**SYMPTOM-BASED DISEASE CLASSIFICATION USING**

**TF-IDF AND KNN**

**A MINI PROJECT REPORT**

***Submitted by***

**LINDA CHRISTINA F 312322205091**

**PAVITHRA G R 312322205123**

***in partial fulfillment for the award of the degree of***

**BACHELOR OF TECHNOLOGY**

**IN INFORMATION TECHNOLOGY**



**St. JOSEPH’S COLLEGE OF ENGINEERING**

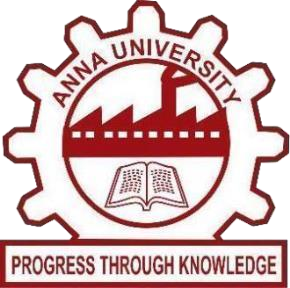
**(An Autonomous Institution) St. Joseph’s Group of Institution**

**OMR, Chennai 600 119**

**ANNA UNIVERSITY: CHENNAI 600 025**

**April - 2025**

**ANNA UNIVERSITY: CHENNAI 600 025**



**BONAFIDE CERTIFICATE**

Certified that this mini project report **“Symptom-Based Disease Classification Using TF-IDF and KNN”** is the bonafide work of **LINDA CHRSTINA F (312321205091)** and **PAVITHRA G R (312321205123)** who carried out the phase I project under my supervision, for the partial fulfillment of the requirements for the award of the degree of Bachelor of Technology in Information Technology.

|  |  |
| --- | --- |
| **SIGNATURE SUPERVISOR**  Dr. J. DIVYA, M.Tech., Ph.D.  Assistant Professor,  Department of Information Technology, St. Joseph’s College of Engineering, OMR, Chennai- 600119. | **SIGNATURE**  **HEAD OF THE DEPARTMENT**  Dr. HELTIN GENITA C, M.E., Ph.D.,  Professor,  Department of Information Technology, St. Joseph’s College of Engineering, OMR, Chennai- 600119. |

CERTIFICATE OF EVALUATION

**College name :** St. Joseph’s College of Engineering

**Branch :** Information Technology

**Semester :** VI

|  |  |  |  |
| --- | --- | --- | --- |
| **SL.NO** | **NAME OF THE STUDENTS** | **TITLE OF THE PROJECT** | **NAME OF THE SUPERVISOR** |
| 1.  2. | LINDA CHRISTINA F  (312322205091)  PAVITHRA G R (312322205123) | Symptom-Based Disease Classification Using TF-IDF and KNN | Dr. DIVYA J |

The report of the Phase I project work submitted by the above students in partial fulfillment for the award of Bachelor of Technology Degree in Information Technology of Anna University were evaluated and confirmed to be reports of the work done by the above students.

Submitted for project and Viva Examination held on .

**INTERNAL EXAMINER EXTERNAL EXAMINER**

**ACKNOWLEDGEMENT**

At the outset we would like to express our sincere gratitude to the beloved **Chairman, Dr. Babu Manoharan, M.A., M.B.A., Ph.D.,** for his constant guidance and support.

We would like to express our heartfelt thanks to our respected **Managing Director,,Mr. B. Shashi Sekar, M.Sc.,** for her kind encouragement and blessings.

We wish to express our sincere thanks to our **Executive Director Mrs. S. Jessie Priya, M.Com.,** for providing ample facilities in the institution.

We express our deepest gratitude and thanks to our beloved **Principal Dr. Vaddi Seshagiri Rao, B.E., M.E., M.B.A., Ph.D., F.I.E.,** for his inspirational ideas during the project.

We are immensely grateful to our esteemed **Dean School of Computing, Mrs. G. Lathaselvi, B.E., M.E., (Ph.D.),** for her invaluable support and academic guidance throughout this endeavor.

We wish to express our sincere thanks and gratitude to **Dr.Heltin Genitha M.E., Ph.D., Head of the department,** Department of Information Technology, St. Joseph’s College of Engineering for her guidance and assistance in solving the various intricacies involved in the project.

It is with deep sense of gratitude that we acknowledge our supervisor, **Dr Divya J M.Tech , Ph.D.** for her expert guidance and connoisseur suggestion.

Finally, we thank our department staff members who helped us in the successful completion of this project.

**ABSTRACT**

Timely and accurate disease identification based on patient-reported symptoms is critical for effective medical intervention, particularly in resource-limited settings. However, many existing diagnostic systems rely on structured data or manual feature engineering, which may not effectively capture the nuances of natural language symptom descriptions. This research proposes a machine learning-based framework that leverages text mining and classification techniques to map unstructured symptom inputs to their corresponding diseases. The system employs Term Frequency–Inverse Document Frequency (TF-IDF) for vectorizing symptom text, transforming it into meaningful numerical features. The K-Nearest Neighbors (KNN) algorithm is then applied to classify the input based on similarity to labeled symptom-disease datasets. To address challenges such as class imbalance, data sparsity, and semantic overlap between symptom descriptions, preprocessing steps including stop-word removal, stemming, and normalization are performed. Additionally, oversampling techniques are explored to enhance model generalization. The model’s performance is evaluated using metrics such as accuracy, precision, recall, and F1-score to ensure reliability across diverse conditions. The proposed system demonstrates promising results in predicting diseases from free-text symptom inputs, offering a lightweight, explainable, and accessible decision-support tool for preliminary diagnosis and healthcare triage.

|  |  |  |
| --- | --- | --- |
|  | **TABLE OF CONTENTS** |  |
| **CHAPTER** | **TITLE** | **PAGE NO** |
|  | ABSTRACT | vi |
|  | LIST OF FIGURES | vii |
|  | LIST OF TABLES | vii |
|  | LIST OF ABBREVIATIONS | viii |
| **1** | **INTRODUCTION** | 1 |
| **2** | **LITERATURE SURVEY** | 2 |
| **3** | **SYSTEM ANALYSIS** | 3 |
|  | 3.1 EXISTING SYSTEM | 3 |
|  | 3.2 PROPOSED SYSTEM | 3 |
| **4** | **SYSTEM DESIGN** | 5 |
|  | 4.1 ARCHITECTURE | 5 |
|  | 4.2 SYSTEM FLOW | 6 |
|  | 4.3 USE CASE DIAGRAM | 7 |
|  | 4.4 SEQUENCE DIAGRAM | 8 |
|  | 4.5 ACTIVITY DIAGRAM | 9 |
|  | 4.6 CLASS DIAGRAM | 10 |
| **5** | **SYSTEM IMPLEMENTATION** | 11 |
|  | 5.1 MODEL DESCRIPTION | 11 |
|  | 5.2 METHODOLOGIES | 11 |
| **6** | **CONCLUSION AND FUTURE SCOPE** | 12 |
|  | 6.1 APPENDICES | 13 |
|  | 6.2 REFERENCES | 21 |

|  |  |  |
| --- | --- | --- |
|  | **LIST OF FIGURES** |  |
| **FIGURE NO** | **TITLE** | **PAGE NO** |
| 3.1 | PROPOSED SYSTEM ARCHITECTURE | 5 |
| 3.2 | FLOW DIAGRAM OF THE SYSTEM | 6 |
| 3.3 | USE CASE DIAGRAM | 7 |
| 3.4 | SEQUENCE DIAGRAM | 8 |
| 3.5 | ACTIVITYDIAGRAM | 9 |
| 3.6 | CLASS DIAGRAM | 10 |

|  |  |  |
| --- | --- | --- |
|  | **LIST OF TABLES** |  |
| **TABLE NO** | **TITLE** | **PAGE NO** |
| 1 | SYSTEM COMPARISON | 3 |

**LIST OF ABBREVIATIONS**

|  |  |
| --- | --- |
| **ABBREVIATION** | **DEFINITION** |
| ML | Machine Learning |
| TF-IDF | Term Frequency–Inverse Document Frequency |
| NLP | Natural Language Processing |
| KNN | K-Nearest Neighbors |

**CHAPTER 1 INTRODUCTION**

In modern healthcare, early and accurate diagnosis based on patient’s descriptions of symptoms is vital to initiate timely treatment and improve patient outcomes. However, a major challenge in clinical diagnosis is that patients often describe their symptoms in free-form, unstructured text, which cannot be easily interpreted by traditional automated systems. Most existing diagnostic models are designed to work with structured data such as predefined symptom checklists, numeric test results, or categorical values. While structured data provides consistency, it fails to capture the subtle variations and nuances present in natural language symptom descriptions, leading to potential loss of information during the diagnostic process.

To bridge this gap, this project adopts a text classification approach that utilizes Natural Language Processing (NLP) and machine learning techniques to interpret unstructured symptom text and classify it into one of several disease categories. The two core techniques employed in this system are Term Frequency–Inverse Document Frequency (TF-IDF) and K-Nearest Neighbors (KNN).

TF-IDF is a statistical method that transforms raw text into meaningful numerical vectors. It works by evaluating how important a word is in a document relative to the entire dataset. Words that frequently occur in a specific document but are rare across the entire corpus are assigned higher weights. This helps the system focus on disease-relevant symptoms while downplaying generic or commonly occurring words that do not contribute significantly to classification.

Once the symptom text has been vectorized using TF-IDF, the K-Nearest Neighbors algorithm is applied for classification. KNN is a simple, non-parametric algorithm that classifies new input based on the majority label among its ‘k’ closest neighbors in the feature space. This makes the approach highly interpretable and effective, especially for datasets where the relationships between samples are more meaningful than the parameters of a learned model.

**CHAPTER 2 LITERATURE SURVEY**

In recent years, machine learning has played an increasingly prominent role in healthcare, particularly in automating diagnosis and symptom classification tasks. A wide range of algorithms and models have been explored for disease prediction based on patient input, especially where data is in textual form. Traditional healthcare diagnostic systems often rely on manual feature engineering, rule-based decision trees, or structured data input. These systems are limited by their rigid data formats and an inability to handle the variability and richness found in natural language symptom descriptions provided by patients.

To overcome these limitations, researchers have increasingly turned to Natural Language Processing (NLP) techniques to process and understand free-text input. One such method is TF-IDF (Term Frequency–Inverse Document Frequency), a statistical approach that assigns weights to words based on their frequency in a document relative to the entire dataset. This technique has proven effective in emphasizing disease-specific terms while downplaying commonly used non-informative words. As a result, TF-IDF has become a widely used tool for converting symptom descriptions into numerical feature vectors suitable for machine learning algorithms.

Alongside TF-IDF, other representation methods such as word embeddings (e.g., Word2Vec, GloVe) have also been used to capture semantic meaning and context in symptom narratives. These techniques map words into dense vectors based on their contextual similarity and co-occurrence, enabling models to understand relationships between symptoms that are not explicitly identical but have similar meanings.

Among the classifiers employed, K-Nearest Neighbors (KNN) stands out for its simplicity, transparency, and effectiveness in text classification tasks. KNN does not require a complex training phase and classifies new instances based on their proximity to labeled data in the feature space. When combined with a robust feature representation such as TF-IDF, KNN has shown high accuracy in predicting diseases from textual symptom data, making it an attractive option for low-resource implementations.

**CHAPTER 3 SYSTEM ANALYSIS**

* 1. **EXISTING SYSTEM**

Traditional diagnostic systems in healthcare often rely on structured data inputs, such as checkboxes, dropdown menus, or manually crafted features. These systems require users to conform to predefined symptom lists, which limits flexibility and fails to capture the real-world way in which patients describe their health issues. Such models are primarily rule-based or depend heavily on manual feature engineering, which can be time-consuming and may not generalize well across diverse patient populations or evolving language patterns.

Moreover, existing systems are typically built for specific conditions or contexts and lack adaptability. They do not support free-text input, which is essential for capturing the full depth and variability of patient symptom narratives. This rigid structure reduces diagnostic accuracy and user engagement, especially in environments where users may not be familiar with medical terminology.

These systems also suffer from scalability issues. When deployed in real-time applications—such as chatbots, telemedicine, or mobile diagnostic apps—they may fail to interpret non-standard language inputs or variations in phrasing. In resource-limited settings, where access to high-end computational infrastructure and expert-designed systems is minimal, traditional models often become unusable. Additionally, the interpretability of many black-box models can be limited, making them less suitable for use in critical domains like healthcare, where trust and transparency are essential.

* 1. **PROPOSED SYSTEM**

To address the limitations of traditional systems, the proposed system introduces a machine learning-based framework that uses TF-IDF vectorization and K-Nearest Neighbors (KNN) classification for disease prediction based on unstructured symptom descriptions. The goal is to provide a lightweight, interpretable, and effective method for mapping free-text symptom inputs to likely disease categories.

The system begins with text preprocessing, a crucial step that includes the removal of stopwords (commonly used but non-informative words), lemmatization (reducing words to their base form), punctuation removal, and conversion to lowercase. This ensures the input data is clean, consistent, and ready for vectorization.

Once preprocessed, the symptom descriptions are transformed into numerical vectors using the Term Frequency–Inverse Document Frequency (TF-IDF) technique. TF-IDF weighs words based on their importance in a document relative to a collection of documents, effectively capturing the relevance of each term in the context of the dataset.

The vectorized input is then classified using the K-Nearest Neighbors (KNN) algorithm. KNN is an instance-based learning method that classifies new samples by analyzing the 'k' most similar data points from the training set. It is particularly useful for its simplicity and effectiveness in text classification tasks, especially when combined with meaningful feature representations like TF-IDF.

To enhance the model’s performance, Grid Search and Cross-Validation are applied for hyperparameter tuning, ensuring the most suitable value of 'k' is selected. This helps in avoiding overfitting and underfitting, improving the model’s generalization capability on unseen data.

**CHAPTER 4 SYSTEM DESIGN**

* 1. **ARCHITECTURE DIAGRAM**

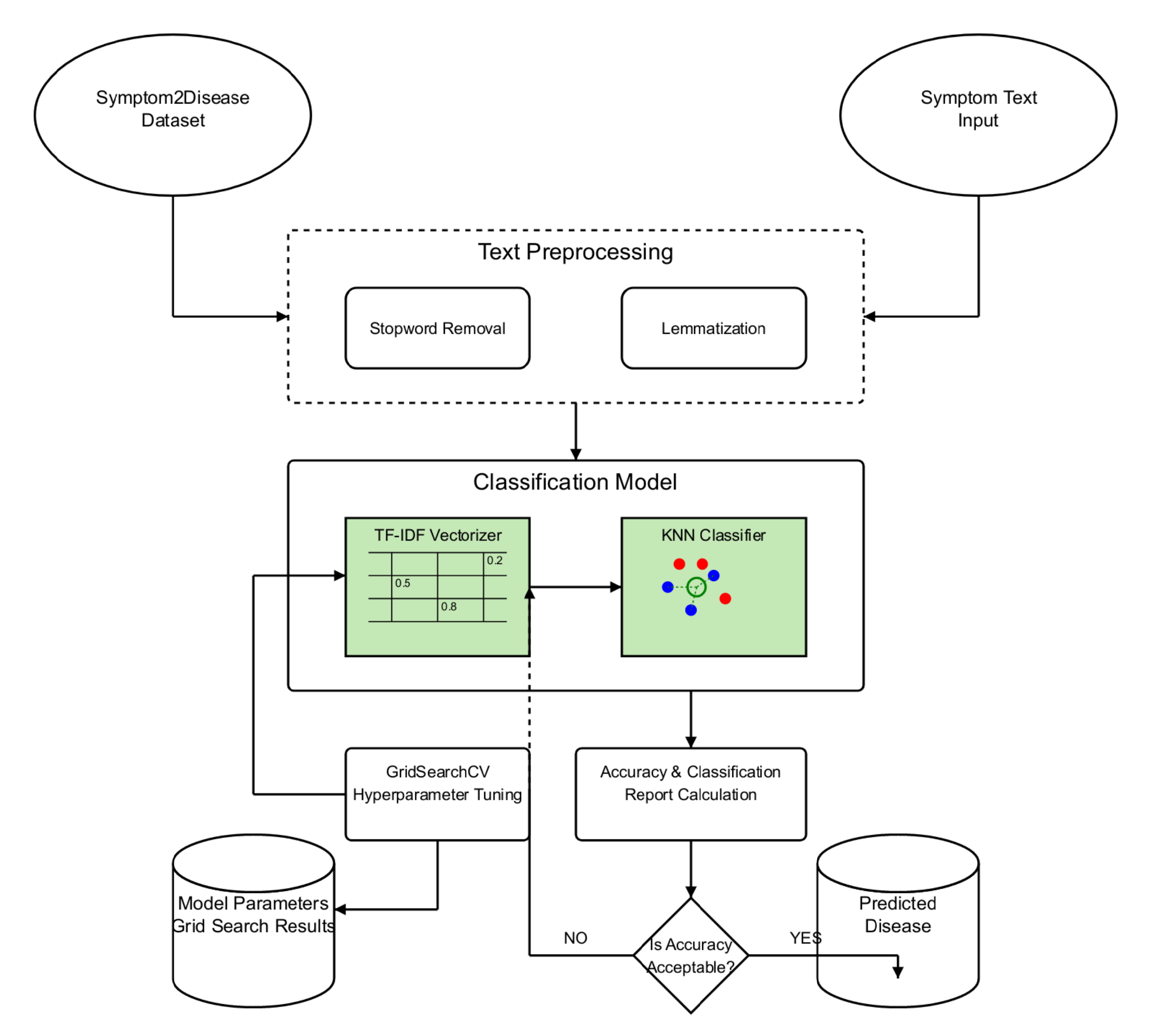


Figure 4.1 Architecture diagram

* 1. **SYSTEM FLOW DIAGRAM**

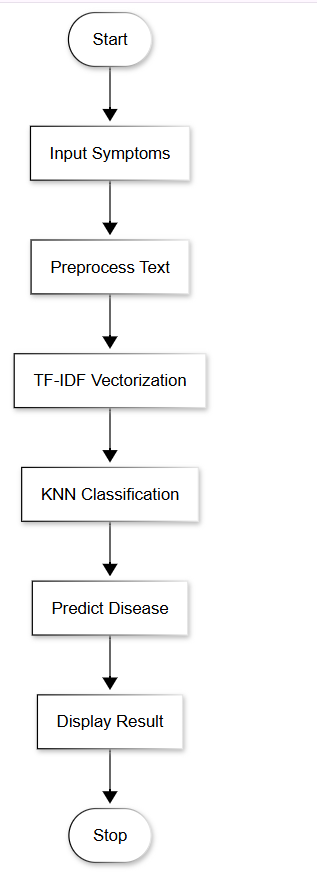


Fig: 4.2: System Flow Diagram

* 1. **USE CASE DIAGRAM**

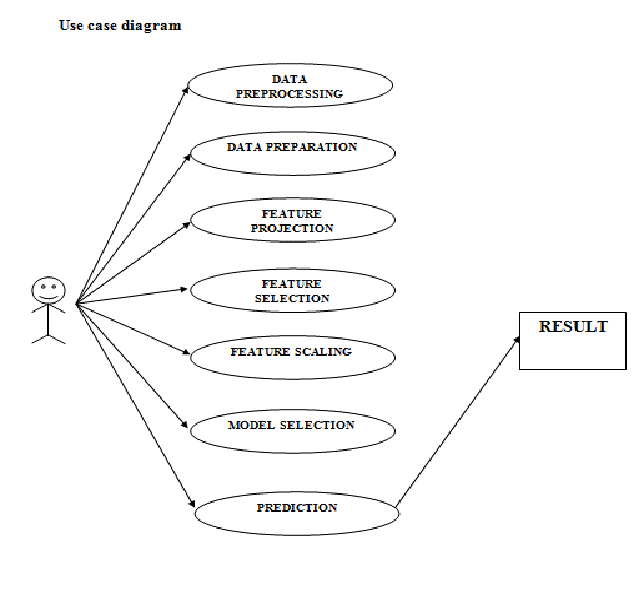


Figure 4.3 Use case diagram

* 1. **SEQUENCE DIAGRAM**

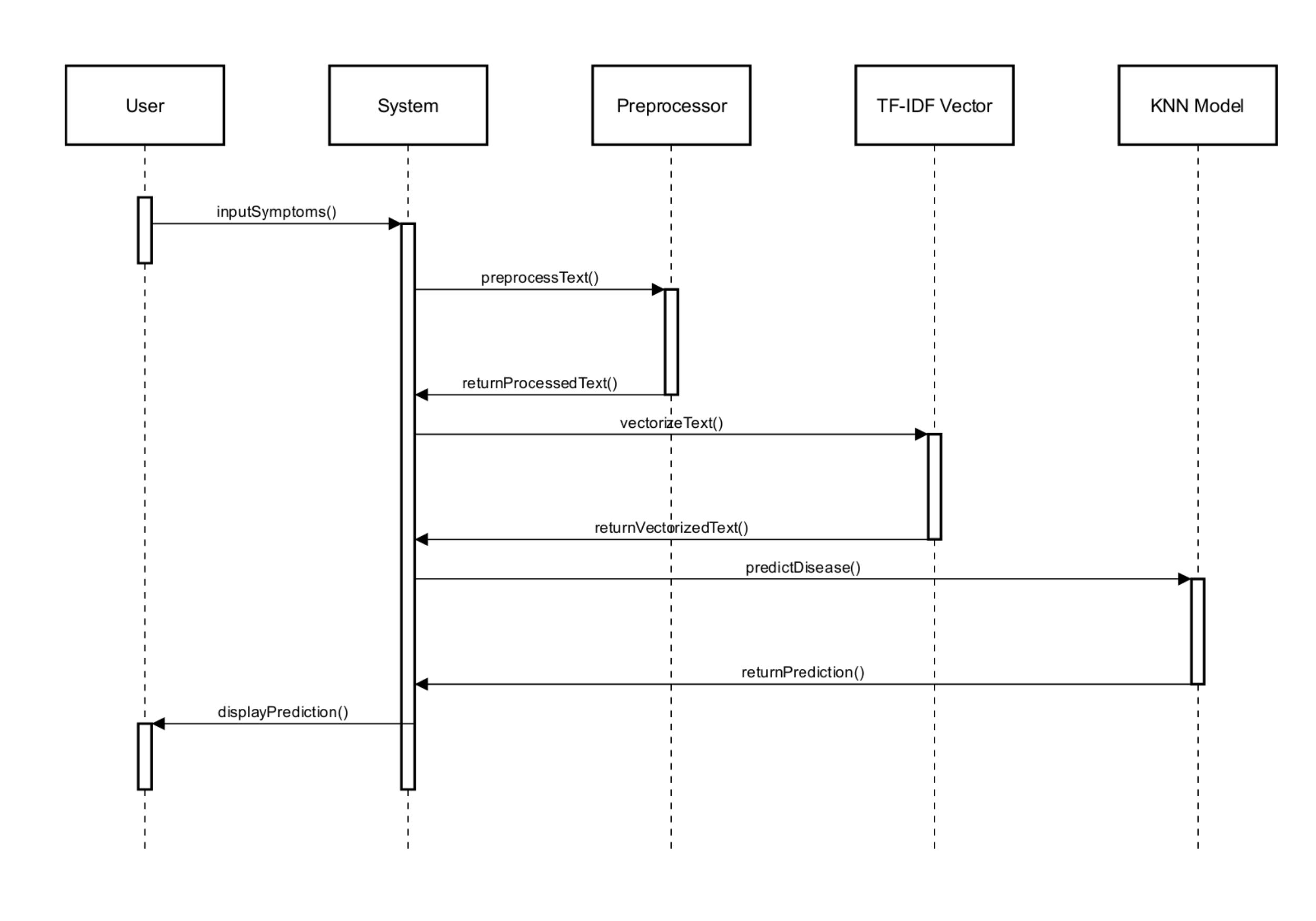


Figure 4.4 Sequence diagram

* 1. **ACTIVITY DIAGRAM**

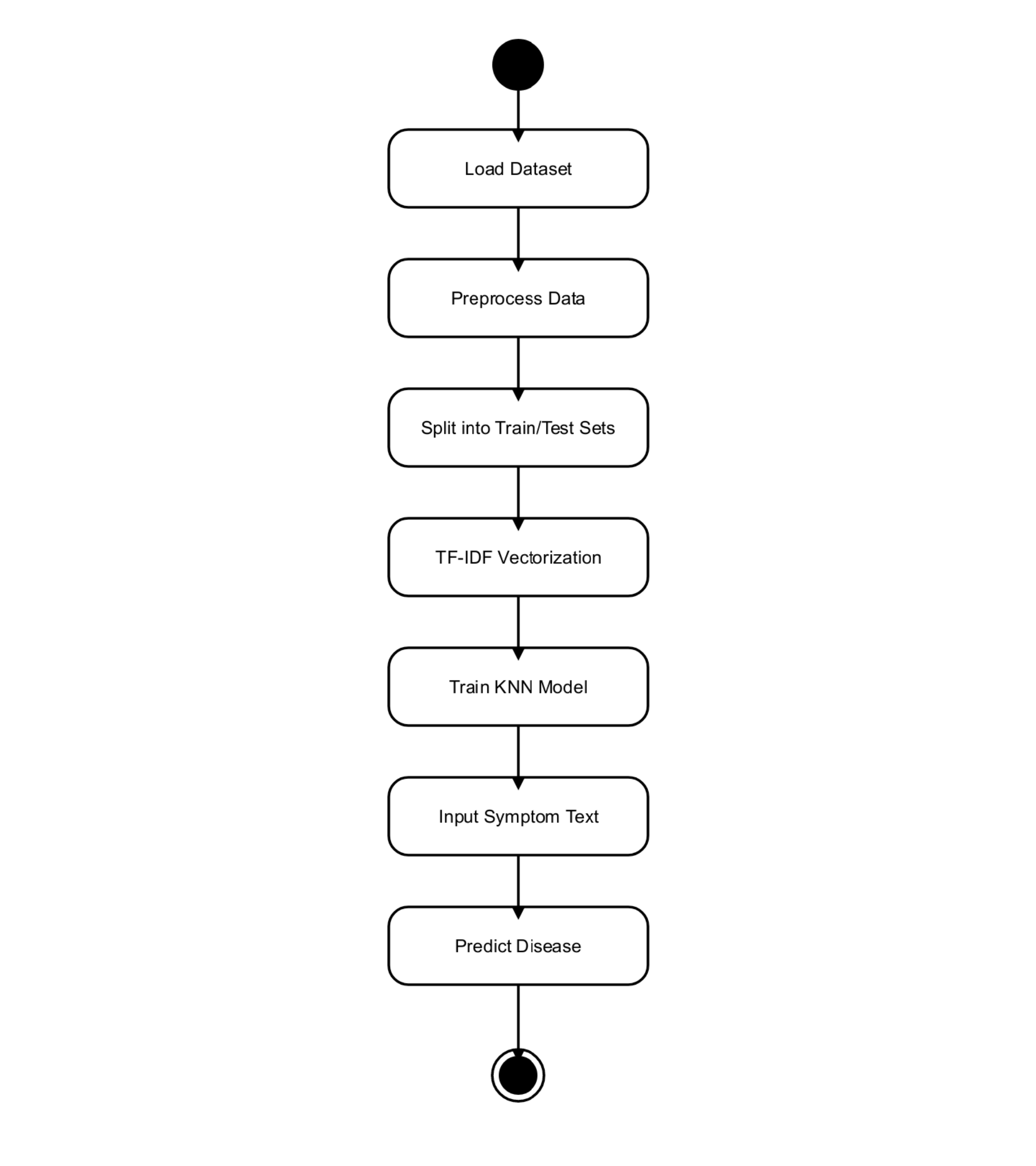


Figure 4.5 Activity diagram

**CHAPTER 5 SYSTEM IMPLEMENTATION**

* 1. **MODEL DESCRIPTION**

The model developed in this project aims to predict diseases based on natural language symptom descriptions using a combination of TF-IDF vectorization and the K-Nearest Neighbors (KNN) algorithm. The core strength of this system lies in its ability to process unstructured textual data—such as free-form patient symptom inputs—and transform it into structured numerical representations that a machine learning model can understand and classify effectively.

The dataset used for this project comprises 1200 records, stored in a CSV file format. Each record contains:

* **text**: a free-text symptom description written in natural language.
* **label**: the associated disease diagnosis for that symptom description.

The dataset includes 24 unique diseases, such as Migraine, Diabetes, Typhoid, Chickenpox, Pneumonia, Fungal Infection, Psoriasis, Dengue, and more. The goal of the model is to correctly predict the disease (label) when given a new, unseen symptom description (text).

* 1. **METHODOLOGIES**

# **5.2.1 PREPROCESSING AND FEATURE ENGINEERING**

The pipeline begins with text preprocessing, which is crucial to ensuring consistent, clean, and informative inputs to the model. This involves:

* **Stopword Removal**: Commonly used but semantically insignificant words are removed.
* **Punctuation Removal**: Symbols and special characters are stripped from the text to reduce noise.
* **Lowercasing**: All characters are converted to lowercase to maintain uniformity.
* **Lemmatization**: Words are reduced to their base or root form to avoid duplication and preserve semantic meaning.

This preprocessing pipeline ensures that all symptom descriptions are represented in a standard format, improving the quality of feature extraction.

# **5.2.2 FEATURE EXTRACTION USING TF-IDF**

After preprocessing, the cleaned text is transformed into numerical vectors using the Term Frequency–Inverse Document Frequency (TF-IDF) method. TF-IDF helps quantify how important a word is in a given symptom description relative to the entire dataset. This ensures that more relevant and disease-indicative terms are given higher importance, while generic terms receive lower weights.

The TF-IDF vectorizer converts each symptom description into a high-dimensional sparse vector, where each dimension represents a unique word from the corpus. These vectors are then used as input features for classification.

**5.2.3 CLASSIFICATION USING K-NEAREST NEIGHBORS (KNN)**

The classification task is handled by the K-Nearest Neighbors (KNN) algorithm. KNN is a distance-based, non-parametric method that classifies new instances based on the majority class among its ‘k’ closest neighbors in the training data. It is particularly well-suited for text classification when combined with meaningful feature representations like TF-IDF.

To optimize the model’s performance, Grid Search is employed to determine the best value of k, and Cross-Validation is used to ensure the model generalizes well across different folds of the dataset.

# **MISCLASSIFICATION RATE CALCULATION:**

The rate of misclassification for any classification that arises from a dataset can be found by observing confusion matrix metrics. Normally, the confusion matrix, which summarizes performances of the classifications done by a model, is used in this regard. Put differently, the misclassification rate, sometimes also called an error rate. Or, to be more precise, the misclassification rate is simply the sum of false negative and false positive which is divided by the total number of samples. The formula for the misclassification rate looks as follows:

**CHAPTER 6**

**CONCLUSION AND FUTURE SCOPE**

This project demonstrates a simple yet effective machine learning approach for disease prediction using natural language symptom descriptions. By leveraging TF-IDF for feature extraction and K-Nearest Neighbors (KNN) for classification, the system successfully transforms unstructured symptom text into accurate disease predictions. Preprocessing techniques such as stopword removal, lemmatization, punctuation removal, and lowercasing were crucial in standardizing the input data and enhancing model performance.

The model was trained and evaluated on a dataset of 1200 entries spanning 24 diseases, achieving a high classification accuracy of 93.611%, with a corresponding misclassification rate of only 6.389%. Performance metrics such as precision, recall, and F1-score further validate the system’s reliability and robustness. Grid Search and cross-validation ensured the model was optimally tuned for generalization.

In future enhancements, the system can be extended to support multi-label classification, which is crucial in real-world scenarios where a single symptom description may correspond to multiple possible diseases. This would increase the model’s practical applicability in clinical settings. Additionally, replacing or augmenting the TF-IDF vectorization with more context-aware word embedding techniques such as Word2Vec, GloVe, or BERT could enable the model to capture deeper semantic relationships within symptom descriptions, thereby improving classification accuracy and robustness. Another important direction is the inclusion of larger and more diverse real-world datasets. Expanding the dataset with clinical records from varied sources would enhance the model’s ability to generalize across different patient populations and symptom expressions, making it more reliable for deployment in real healthcare environments.

**APPENDICES**

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

from wordcloud import WordCloud

import os

for dirname, \_, filenames in os.walk('/kaggle/input'):

    for filename in filenames:

        print(os.path.join(dirname, filename))

symptom\_df = pd.read\_csv('/content/Symptom2Disease.csv')

symptom\_df['label'] = symptom\_df['label'].apply(str.title)

symptom\_df.head(10)

symptom\_df = symptom\_df.drop('Unnamed: 0', axis=1)

symptom\_df

# Exploratory Data Analysis

# Distribution Frequency of Diseases

disease\_counts = symptom\_df['label'].value\_counts()

plt.figure(figsize=(12, 6))

sns.barplot(x=disease\_counts.index, y=disease\_counts.values, palette='viridis')

plt.title('Distribution of Diseases', fontsize=16)

plt.xlabel('Disease', fontsize=12)

plt.ylabel('Number of Symptom Descriptions', fontsize=12)

plt.xticks(rotation=90)

plt.show()

# Word Cloud of Different Diseases

disease = 'Psoriasis'

disease\_text = " ".join(symptom\_df[symptom\_df['label'] == disease]['text'])

print(f"Text for {disease}: {disease\_text[:200]}...")  # Print first 200 characters

# Generate the word cloud

wordcloud = WordCloud(width=800, height=400, background\_color='white').generate(disease\_text)

# Plot the word cloud

plt.figure(figsize=(10, 5))

plt.imshow(wordcloud, interpolation='bilinear')

plt.axis('off')  # No axes for word cloud

plt.title(f'Word Cloud for {disease}', fontsize=16)

plt.show()

# Preprocessing

from nltk.stem import WordNetLemmatizer

from nltk.corpus import wordnet

from nltk.tokenize import word\_tokenize

from nltk import pos\_tag

from nltk.corpus import stopwords

from textblob import TextBlob

import nltk

!unzip /usr/share/nltk\_data/corpora/wordnet.zip -d /usr/share/nltk\_data/corpora/

import string

import nltk

nltk.download('stopwords')

stop\_words = set(stopwords.words('english'))

lemmatizer = WordNetLemmatizer()

# Function to get the WordNet POS tag from NLTK POS tag

def get\_wordnet\_pos(treebank\_tag):

    if treebank\_tag.startswith('J'):

        return wordnet.ADJ

    elif treebank\_tag.startswith('V'):

        return wordnet.VERB

    elif treebank\_tag.startswith('N'):

        return wordnet.NOUN

    elif treebank\_tag.startswith('R'):

        return wordnet.ADV

    else:

        return wordnet.NOUN

# Function to lemmatize a single text

def lemmatize\_text(text):

    tokens = word\_tokenize(text)

    tagged\_tokens = pos\_tag(tokens)

    lemmatized\_tokens = [lemmatizer.lemmatize(word, get\_wordnet\_pos(pos)) for word, pos in tagged\_tokens]

    return " ".join(lemmatized\_tokens)

# Function to remove common stopwords

def remove\_stopwords(text):

    words = word\_tokenize(text)

    filtered\_words = [word for word in words if word.lower() not in stop\_words]

    return " ".join(filtered\_words)

# Function to remove punctuations and make texts lowercase

def remove\_punctuations\_and\_lowercase(text):

    # Convert text to lowercase

    text = text.lower()

    # Remove punctuation

    text = text.translate(str.maketrans('', '', string.punctuation))

    return text

# Function that encapsulates the other preprocessing techniques

def preprocessing(text):

    removed\_stopwords = remove\_stopwords(text)

    lemmatized = lemmatize\_text(removed\_stopwords)

    finalised\_text = remove\_punctuations\_and\_lowercase(lemmatized)

    return finalised\_text

import nltk

nltk.download('punkt\_tab')

nltk.download('averaged\_perceptron\_tagger\_eng')

nltk.download('wordnet')

nltk.download('omw-1.4')

symptom\_df\_w2v = symptom\_df.copy()

symptom\_df\_w2v['text'] = symptom\_df['text'].apply(preprocessing)

symptom\_df\_w2v

# NLP Models Creation

#TF-IDF Vectorization with KNN Classification

from sklearn.neighbors import KNeighborsClassifier

from sklearn.feature\_extraction.text import TfidfVectorizer

from sklearn.model\_selection import GridSearchCV

from sklearn.model\_selection import train\_test\_split

tfidf\_vectorizer = TfidfVectorizer(max\_features=1500)

X\_train\_text, X\_test\_text, y\_train, y\_test = train\_test\_split(symptom\_df\_w2v['text'],

                                                              symptom\_df['label'],

                                                              stratify=symptom\_df['label'],

                                                              test\_size=0.3,

                                                              random\_state=42)

X\_train = tfidf\_vectorizer.fit\_transform(X\_train\_text)

X\_test = tfidf\_vectorizer.transform(X\_test\_text)

# Cross Validation with Grid Search to Determine the Best Model Parameters

param\_grid = {

    'n\_neighbors': [3, 5, 7, 10],  # Number of neighbors to test

    'weights': ['uniform', 'distance'],  # Weight function used in prediction

    'algorithm': ['auto', 'ball\_tree', 'kd\_tree', 'brute']  # Algorithm to compute the nearest neighbors

}

knn\_classifier = KNeighborsClassifier()

grid\_search = GridSearchCV(estimator=knn\_classifier,

                           param\_grid=param\_grid,

                           scoring='accuracy',

                           cv=5,

                           verbose=4,

                           n\_jobs=-1)

grid\_search.fit(X\_train, y\_train)

best\_params = grid\_search.best\_params\_

best\_score = grid\_search.best\_score\_

print("Best Parameters:", best\_params)

print("Best Score:", best\_score)

best\_knn\_classifier = grid\_search.best\_estimator\_

# Predict on test set

y\_pred = best\_knn\_classifier.predict(X\_test)

from sklearn.metrics import confusion\_matrix, classification\_report, accuracy\_score

cm = confusion\_matrix(y\_test, y\_pred)

# Plot the confusion matrix using Seaborn

plt.figure(figsize=(10, 7))

sns.heatmap(cm, annot=True, fmt='d', cmap='Blues',

            xticklabels=symptom\_df['label'].unique(),

            yticklabels=symptom\_df['label'].unique())

plt.xlabel('Predicted Label')

plt.ylabel('True Label')

plt.title('Confusion Matrix')

plt.show()

# Generate classification report

report = classification\_report(y\_test, y\_pred, target\_names=symptom\_df['label'].unique())

# Print classification report

print("Classification Report:")

print(report)

# Calculate and print accuracy

accuracy = accuracy\_score(y\_test, y\_pred)

print(f"Accuracy: {accuracy:.5f}")

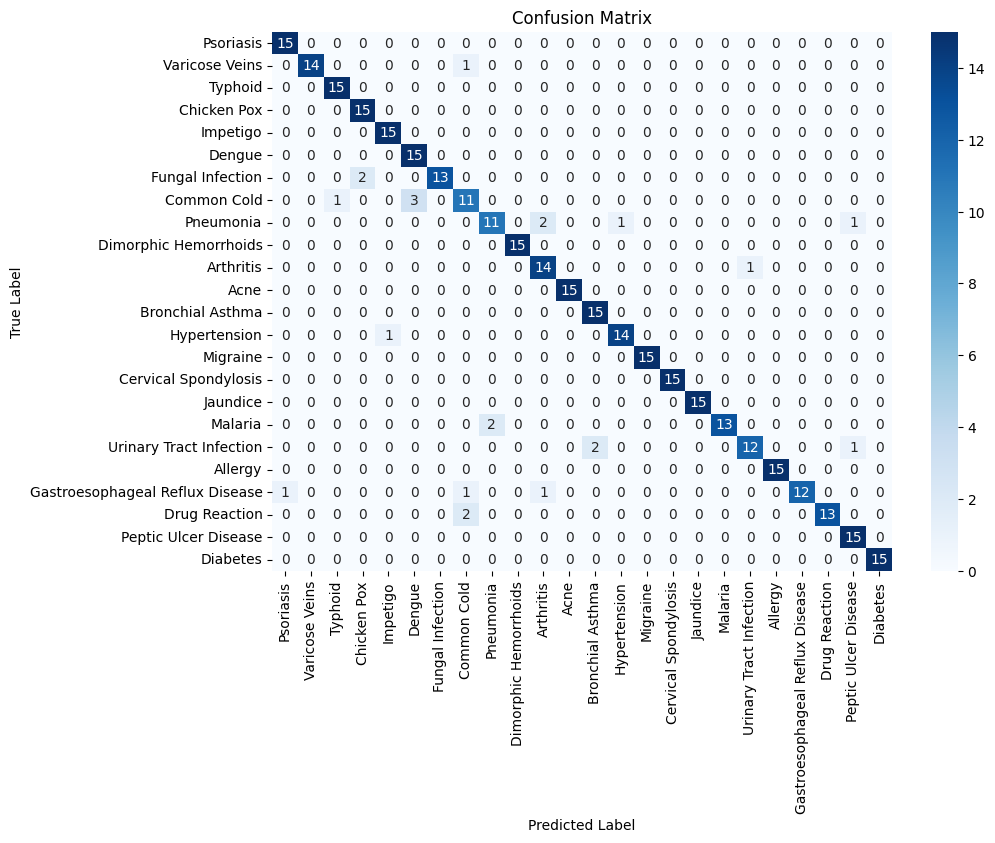
symptom = "I have a lot of trouble breathing, my chest feels heavy and i cough mucus"

preprocessed\_symptom = preprocessing(symptom)

symptom\_tfidf = tfidf\_vectorizer.transform([preprocessed\_symptom])

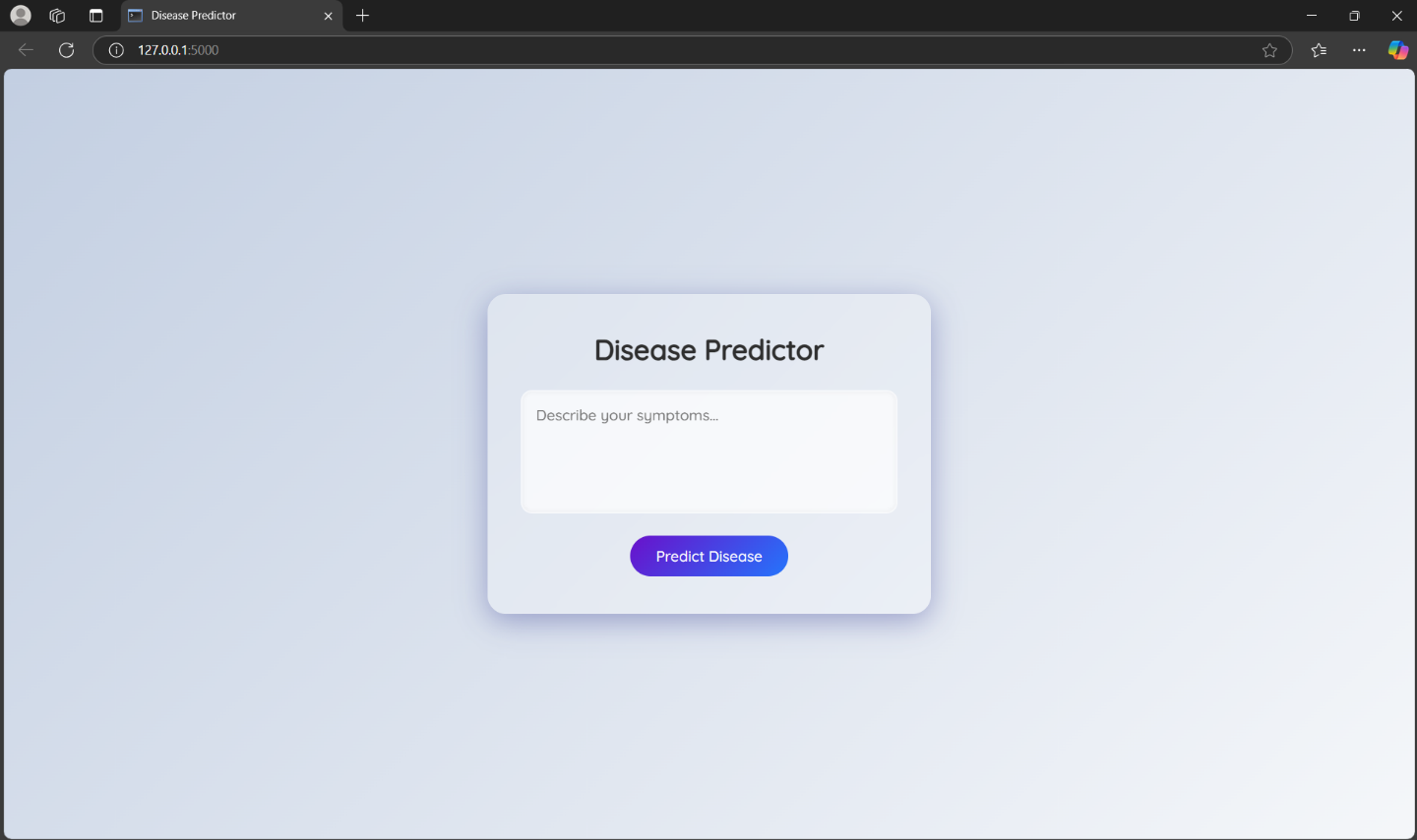
predicted\_disease = best\_knn\_classifier.predict(symptom\_tfidf)

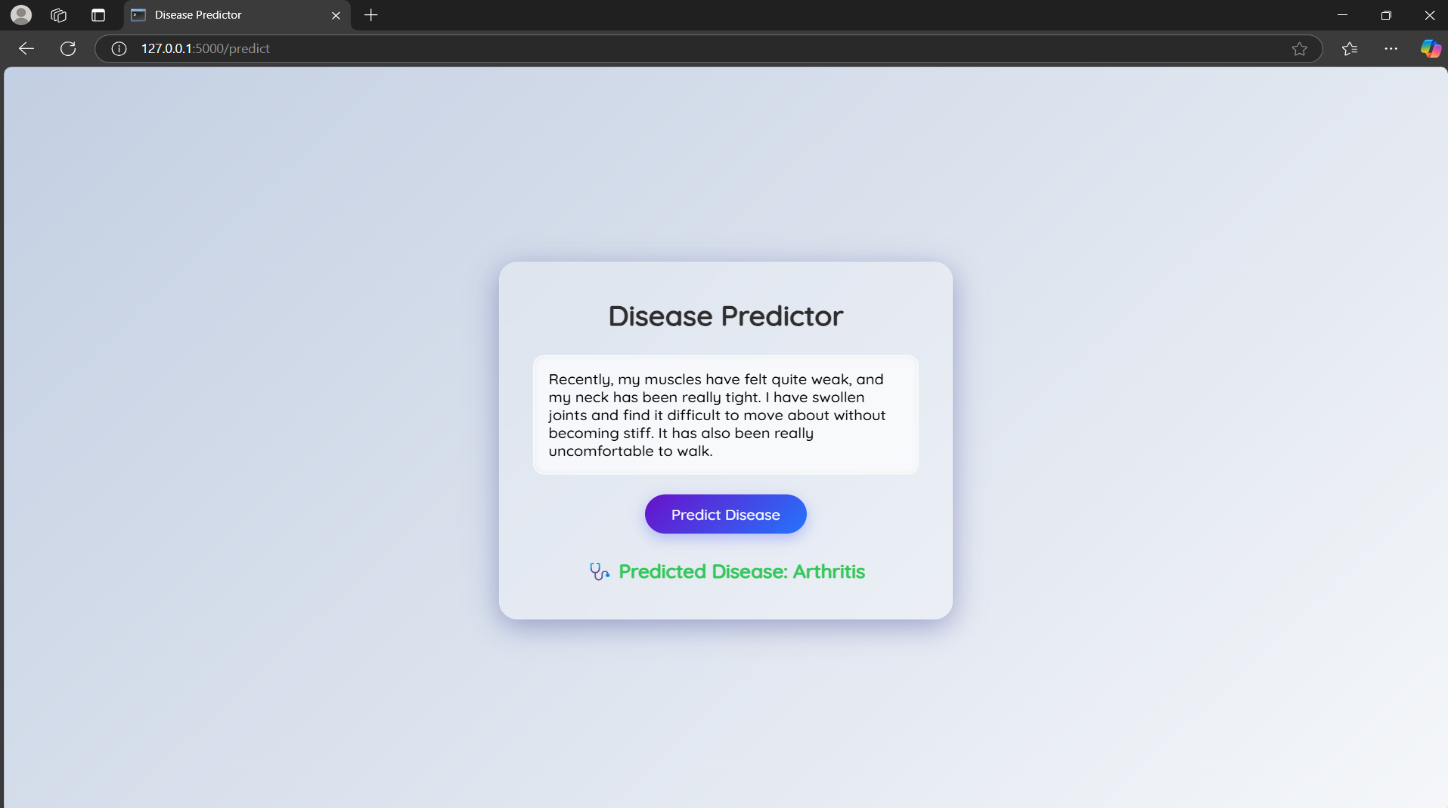
print(f'Predicted Disease: {predicted\_disease[0]}')



**APPENDIX 2**

| **Approach** | **Accuracy (%)** | **Precision (%)** | **Recall (%)** | **F1-score (%)** | **Support (%)** |
| --- | --- | --- | --- | --- | --- |
| TD-IDF + KNN | 93.61 | 93.67 | 94.05 | 94.3 | 93.6 |





This project presents a machine learning-based system designed to predict diseases from natural language symptom descriptions using TF-IDF vectorization and K-Nearest Neighbors (KNN) classification. The model is trained on a dataset of 1200 records, each containing free-text symptom descriptions and corresponding disease labels across 24 distinct medical conditions.

To ensure clean and consistent input for vectorization, the text undergoes a thorough preprocessing pipeline, including stopword removal, lemmatization, punctuation removal, and lowercasing. The TF-IDF technique is used to convert the preprocessed text into numerical feature vectors that highlight term relevance. The KNN algorithm then classifies each symptom input based on the majority label among its nearest neighbors.

The model achieved a high accuracy of 93.611%, with performance validated using precision, recall, and F1-score metrics. Techniques such as Grid Search, cross-validation, and oversampling were used to fine-tune the model and handle class imbalance.

This project demonstrates a lightweight, interpretable, and efficient solution for symptom-based disease classification, offering potential applications in clinical triage, telemedicine, and low-resource healthcare environments. Future enhancements include multi-label classification, word embeddings for deeper semantic understanding, and the use of larger real-world datasets to improve generalizability.

**REFERENCES**

1. Ramos, J. (2003). *Using TF-IDF to Determine Word Relevance in Document Queries*. Proceedings of the First Instructional Conference on Machine Learning.
2. Cover, T., & Hart, P. (1967). *Nearest neighbor pattern classification*. IEEE Transactions on Information Theory, 13(1), 21–27.
3. Gupta, D., & Khanna, A. (2017). *A Study of Applications of Machine Learning in Healthcare*. International Journal of Computer Applications, 162(8), 1–5.
4. Liu, L., Tang, L., Dong, W., Yao, S., & Zhou, W. (2016). *An overview of topic modeling and its current applications in bioinformatics*. SpringerPlus, 5(1), 1608.
5. Tiwari, R., & Pant, M. (2019). *Disease Prediction System Based on Symptoms Using Machine Learning*. In Proceedings of the 2019 International Conference on Machine Learning, Big Data, Cloud and Parallel Computing (pp. 229–234). IEEE.
6. Mikolov, T., Sutskever, I., Chen, K., Corrado, G., & Dean, J. (2013). *Distributed Representations of Words and Phrases and their Compositionality*. Advances in Neural Information Processing Systems (NeurIPS).
7. Kaggle Dataset: Niyar Barman. (2022). *Symptom to Disease Dataset*. <https://www.kaggle.com/datasets/niyarrbarman/symptom2disease>