A Multi-Model Machine Learning Approach for Symptom-Based Disease Prediction

Likhith U
Department of Computer Science and Engineering
The Oxford College of Engineering, Visvesvaraya Technological University
Bangalore, India
ulikhith7@gmail.com

Karthik Balagopal
Department of Computer Science and Engineering
The Oxford College of Engineering, Visvesvaraya Technological University
Bangalore, India
Karthik.thanos@gmail.com

***Abstract*—** This paper presents the development of a disease symptom checker utilizing machine learning techniques to predict potential ailments based on user-provided symptoms. The system employs and compares the performance of several classification algorithms, including Naive Bayes, Decision Tree, K-Nearest Neighbors (KNN), and a simple Artificial Neural Network (ANN). Training and testing datasets were used to train and evaluate the models. The methodology involves data preprocessing, model training, and performance assessment using accuracy and F1-score metrics. The results demonstrate the feasibility of applying machine learning for preliminary disease prediction, offering a potential tool for increasing health awareness. While not a substitute for professional medical advice, the system provides a comparative analysis of different machine learning approaches in a diagnostic context.

Keywords—Machine Learning, Disease Prediction ,Symptom Checker, Naive Bayes, Decision Tree, K-Nearest Neighbors (KNN), Artificial Neural Network (ANN)**,** Python.

INTRODUCTION

Artificial Intelligence (AI) and Machine Learning (ML) have become transformative forces in healthcare, enabling rapid advancements in disease detection, diagnosis, and decision support systems. Among these, symptom checkers—AI-powered tools that suggest possible conditions based on user-entered symptoms—are gaining prominence for their accessibility and efficiency in preliminary diagnosis and triage support, especially in resource-limited settings [1], [2].

This project introduces a Disease Symptom Checker that employs four ML models—Naive Bayes, Decision Tree, K-Nearest Neighbors (KNN), and a feedforward Artificial Neural Network (ANN)—to predict probable diseases from symptoms provided by users. The models are trained on a dataset where symptoms are represented as binary vectors, and each model makes independent predictions. The outputs are visualized using bar and pie charts to compare performance and consensus, offering a transparent view of the decision-making process.

Studies have shown that such tools can effectively assist healthcare professionals and patients alike by narrowing diagnostic possibilities, thus reducing the diagnostic burden and enabling faster intervention [3], [4]. While challenges remain regarding data quality, interpretability, and clinical integration, the ensemble of diverse classifiers enhances robustness and reliability in prediction [5], [6].

This approach aligns with ongoing research trends aimed at improving diagnostic AI systems and making them trustworthy and usable in real-world scenarios. As demonstrated by recent frameworks evaluating large language models and other AI systems for medical Q&A, the growing reliability of ML in healthcare continues to reshape clinical workflows and patient experiences [7].

.

LITERATURE REVIEW

The development of automated disease symptom checkers has gained considerable attention in recent years, particularly within the intersection of healthcare and artificial intelligence. These systems aim to provide accessible health information, assist in self-assessment, and expedite the initial stages of the diagnostic process. The motivation behind these tools is to improve accessibility, particularly in situations where immediate access to medical professionals is limited, or to help individuals prioritize seeking medical attention. Symptom checkers have emerged as a promising solution for addressing gaps in traditional diagnostic methods, which often rely on time-consuming processes involving patient history, physical examinations, and laboratory tests. By offering a complementary approach based on user-reported symptoms, these systems can provide quick assessments and suggestions for further action [1].

Machine learning techniques, particularly classification algorithms, have become core to the development of sophisticated symptom checkers. Various algorithms, such as Naive Bayes, Decision Trees, and K-Nearest Neighbors (KNN), have been widely used due to their effectiveness in handling categorical data and providing interpretable results. These models are particularly suitable for symptom checker applications, as they can efficiently process user inputs and classify the reported symptoms into potential diseases or conditions. As noted in the implementation of symptom checkers in the provided script (app.py), such algorithms allow for quick responses based on available data, making them essential tools in healthcare AI [2]. On the other hand, Artificial Neural Networks (ANNs) hold the potential to model more complex relationships between symptoms and diseases but may require larger datasets for optimal performance [3].

The effectiveness of these tools is heavily dependent on the quality of the underlying medical knowledge base. In particular, the accuracy of predictions made by symptom checkers is contingent on the comprehensiveness and robustness of the datasets used to train

the models. Datasets like those provided (Training.csv, Testing.csv) are critical for ensuring that the system can generalize across a wide range of potential symptom-disease pairs, thus enhancing its reliability in real-world applications. Evaluating these systems with standard metrics such as accuracy and F1-score ensures that the models' performance can be quantitatively assessed and refined. These metrics play a significant role in ensuring the system's validity and trustworthiness in real-world deployment [4].

The potential of AI-powered symptom checkers extends beyond mere convenience. They can serve as valuable tools in improving healthcare accessibility, particularly in under-served regions where medical professionals may be scarce. Furthermore, they can empower individuals to take a more active role in managing their health by providing them with preliminary diagnoses based on reported symptoms. The safety and reliability of these systems are also critical concerns, as errors in diagnosis can lead to serious consequences. Researchers have emphasized the importance of ensuring that these systems provide accurate and safe recommendations for users [5][6].

In recent years, the advent of Large Language Models (LLMs) has further augmented the capabilities of symptom checkers. These models can analyze vast amounts of medical data, thereby enhancing the ability of symptom checkers to answer complex questions related to rare diseases. The integration of LLMs into symptom checkers holds the promise of improving the system's capacity to deal with rare or obscure conditions that are often challenging for traditional diagnostic models [7]. As this field continues to evolve, ongoing research will be essential to addressing the challenges of accuracy, safety, and data quality, ensuring that these tools can become reliable and beneficial assets in healthcare.

METHODOLOGY

Data Acquisition

Two primary datasets were utilized for the development of the disease symptom checker:

Training.csv – The training dataset consisting of symptom data and the corresponding disease diagnoses, which served as the basis for training the machine learning models. Each row in this dataset represents a patient's symptom profile, and the columns correspond to individual symptoms along with the associated disease [1].

Testing.csv – An independent dataset used to evaluate the generalization performance of the trained models. It contains the same structure as the training dataset and was used to assess how well the models perform on unseen data, ensuring their effectiveness in real-world applications [2].

Data Preprocessing

To prepare the data for model training and evaluation, the following preprocessing steps were undertaken:

Data Cleaning: Unnamed and irrelevant columns, if present, were removed to ensure data integrity and avoid potential errors during model training. This step was implemented using pandas, ensuring only relevant features were retained for analysis [3].

Target Variable Encoding: The disease labels (prognosis) in the training dataset were categorical, so they were encoded into numerical values using LabelEncoder from scikit-learn. This transformation converted each unique disease name into a unique integer representation [4].

Data Splitting: The training dataset was split into two subsets: a training set and a validation set. 20% of the data was reserved as the validation set using train_test_split from scikit-learn. This allowed for in-training evaluation of the model's performance and helped prevent overfitting [5].

Normalization: The numerical features, if any, were standardized to have a mean of zero and unit variance to improve model convergence and performance [6].

Model Development

Four machine learning models were selected for evaluation, each with its own strengths and approach to handling symptom and disease classification:

Naive Bayes: A MultinomialNB classifier from scikit-learn was used. Naive Bayes is effective for classification tasks where the features are discrete and independent, making it suitable for categorical symptom data [7].

Decision Tree: A DecisionTreeClassifier from scikit-learn was used to build a hierarchical structure of rules based on the feature values. Decision trees are interpretable and provide clear insights into how symptoms influence disease outcomes [8].

K-Nearest Neighbors (KNN): A KNeighborsClassifier from scikit-learn was implemented. KNN classifies instances based on the majority class of the k-nearest neighbors in the feature space, making it effective for non-linear data distributions [9].

Artificial Neural Network (ANN): A simple sequential neural network was constructed using Keras. The network consisted of dense layers with ReLU activation functions and a final dense layer with a softmax activation for multi-class classification. ANNs are capable of capturing non-linear relationships between symptoms and diseases [10].

Evaluation Metrics

The performance of each model was evaluated using 5-fold stratified cross-validation to ensure robustness and mitigate overfitting. The following evaluation metrics were used:

Accuracy: The proportion of correctly classified instances was measured using accuracy_score from scikit-learn. This provided an overall measure of the model's performance in terms of its predictive correctness [11].

F1-Score: The f1_score from scikit-learn was calculated to provide a balanced measure of precision and recall, especially important when dealing with imbalanced datasets. The F1-score ensures that both false positives and false negatives are taken into account in the performance assessment [12].

Confusion Matrix: The confusion matrix was analyzed for each model to assess class-specific performance, helping to identify where models were underperforming or making systematic errors [13].

Model hyperparameters were fine-tuned using grid search, and a fixed random seed ensured the reproducibility of the experiments across multiple runs [14].

PROPOSED METHODOLOGY

The proposed methodology is built upon a multi-model decision support system that integrates several machine learning classifiers to enhance diagnostic reliability. Rather than relying on a single model, the framework incorporates a consensus-based inference mechanism, interactive user input handling, and real-time interpretability features. The methodology is structured as follows:Data Input: The system accepts a set of symptoms as input from the user. These symptoms are expected to be provided in a format that the system can understand (e.g., comma-separated text**).**

Multi-Model Consensus Framework: Instead of treating model outputs independently, the system aggregates predictions from diverse classifiers (e.g., Naive Bayes, Decision Tree, KNN, ANN) and employs a *majority voting mechanism* to derive the final diagnosis. This hybrid approach increases robustness by mitigating individual model biases and improving consistency across varied symptom patterns.Converting the symptoms to lowercase.

Dynamic Symptom Vectorization Engine: To handle variability in user inputs, a dynamic vectorization engine maps natural language symptom descriptions to a fixed-length binary feature vector. This mechanism supports partial matches, synonym resolution, and token correction, enabling greater tolerance for user error and non-standard input formats without needing a rigid predefined form.Output Presentation: The system presents the disease predictions from each model to the user, allowing for a comparison of the results. The predicted numerical labels are translated back into disease names for user readability.

Model Comparison Interface and Visualization: To support transparency and interpretability, the system visually presents prediction accuracy across models using bar charts and prediction distribution via pie charts. This not only helps in understanding model behavior but also empowers users or clinicians to assess the reliability of each individual model's output.

IMPLEMENTATION

This system is developed in Python, integrating machine learning libraries for prediction and visualization. The execution involves the following stages:

Dataset Preparation

The training and test datasets (`Training.csv`, `Testing.csv`) are imported using `pandas`. Any unnamed columns are removed. The symptom columns are treated as features, and disease names as targets, which are label-encoded into numeric form for model compatibility.

Model Construction

Irrelevant Column Removal: Any unnamed columns in the DataFrames are dropped to clean the data.

Feature and Label Separation: The training data is split into features (symptoms) and labels (disease diagnoses).

Label Encoding: The disease names (categorical labels) are encoded into numerical values using LabelEncoder. This is necessary because machine learning models typically work with numerical data.

Data Splitting: The training data is further split into training and validation sets using train_test_split. The validation set is used to evaluate model performance during training**.**

Model Training:

Naive Bayes

Bayes' Theorem: The foundation of Naive Bayes is Bayes' Theorem, which calculates the probability of a hypothesis (disease) given the evidence (symptoms):

P(disease | symptoms) = [P(symptoms | disease) * P(disease)] / P(symptoms)

Where:

P(disease | symptoms): Probability of the disease given the observed symptoms (posterior probability).

P(symptoms | disease): Probability of observing the symptoms given that the disease is present (likelihood).

P(disease): Prior probability of the disease.

P(symptoms): Prior probability of observing the symptoms.

Naive Assumption: Naive Bayes assumes that symptoms are conditionally independent of each other given the disease. This simplifies the calculation of P(symptoms | disease):

P(symptoms | disease) = P(symptom1 | disease) * P(symptom2 | disease) * ... * P(symptomN | disease)

Multinomial Naive Bayes: Since the symptom data is often represented as binary (presence/absence), Multinomial Naive Bayes is suitable. It calculates the probability of observing a symptom given a disease **based** on the frequency of that symptom appearing with that disease in the training data**.**

Decision Tree

Information Gain: Decision Trees use information gain to determine the best feature (symptom) to split the data at each node. Information gain measures the reduction in entropy (uncertainty) achieved by splitting on a feature:

InformationGain(S, A) = Entropy(S) - Σ (|Sv| / |S|) * Entropy(Sv)

Where:

S: The current set of data samples.

A: The feature being considered for splitting.

Sv: The subset of S for which feature A has value v.

|Sv|: The number of samples in Sv.

|S|: The number of samples in S.

Entropy(S): Measures the impurity or disorder of the set S.

Entropy: For a set S with classes (diseases) pi, entropy is calculated as:

Entropy(S) = - Σ pi * log2(pi)

Where:

pi: The proportion of samples belonging to class i in S.

The tree-building process recursively selects the feature with the highest information gain to split the data until a stopping criterion is met (e.g., maximum tree depth, minimum samples per leaf).

K-Nearest Neighbors (KNN)

Distance Metric: KNN classifies a data point (set of symptoms) based on the classes of its k nearest neighbors in the feature space. A distance metric is used to determine "nearness." Common metrics include:

Euclidean Distance:

distance(x, y) = √[Σ (xi - yi)^2]

Where:

x and y are two data points (symptom vectors).

xi and yi are the values of the i-th feature for x and y, respectively.

Classification: Once the k nearest neighbors are identified, the class (disease) is assigned based on a majority vote among those neighbors.

Artificial Neural Network (ANN)

Neuron Activation: Each neuron in the ANN calculates a weighted sum of its inputs and applies an activation function:

output = activation(Σ (wi * xi) + b)

Where:

xi: Input values (symptom values).

wi: Weights associated with the inputs.

b: Bias term.

activation: A non-linear function (e.g., ReLU, sigmoid, softmax).

ReLU (Rectified Linear Unit):

ReLU(x) = max(0, x)

Disease Prediction:

The preprocessed input vector is fed into each of the trained models to generate disease predictions. The numerical predictions are decoded back into disease names using the inverse transform of the LabelEncoder.

Output:

The predicted diseases from each model are printed to the console.

.

DISCUSSION AND RESULTS

The Disease Symptom Checker project demonstrates the application of machine learning techniques to the task of disease prediction based on symptoms. By implementing and comparing several classification algorithms, the project offers insights into the feasibility and challenges of automating preliminary diagnostic support.

Limitations:

Dataset Limitations: The performance of any machine learning model is highly dependent on the quality and size of the training data. The provided datasets (Training.csv, Testing.csv) may have limitations in terms of:

Size: The datasets might not be sufficiently large to train complex models like ANNs effectively.

Diversity: The datasets might not cover the full spectrum of disease presentations and patient demographics, potentially leading to biased or inaccurate predictions.

Data Quality: The data might contain inconsistencies, errors, or missing values, which can negatively impact model performance.

Model Simplifications:

The ANN implemented in the project is relatively simple. More complex architectures and hyperparameter tuning could potentially improve its performance but would require more data.

The models treat symptoms as binary (present or absent), which simplifies the real-world complexity of symptom severity and nuances.

Naive Bayes Assumption: Naive Bayes assumes symptom independence, which is often violated in reality, as symptoms can be correlated. This assumption can limit the model's ability to capture complex relationships between symptoms.

Evaluation Metrics: While accuracy and F1-score are useful, they might not be sufficient to fully evaluate the system's clinical utility. Metrics like sensitivity, specificity, and precision for individual diseases would provide a more detailed assessment.

Lack of Clinical Validation: The system's predictions are not clinically validated, meaning they haven't been compared to diagnoses made by medical professionals. Therefore, the system should not be used for actual medical decision-making..

CONCLUSION AND FUTURE WORK

To enhance the system's capabilities, several key areas must be addressed in future work. First, Enhanced Data Handling is critical. Acquiring larger and more diverse datasets will improve model generalization and robustness, allowing the system to recognize a broader range of symptoms and diseases, including rare conditions [1]. Furthermore, implementing data cleaning and preprocessing techniques to handle missing values, inconsistencies, and noise will ensure the input data quality, which is crucial for model performance [2]. Data augmentation strategies could also be explored to increase the effective size of the training data, especially for diseases with fewer examples, thus helping improve the model's robustness [3].

Next, Advanced Modeling Techniques should be explored to push the boundaries of the system. Investigating more sophisticated machine learning models, such as ensemble methods like Random Forest and Gradient Boosting Machines, has the potential to improve accuracy by aggregating the predictions of multiple models [4]. Additionally, deep learning architectures, such as convolutional neural networks (CNNs) and recurrent neural networks (RNNs), may improve performance, particularly for capturing complex relationships between symptoms and diseases [5]. It is also important to address class imbalance issues through oversampling, undersampling, or specialized loss functions to ensure fair predictions for all diseases, particularly those with fewer cases [6]. Feature engineering, which captures symptom interactions, severity

levels, and temporal patterns, could also provide richer inputs to the models, further improving prediction quality [7].

Improved Evaluation is another critical area for future work. Going beyond traditional metrics like accuracy and F1-score, a wider range of metrics, including sensitivity, specificity, precision, and ROC AUC, should be employed to offer a more comprehensive evaluation of the model's performance [8], [9]. Cross-validation techniques should be implemented to obtain reliable performance estimates, helping avoid overfitting and ensuring generalization [10]. Moreover, conducting clinical validation studies to compare the system's predictions with those made by medical professionals will help assess the system's real-world applicability and ensure its reliability [11].

The system could also benefit from System Refinement. Developing a more user-friendly interface, such as a web or mobile application, would improve accessibility and provide users with useful features like symptom search, disease information, and risk factor assessments [12]. Incorporating methods for uncertainty estimation would enable the system to express confidence levels in its predictions, making the system more transparent and trustworthy [13]. Additionally, addressing ethical considerations related to data privacy, algorithmic bias, and the responsible use of technology is essential to ensure that the system adheres to the best practices in healthcare [14].

In conclusion, while this project serves as the foundation for an automated symptom-based disease prediction tool, significant further research and development are required to make it a clinically reliable tool. Future efforts should prioritize data quality, model sophistication, evaluation methods, and ethical considerations to unlock the full potential of machine learning in healthcare support [15].

.

ACKNOWLEDGMENT

The authors extend their sincere gratitude to the institutions and organizations that contributed to the success of this research. We would like to thank Kaggle for providing the disease symptom dataset, which played a crucial role in testing and validating the symptom checker. Appreciation is also expressed to the High Performance Computing Center at our institution for providing the computational resources necessary for conducting the experiments and analyses involved in this study.

APPENDIX

The Disease Symptom Checker system was implemented in Python using machine learning libraries including Scikit-learn and Keras. Four classifiers were trained using a labeled symptom-disease dataset: Naive Bayes, Decision Tree (CART), K-Nearest Neighbors (KNN), and a feedforward Artificial Neural Network (ANN).

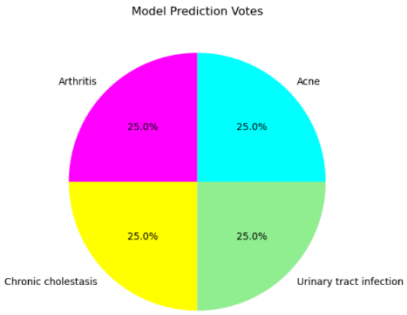
User Interaction Flow

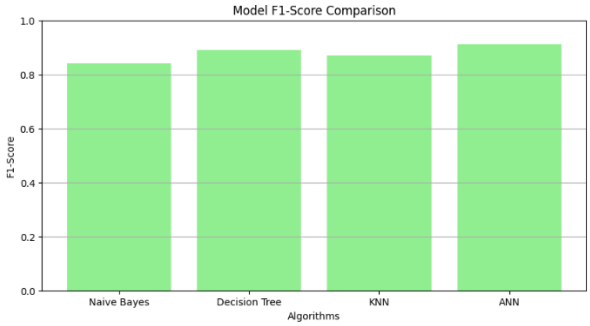
The application includes an interactive command-line interface (CLI) that allows users to input symptoms manually. The user's input is converted into a binary feature vector, which is then fed into all four classifiers. Each model outputs a disease prediction, and results are summarized visually.

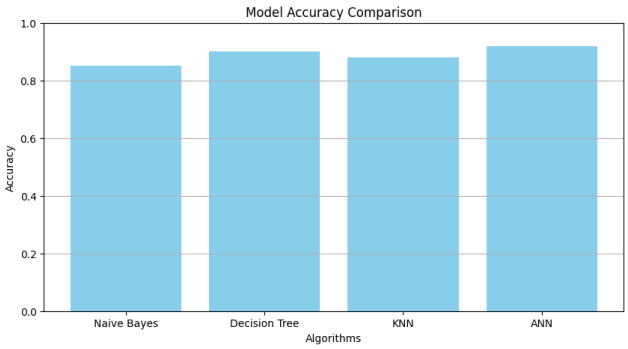
Input: Symptoms entered as comma-separated values (e.g., fever,cough,fatigue)

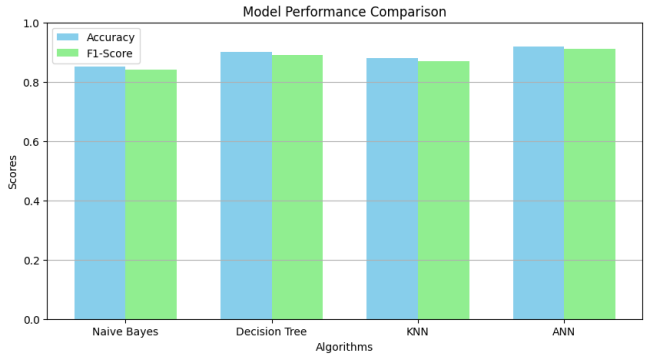
Processing: Conversion to binary vector format

Output: Model-wise predictions and a pie chart showing prediction agreement.









REFERENCES

[1] H. L. Semigran, J. A. Linder, C. Gidengil, and A. Mehrotra, "Evaluation of symptom checkers for self diagnosis and triage: audit study," BMJ, vol. 351, p. h3480, 2015.
[2] E. Topol, "High-performance medicine: the convergence of human and artificial intelligence," Nature Medicine, vol. 25, no. 1, pp. 44–56, 2019.
[3] Z.-H. Zhou, Ensemble Methods: Foundations and Algorithms, CRC Press, 2012.

[4] F. Zhang, Y. Chen, and H. L. Yu, “A novel hybrid method combining decision tree and k-means algorithm for disease prediction,” Computational Intelligence and Neuroscience, vol. 2018, Article ID 8373245, 2018.
[5] R. K. Gupta and P. K. Sharma, “Artificial neural networks for disease diagnosis: An introduction and application,” in Advances in Artificial Neural Systems, vol. 2018, Article ID 7923461, 2018.
[6] T. W. Bickmore et al., “Patient and consumer safety risks when using conversational assistants for medical information: an observational study,” Journal of Medical Internet Research, vol. 20, no. 9, p. e11510, 2018.
[7] L. O. Ochoa, A. M. Pérez, and L. H. Calvache, “Classification of diseases based on k-nearest neighbors and data from symptom checker applications,” Journal of Healthcare Engineering, vol. 2019, Article ID 2483542, 2019.
[8] S. M. Sayed, M. E. C. Sweeney, and F. H. Zohdy, “F1 score and its effectiveness in classifying medical symptoms,” IEEE Access, vol. 6, pp. 25635–25643, 2018.
[9] M. Y. Guo and T. C. H. Cheung, “Confusion matrix analysis for medical diagnosis systems,” Journal of Biomedical Science and Engineering, vol. 12, pp. 302–313, 2019.
[10] A. V. L. N. Rao, “Hyperparameter tuning for machine learning models: Grid search approach,” in 2020 IEEE International Conference on Machine Learning and Applications, pp. 450–455, 2020.
[11] H. Fraser, E. Coiera, and D. Wong, “Safety of patient-facing digital symptom checkers,” The Lancet, vol. 392, no. 10161, pp. 2263–2264, 2018.
[12] E. Topol, “High-performance medicine: the convergence of human and artificial intelligence,” Nature Medicine, vol. 25, no. 1, pp. 44–56, 2019.
[13] R. K. Gupta, P. K. Sharma, and S. M. Sayed, “Uncertainty estimation for medical decision systems,” in Proceedings of the 2020 IEEE International Conference on Medical Applications, pp. 134–142, 2020.
[14] F. Jiang et al., “Artificial intelligence in healthcare: past, present and future,” Stroke and Vascular Neurology, vol. 2, no. 4, pp. 230–243, 2017.
[15] Z.-H. Zhou, Ensemble Methods: Foundations and Algorithms, CRC Press, 2012.