recall and F1-score across all classes for 'n_neighbor=19' and at 'n_neighbor=20' because it reaches the optimal performance at this point. For the train test ratio of 90:10, the accuracy peaks at 78.24% for 'n_neighbors=16', 'n_neighbors=17' and 'n_neighbors=18'. Among these three numbers of neighbors, 'n_neighbors=16' is considered as the best parameter for KNN model in 90:10 split in terms of a slightly higher precision, recall and F1-score for class 1 and class 2 because these two classes are the focused features in this research. In overall, the higher values of n_neighbors generally improve the accuracy of the model as observed in 70:30 and 80:20 splits. In KNN model, the best performance is given by 'n_neighbors=20' with the highest accuracy of 80.16% for train test split ratio of 80:20.



Train test ratio: 70:30

Parameter	Accuracy (%)	Both Not Exist (0)			Symptom(s) Exist (1)			Treatment(s) Exist (2)		
		Precision	Recall	F1 Score	Precision	Recall	F1 Score	Precision	Recall	F1 Score
n_neighbors=16	77.22	0.74	0.89	0.81	0.90	0.49	0.63	0.81	0.68	0.74
n_neighbors=17	77.66	0.75	0.89	0.81	0.88	0.47	0.61	0.81	0.71	0.76
n_neighbors=18	78.09	0.75	0.88	0.81	0.88	0.49	0.63	0.81	0.72	0.76
n_neighbors=19	78.85	0.76	0.89	0.82	0.92	0.49	0.64	0.83	0.72	0.77
n_neighbors=20	78.85	0.75	0.90	0.82	0.94	0.47	0.62	0.83	0.72	0.77

Train test ratio: 80:20

Parameter	Accuracy (%)	Both Not Exist (0)			Symptom(s) Exist (1)			Treatment(s) Exist (2)		
		Precision	Recall	F1 Score	Precision	Recall	F1 Score	Precision	Recall	F1 Score
n_neighbors=16	78.21	0.75	0.90	0.82	0.83	0.52	0.64	0.83	0.68	0.75
n_neighbors=17	79.02	0.76	0.90	0.83	0.85	0.50	0.63	0.83	0.71	0.77
n_neighbors=18	79.19	0.76	0.90	0.83	0.87	0.48	0.62	0.83	0.71	0.77
n_neighbors=19	80.16	0.77	0.90	0.83	0.88	0.50	0.64	0.84	0.73	0.78
n_neighbors=20	80.16	0.77	0.90	0.83	0.86	0.54	0.66	0.84	0.73	0.78

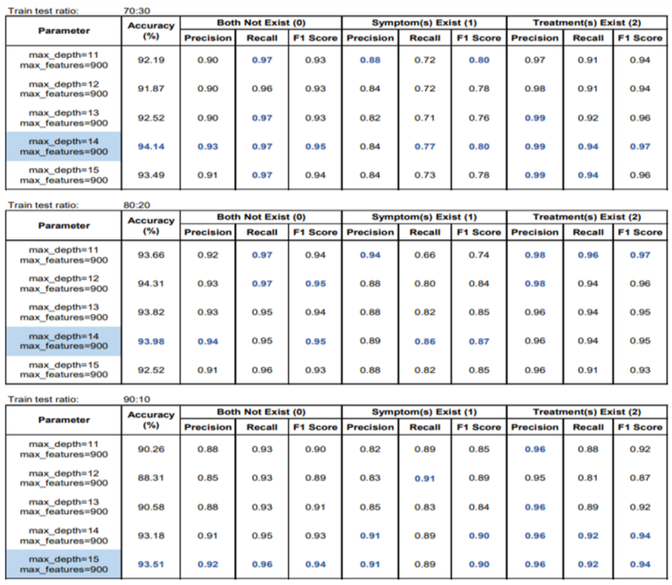
Train test ratio: 90:10

Parameter	Accuracy (%)	Both Not Exist (0)			Symptom(s) Exist (1)			Treatment(s) Exist (2)		
		Precision	Recall	F1 Score	Precision	Recall	F1 Score	Precision	Recall	F1 Score
n_neighbors=16	78.24	0.72	0.92	0.81	0.88	0.63	0.73	0.89	0.65	0.75
n_neighbors=17	78.24	0.73	0.92	0.81	0.91	0.57	0.70	0.86	0.86	0.76
n_neighbors=18	78.24	0.74	0.92	0.82	0.95	0.57	0.71	0.84	0.68	0.75
n_neighbors=19	77.27	0.72	0.90	0.80	0.91	0.60	0.72	0.84	0.66	0.74
n_neighbors=20	77.60	0.73	0.92	0.81	0.91	0.60	0.72	0.85	0.65	0.74

Fig. 6. Results of K-Nearest Neighbor

E. Performance Measure of Random Forest

Fig. 7 shows the performance of Random Forest model trained with balanced class weight, 900 features and different maximum depth of the trees, evaluated on three different train-test splits (70:30, 80:20, 90:10). Based on Fig. 7, the parameter that gives the best result in train test split of 70:30 and 80:20 is max_depth=14 while in train test split of 90:10 is max_depth=15. For the train test ratio of 70:30, the model shows the highest accuracy of 94.14% with averagely highest precision, recall and F1-score across all classes. For train test ratio of 80:20, the highest accuracy is 94.31% when max_depth=12. However, the overall results in terms of all metrics for max_depth=14 is better than max_depth=12 especially for class 1 although it has slightly lower accuracy which is the second highest, 93.98%. For train test ratio of 90:10, the model with max_depth=15 shows the highest accuracy of 93.51%. In this train test ratio, the model with max_depth=14 and max_depth=15 shows similar precision, recall and F1-score for class 1 and class 2 but the accuracy for model with max_depth=15 is still the best across all classes. Generally, the highest accuracy is consistently observed for max_depth=14 across all train-test splits. The accuracy tends to increase with max_depth until it reaches the optimal performance and remains stable. The Random Forest model with train test ratio of 70:30 and max_depth=14 gives the best performance.



Train test ratio: 70:30

Parameter	Accuracy (%)	Both Not Exist (0)			Symptom(s) Exist (1)			Treatment(s) Exist (2)		
		Precision	Recall	F1 Score	Precision	Recall	F1 Score	Precision	Recall	F1 Score
max_depth=11 max_features=900	92.19	0.90	0.97	0.93	0.88	0.72	0.80	0.97	0.91	0.94
max_depth=12 max_features=900	91.87	0.90	0.96	0.93	0.84	0.72	0.78	0.98	0.91	0.94
max_depth=13 max_features=900	92.52	0.90	0.97	0.93	0.82	0.71	0.76	0.99	0.92	0.96
max_depth=14 max_features=900	94.14	0.93	0.97	0.95	0.84	0.77	0.80	0.99	0.94	0.97
max_depth=15 max_features=900	93.49	0.91	0.97	0.94	0.84	0.73	0.78	0.99	0.94	0.96

Train test ratio: 80:20

Parameter	Accuracy (%)	Both Not Exist (0)			Symptom(s) Exist (1)			Treatment(s) Exist (2)		
		Precision	Recall	F1 Score	Precision	Recall	F1 Score	Precision	Recall	F1 Score
max_depth=11 max_features=900	93.66	0.92	0.97	0.94	0.94	0.66	0.74	0.98	0.96	0.97
max_depth=12 max_features=900	94.31	0.93	0.97	0.95	0.88	0.80	0.84	0.98	0.94	0.96
max_depth=13 max_features=900	93.82	0.93	0.95	0.94	0.88	0.82	0.85	0.96	0.94	0.95
max_depth=14 max_features=900	93.98	0.94	0.95	0.95	0.89	0.86	0.87	0.96	0.94	0.95
max_depth=15 max_features=900	92.52	0.91	0.96	0.93	0.88	0.82	0.85	0.96	0.91	0.93

Train test ratio: 90:10

Parameter	Accuracy (%)	Both Not Exist (0)			Symptom(s) Exist (1)			Treatment(s) Exist (2)		
		Precision	Recall	F1 Score	Precision	Recall	F1 Score	Precision	Recall	F1 Score
max_depth=11 max_features=900	90.26	0.88	0.93	0.90	0.82	0.89	0.85	0.96	0.88	0.92
max_depth=12 max_features=900	88.31	0.85	0.93	0.89	0.83	0.91	0.89	0.95	0.81	0.87
max_depth=13 max_features=900	90.58	0.88	0.93	0.91	0.85	0.83	0.84	0.96	0.89	0.92
max_depth=14 max_features=900	93.18	0.91	0.95	0.93	0.91	0.89	0.90	0.96	0.92	0.94
max_depth=15 max_features=900	93.51	0.92	0.96	0.94	0.91	0.89	0.90	0.96	0.92	0.94

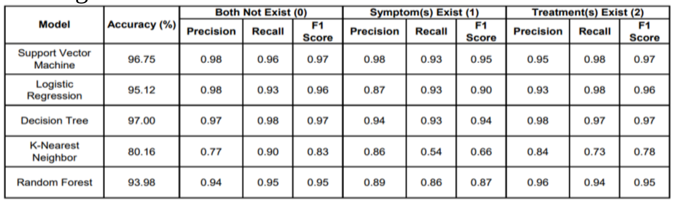
Fig. 7. Results of Random Forest

V. Discussion

According to the performance of the five different machine learning models using different train-test splits described in IV, most of the models such as Support Vector Machine, K-Nearest Neighbor and Decision Tree performed best with train test ratio of 80:20. Therefore, the comparison and the justification between all models will be made based on 80:20 split results.

A. Comparison on Overall Performance

Fig. 8 illustrates the tabulated results for the five machine learning classifiers.



Model	Accuracy (%)	Both Not Exist (0)			Symptom(s) Exist (1)			Treatment(s) Exist (2)		
		Precision	Recall	F1 Score	Precision	Recall	F1 Score	Precision	Recall	F1 Score
Support Vector Machine	96.75	0.98	0.96	0.97	0.98	0.93	0.95	0.95	0.98	0.97
Logistic Regression	95.12	0.98	0.93	0.96	0.87	0.93	0.90	0.93	0.98	0.96
Decision Tree	97.00	0.97	0.98	0.97	0.94	0.93	0.94	0.98	0.97	0.97
K-Nearest Neighbor	80.16	0.77	0.90	0.83	0.86	0.54	0.66	0.84	0.73	0.78
Random Forest	93.98	0.94	0.95	0.95	0.89	0.86	0.87	0.96	0.94	0.95

Fig. 8. Results of all models with 80:20 split

Based on the results in Fig. 8, the Decision Tree has the highest accuracy at 97.00%. SVM and Logistic Regression also have high accuracy, with 96.75% and 95.12% respectively followed by Random Forest with 93.98% accuracy. KNN has the lowest accuracy at 80.16%. To understand the performance of each classifier in detail, the precision, recall and F1-score for each label will be discussed. For “Both Not Exist (0)”, SVM and Logistic Regression show strong performance with F1-score of 0.97 and 0.96 respectively. Decision Tree and Random Forest are also effective with F1-Score of 0.97 and 0.95 respectively. KNN lags behind with an F1-score of 0.83. For “Symptom(s) Exist (1)”, SVM and Decision Tree excel with F1-score of 0.95 and 0.94 respectively while Logistic Regression and Random Forest show slightly lower F1-score of 0.90 and 0.87 respectively. KNN performs poorly with an F1-score of 0.66. For “Treatment(s) Exist (2)”, SVM, Decision Tree, and Logistic Regression show comparable performance with F1-score of 0.97, 0.97, and 0.96 respectively. Random Forest is also strong with an F1-score of 0.95 but KNN again underperforms with an F1-score of 0.78.

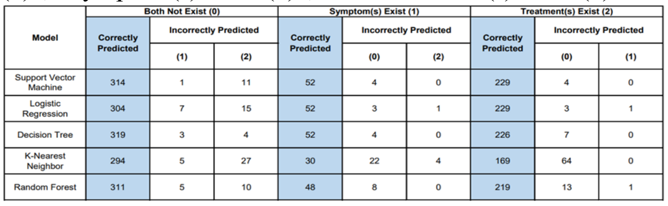
From the results, SVM gives high accuracy and balanced performance across all classes. This might be due to it being able to effectively separate the data points and finding clear boundary between classes. Strong precision and recall, especially in the “Both Not Exist (0)” and “Treatment(s) Exist (2)” classes, indicate that SVM handles these categories well due to its robustness in finding optimal margins. Slightly lower performance in “Symptom(s) Exist (1)” might indicate there is more complexity in distinguishing this class, but SVM still manages it well with an F1-score of 0.95. Moreover, Logistic Regression also shows high accuracy and reliable performance across classes. This means that it can model the decision boundary effectively for this dataset. But same with SVM, the precision and recall for “Symptom(s) Exist (1)” are slightly lower, suggesting that the training data and testing data might be not enough to train the model as compared to the other two labels.

Among the five models, Decision Tree shows the highest accuracy and consistent performance. This is because Decision Tree is effective at capturing the intricacies of the dataset, reflecting in high precision, recall, and f1-score across all classes. Random Forest also shows good accuracy and strong performance for most classes but slightly lower metrics for the “Symptom(s) Exist (1)” class. It handles the dataset well overall, though not as well as Decision Tree or SVM. On the other hand, KNN gives significantly lower accuracy as compared to other models and poor performance for the “Symptom(s) Exist (1)” class. These results might suggest that KNN struggles with the dataset and is likely hindered by high dimensionality and noise in the data, which can mislead the distance calculations and result in poor classification.

In conclusion, SVM and Decision Tree are the top-performing models, demonstrating high accuracy and balanced metrics across all classes. The reasons for getting such results can be lies on the data characteristics, model strengths and model limitations. For the dataset to be used in this research, it is likely having well-defined boundaries for the “Both Not Exist (0)” and “Treatment(s) Exist (2)” classes, making them easier to classify with high precision and recall. The “Symptom(s) Exist (1)” class might be less frequent, making it harder for models to learn and distinguish this class accurately. In terms of model strengths, SVM and Decision Tree excel due to their ability to capture complex patterns. Logistic Regression performed well due to the data’s structure being somewhat linearly separable while KNN struggles with high dimensionality and noise, leading to poor performance. Random Forest balances variance and bias effectively, performing well but rather less effective for “Symptom(s) Exist (1)”.

B. Comparison on Confusion Matrix

Fig. 9 illustrates the tabulated confusion matrix results for all five machine learning classifiers: Support Vector Machine (SVM), Logistic Regression, Decision Tree, K-Nearest Neighbor (KNN), and Random Forest. The table provides predictions for three different classes (labels): “Both Not Exist (0)”, “Symptom(s) Exist (1)”, and “Treatment(s) Exist (2)”.



Model	Both Not Exist (0)			Symptom(s) Exist (1)			Treatment(s) Exist (2)		
	Correctly Predicted	Incorrectly Predicted		Correctly Predicted	Incorrectly Predicted		Correctly Predicted	Incorrectly Predicted	
		(1)	(2)		(0)	(2)		(0)	(1)
Support Vector Machine	314	1	11	52	4	0	229	4	0
Logistic Regression	304	7	15	52	3	1	229	3	1
Decision Tree	319	3	4	52	4	0	226	7	0
K-Nearest Neighbor	294	5	27	30	22	4	169	64	0
Random Forest	311	5	10	48	8	0	219	13	1

Fig. 9. Confusion Matrix of all models with 80:20 split

According to Fig. 9, SVM has a high accuracy in predicting class 0 with 314 correctly predicted instances and class 2 with 229 correctly predicted instances but has a few misclassifications for class 1 with 4 instances misclassified as “Both Not Exist (0)”. The model shows strong performance for class 2 with no misclassification into class 1. Logistic Regression also shows high accuracy, particularly in predicting class 2 with 229 correctly predicted instances. However, it has more misclassifications for class 0 with 15 instances misclassified as “Treatment(s) Exist (2)” compared to SVM. The Decision Tree model performs well in predicting class 0 with 319 correctly predicted instances and class 2 with 226 correctly predicted instances. It has fewer misclassifications for class 0 compared to SVM and Logistic Regression. KNN has the lowest performance among the models, especially for class 2, where 64 instances are misclassified as “Both Not Exist (0)”. It also struggles with class 1 predictions, with a significant number of instances, 22 instances misclassified as “Both Not Exist (0)”. Random Forest performs well overall but has some misclassifications for class 2, where 13 instances are misclassified as “Both Not Exist (0)”. It has better accuracy for class 1 compared to KNN but is not as accurate as SVM or Logistic Regression. In summary, SVM and Logistic Regression models show the best overall performance, with high accuracy in predicting all labels and minimal misclassifications based on confusion matrix results. KNN has the highest number of misclassifications, especially for class 2, indicating that it may not be the best model for this dataset.

VI. CONCLUSION

In conclusion, most of the classifiers show their best results in the text classification on DM symptoms and treatments using journal articles from PubMed website when the splitting is 80% for training data and 20% for testing data. By comparing the results, SVM was proven as the best model in terms of performance metrics and confusion matrix followed by Decision Tree, Logistic Regression and Random Forest. The worst performing model fell to the KNN model. Although majority of the proposed machine learning algorithms have been proved that it is capable to be employed to classify text data based on the keywords defined for DM, there are some suggestions for future improvements includes trying different types of feature extraction methods, implementation of hyperparameter tuning using search techniques and implementation of imbalance data handling technique such as MLSMOTE to handle data imbalance problems in multi-label text data.

ACKNOWLEDGEMENT

The author would like to express gratitude and appreciation to Faculty of Computing, Universiti Teknologi Malaysia (UTM) for their support in this research.

REFERENCES

[1] Sapra, A. and Bhandari, P. (2022). *Diabetes Mellitus*. In Starpearls [Internet]. Treasure Island (FL): Starpearls.

[2] Banday, M. Z., Sameer, A. S. and Nissar, S. (2020). Pathophysiology of diabetes: An overview. *Avicenna journal of medicine*, 10(4), pp. 174–188.

[3] Dogra, V., Verma, S., Kavita, Chatterjee, P., Shafi, J., Choi, J., and Ijaz, M. F. (2022). A complete process of text classification system using state-of-the-art NLP models. *Computational Intelligence and Neuroscience*, pp. 1–26.

[4] Wahdan, A., Hantoobi, S., Salloum, S. A. and Shaalan, K. (2020). A systematic review of text classification research based on deep learning models in Arabic language. *International Journal of Electrical and Computer Engineering (IJECE)*, 10(6), pp. 6629-6643.

[5] Kowsari, K, Meimandi, K. J., Heidarysafa, M., Mendu, S., Barnes, L. and Brown, D. (2019). Text Classification Algorithms: A Survey. *Information*, 10(4), p. 150.

[6] Kadhim, A. I. (2019). Survey on supervised machine learning techniques for automatic text classification. *Artificial Intelligent Review*, 52, pp. 273–292.

[7] Chowdhury, S. and Schoen, M. P. (2020). Research Paper Classification using Supervised Machine Learning Techniques. *2020 Intermountain Engineering, Technology and Computing (IETC)*, pp. 1-6.

[8] Zhang, M. -L. and Zhou, Z. -H. (2014). A Review on Multi-Label Learning Algorithms. *EEE Transactions on Knowledge and Data Engineering*, 26(8), pp.1819-1837.

[9] Shaikh, R., Rafi, M., Mahoto, N. A., Sulaiman, A. and Shaikh, A. (2023). A filter-based feature selection approach in multilabel classification. *Machine Learning: Science and Technology*, 4(4).

[10] Bogatinovski, J., Todorovski, L., Džeroski, S. and Kocev, D. (2022). Comprehensive comparative study of multi-label classification methods. *Expert Systems with Applications*, 203.

[11] Hassan, S. U., Ahamed, J. and Ahmad, K. (2022). Analytics of machine learning-based algorithms for text classification. *Sustainable Operations and Computers*, 3, pp. 238-248.

[12] Ślusarczyk, B. (2018). Industry 4.0: are we ready?. *Polish Journal of Management Studies*, 17(1), pp. 232–248.

[13] Mohammed, M., Khan, M. B. and Bashier, E. B. M. (2017) *Machine Learning: Algorithms and Applications*. Boca Raton: CRC Press.

[14] Rasheed, I., Gupta, V., Banka, H. and Kumar, C. (2018). Urdu Text Classification: A comparative study using machine learning techniques. *2018 Thirteenth International Conference on Digital Information Management (ICDIM)*, pp. 274-278.

[15] Singh, N. K. and Chand, S. (2022). Machine Learning-based Multilabel Toxic Comment Classification. *2022 International Conference on Computing, Communication, and Intelligent Systems (ICCCIS)*, pp. 435-539.

[16] Rahul, Kajla. H., Hooda, J. and Saini, G. (2020). Classification of Online Toxic Comments Using Machine Learning Algorithms. *2020 4th International Conference on Intelligent Computing and Control Systems (ICICCS)*, pp.1119-1123.