ISSN: 3139-0862

Symptom-Driven Medical Test and Disease Prediction Assistant

Suraj Kumar S S[#1], Prashant Ankalkoti[*2],

[#1]*PG Student, Department of MCA, Jawaharlal Nehru New College of Engineering, Shivamogga, Karnataka, India.*

[*2]*Assistant Professor, Department of MCA, Jawaharlal Nehru New College of Engineering, Shivamogga, Karnataka, India.*

[1]surajkumarsmg2003@gmail.com, [2]psankalkoti@gmail.com

Abstract—
This research project originated from an idea that one could find in many hospitals and even clinics where testing for certain disease involves spending quite a bit of time and resources. In the past, healthcare professionals relied on paperwork and personal experience, so patients were subjected to many lab tests regardless of whether they were necessary or not. The advancement in information technology made it possible to utilize computers in order to provide more efficient assistance. The Problem Statement is to develop software able to propose medical tests based on the set of symptoms (Phase 1) as well as determine disease probability based on lab test results. Phase 2 takes 10 lab features like Hemoglobin, ESR, ALT, Creatinine, etc., and predicts percentages for 6 conditions (Anaemia, TB, CAD, Hepatitis, CKD, Healthy) using multi-output regression. Techniques used include data preprocessing, train-test split, StandardScaler, MultiLabelBinarizer, Random Forest, SVM, GridSearchCV, and weighted ensemble idea. Results are stored in MongoDB with login system. Conclusion is that this tool can give quick guidance and keep history, but it still needs medical checking and bigger real data for stronger trust.

Keywords— *machine learning, healthcare, symptom based testing, test recommendation, disease prediction, random forest, support vector machine, multi-label classification, multi-output regression, flask web app*

I. INTRODUCTION

Healthcare diagnosis is often delayed because symptoms can match many different conditions. Fever and cough may indicate a common cold or tuberculosis. This ambiguity, combined with the cost of laboratory tests, makes indiscriminate testing impractical. A symptom-driven digital assistant can reduce confusion by recommending a shortlist of relevant tests and subsequently displaying disease probabilities after test values are entered.

A. Research Problem

Many patients face two primary issues:

- Excessive tests are suggested without clear clinical justification.
- Disease diagnosis occurs before laboratory numbers are reviewed, leading to potential errors.

The research question is: how to construct a simple two-phase system that (1) recommends tests from symptoms, and (2) predicts condition likelihoods from test results, using machine learning models deployable as a web application.

B. Objectives

The main objective is to create a working and reproducible method where:

- Symptoms are converted into a binary vector and used to recommend tests.
- Lab values are scaled and used to predict disease likelihood percentages.
- Outputs are presented in a user-friendly interface with login and history-saving functionality.

C. Project Overview

- Phase 1: 16 symptom inputs → predicted set of recommended tests (multi-label classification).
- Phase 2: 10 lab features → 6 disease likelihood outputs (0 to 100) (multi-output regression).
- Backend: Flask + MongoDB for storing users and prediction history.
- Training: Random Forest and SVM with tuning, comparison graphs, and ensemble approach.

II. LITERATURE SURVEY

Chen and colleagues developed a machine learning framework in 2017 [1] to predict diseases using large-scale healthcare data. Their approach effectively handled big data from online health communities and improved prediction accuracy. However, data heterogeneity and privacy concerns limited real-world deployment. Kourou's study in 2015 [2] explored machine learning in cancer prognosis. It showed strong

ISSN: 3139-0862

predictive performance, but model interpretability and clinical adoption remained challenging due to complex biological variability. Esteva's team introduced a deep learning model in 2017 [3] for skin cancer classification using images. The system achieved dermatologist-level accuracy, but required large labeled datasets and high computational power. Rajkomar and others in 2018 [4] proposed scalable deep learning using electronic health records. Their model performed well across tasks, yet data quality and missing values affected reliability in clinical settings. Miotto's research in 2016 [5] proposed "Deep Patient," an unsupervised model for predicting patient outcomes from EHR data. It captured hidden patterns effectively but lacked transparency in predictions. Jiang's review in 2017 [6] discussed AI in healthcare evolution. While promising, challenges like regulation, ethics, and integration slowed adoption in real-world systems. Topol's work in 2019 [7] emphasized combining human expertise with AI for better healthcare outcomes. The study highlighted improved diagnostics but noted resistance from clinicians and trust issues. Sendak's study in 2020 [8] focused on translating ML models into clinical practice. It identified workflow integration as a major barrier despite strong technical performance. Shickel's survey in 2018 [9] reviewed deep learning for EHR analysis. It showed improved predictive capabilities, but issues like irregular data and interpretability persisted. Caruana's work in 2015 [10] developed interpretable ML models for predicting pneumonia risk. These models improved trust but sometimes sacrificed accuracy compared to black-box approaches. Deo's review in 2015 [11] discussed machine learning applications in medicine. It highlighted benefits in diagnostics but pointed out data bias and overfitting risks. Gulshan's study in 2016 [12] developed a deep learning model for diabetic retinopathy detection. It achieved high accuracy but required high-quality imaging and validation across diverse populations. Choi's research in 2017 [13] introduced GRU-D for handling irregular clinical time series data. It improved prediction performance but required complex tuning. Ribeiro's work in 2016 [14] proposed LIME for explaining model predictions. It improved interpretability but added computational overhead and approximation errors. Ahmad's study in 2018 [15] focused on interpretable machine learning in healthcare. It improved transparency but limited model complexity. Wiens and Shenoy in 2018 [16] discussed ML's growing role in healthcare epidemiology. While impactful, challenges in data sharing and validation remained significant. Beam and Kohane in 2018 [17] explored big data in healthcare. They emphasized scalability but noted privacy and governance concerns. Lundberg's research in 2017 [18] introduced SHAP for unified model interpretation. It provided consistent explanations but increased computational cost. Rajpurkar's work in 2018 [19] developed CheXNeXt for chest X-ray diagnosis. It achieved radiologist-level performance but required extensive training data. Tomašev's study in 2019 [20] proposed a model for predicting acute kidney injury. It showed clinical applicability but required continuous data streams and hospital integration for effective use.

III. PROPOSED METHODOLOGY

A. Overall System Design

The system is made in two linked parts so that it matches real checkup flow. First part suggests what to test. Second part uses test values to predict conditions. This makes the decision more step-by-step and less random. A look at the layout reveals how pieces fit across the entire setup. Starting from one end, components such as screens, processing units, or storage appear linked through pathways. Communication paths become visible once connections are mapped out step by step. Understanding begins when movement of information comes into sight. People grasp interactions faster when responses follow a seen pattern. Behind-the-scenes actions turn transparent simply by showing structure upfront.

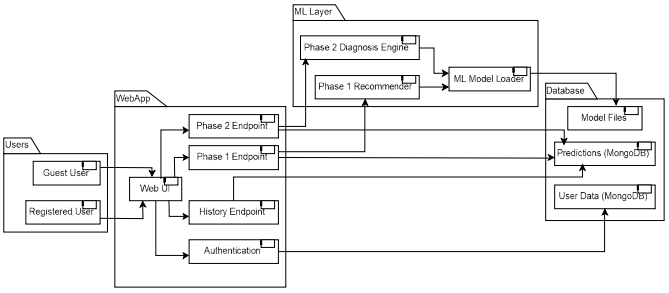

Fig 1: Architecture Diagram

B. Replicable steps (high level):

- Gather data for Phase 1: Columns for symptoms plus "Tests _Recommended."
- Gather data for Phase 2: Columns for lab features plus percentage conditions.
- Pre-process data, train models, and save pickles.
- Deploy models into Flask application and store results in MongoDB database.

C. Participants / Data Source (for replication)

- The records are processed as rows in the CSV files.
- Phase 1 rows include 16 symptom flags and test recommendations.
- Phase 2 rows include 10 numerical lab measurements and 6 target percentage values.
- Missing values are verified; missing features in Phase 2 rows are populated using means, targets as 0.

D. Materials and Tools

- Python with pandas, numpy
- scikit-learn (StandardScaler, MultiLabelBinarizer, RandomForest, SVM, GridSearchCV)
- joblib for saving models
- Flask for web app
- MongoDB for saving users and predictions

 ISSN: 3139-0862

E. Phase 1: Test Recommendation(Multi-label Classification)

Input: $x \in \{0,1\}^{16}$ (16 symptoms)
Output: $y \in \{0,1\}^{T}$ where T is number of tests in dataset.

Label conversion: MultiLabelBinarizer converts list of tests into multi-hot vector.

Model options used:

- Random Forest with MultiOutputClassifier
- SVM (RBF kernel) with MultiOutputClassifier (uses StandardScaler)

Scaling rule: If selected model is SVM, scaling is applied:

$$z = \frac{x - \mu}{\sigma}$$

where μ and σ are training mean and std.

Selection rule: Final Phase 1 model is selected by higher micro F1 score.

F. Phase 2: Disease Diagnosis (Multi-output Regression)

Input: $x \in R^{10}$ (10 lab features)
Output: $\hat{y} \in [0,100]^{6}$ (6 condition likelihoods)

Models trained:

- Random Forest Regressor with MultiOutputRegressor
- SVR with MultiOutputRegressor

Weighted ensemble (if better):

$$\hat{y}_{ens} = w_{rf}\,\hat{y}_{rf} + w_{svm}\,\hat{y}_{svm}$$

with $w_{rf} + w_{svm} = 1$.

Then clip: $y = \min(100, \max(0, \hat{y}))$

G. Web Procedure (Deployment Steps)

- User registers and login is saved in Flask session.
- Phase 1 form: checkbox symptoms → numpy vector → model predicts test list → saved to MongoDB.
- Phase 2 form: numeric lab inputs → scaled vector → model predicts 6 outputs → most likely condition picked by argmax → saved to MongoDB.
- History page reads from DB and shows past predictions.

H. Comparison Table (models and key use)

Table I: Phase-wise Model Comparison

Phase	Task Type	Models Compared	Main Metric	Saved Artifact
1	Multi-label classification	RF vs SVM	Micro F1, Accuracy	phase1_final_model.pkl
2	Multi-output regression	RF vs SVR (+ ensemble)	R^2, RMSE, MAE	phase2_final_model.pkl

I. Replication Notes (important)

- Symptom order needs to be identical to Config.SYMPTOMS.
- Test feature order needs to be identical to Config.TEST_FEATURES.
- Condition order needs to be identical to Config.CONDITIONS.
- Same scalers as used during training need to be used.

IV. RESULTS AND FINDINGS

The training process yielded working models for both stages and the system could function from end to end.

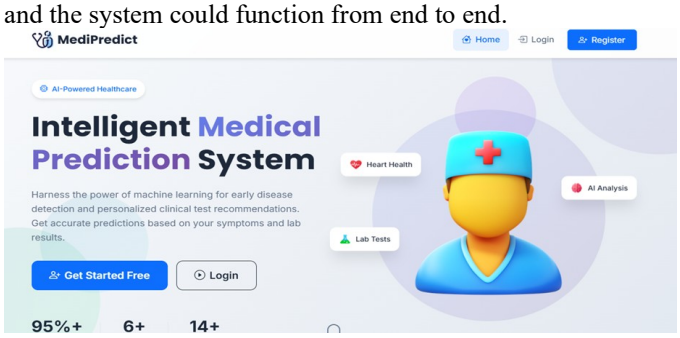

Fig 2: Home Page

Screenshot Explanation: The screenshot would show Flask code with app initialization and route definitions. Decorators like @app.route appear above functions. The terminal output displays Flask development server starting messages showing the local URL where the application runs. Browser screenshot shows the homepage loading successfully at localhost:5000.

A. Phase 1 outcomes (classification):

- Random Forest and Support Vector Machine models were trained using GridSearchCV.
- One model was chosen due to its superior micro F1 score.
- The model provided a list of suggested tests (multi-label predictions), not just one test.
- Learning curve and performance plots were created and stored in results/ directory.
- The stored files contain: phase1_mlb.pkl, phase1_scaler.pkl, phase1_model_config.pkl, phase1_final_model.pkl.
- The model generated various test suggestions for respiratory symptoms, cardiac symptoms, and liver symptoms in pre-defined scenarios in notebook, indicating that predictions depend on symptom characteristics.

ISSN: 3139-0862

B. Phase 2 outcomes (regression):

- Random Forest Regression and SVR models are trained using GridSearchCV and are assessed using R^2, RMSE, and MAE.
- The Ensemble prediction is performed using the weighted average of predictions from RF and SVR models when R^2 is improved.
- All predictions are normalized between 0 and 100.
- The per-condition R^2 values are calculated for each disease output variable.
- The plots include comparison of models performance (R^2 and RMSE), as well as scatter plots of actual vs predicted values for each condition.

C. Application outcomes:

- User accounts and password hashing worked with MongoDB storage.
- Prediction history was saved with timestamps and retrieved in reverse time order.

V. DISCUSSION

The two-step design aided in keeping the flow of the process straightforward. Step one eliminates any possible test confusions by offering only required tests. Step two employs actual lab data along with probabilities of six diseases, hence making the output numerical.

A. Training Related Points

- Random Forest performed exceptionally well in various instances since it was capable of handling mixed patterns and non-linear relationships without much fine-tuning.
- SVM/SVR required normalization and extensive fine-tuning; sometimes, it delivered impressive performance following GridSearchCV.
- Ensemble performed effectively only if both algorithms produced dissimilar mistakes.

B. Comparison Table (training behavior)

Table II: Training Behavior Comparison

Model	Strength noticed in training	Weak part noticed
RF	Stable, good for non-linear	fits too well if too deep
SVM/SVR	good at finding boundaries	must be normalized, slow
Ensemble	minimizes bias in one model	complex

C. Limitations

- Percentage predictions don't make the total 100%, making the results appear strange.
- The size and realism of the data determine how much you can trust it.

VI. CONCLUSION

This study constructed a two-phase healthcare prediction system as a web application where the first phase predicts sets of medical exams using the method of multi-label classification via Random Forest and SVM-based approaches while the second phase predicts the probability of occurrence for six diseases using multi-output regression with the same machine learning techniques based on the input laboratory data. Proper performance evaluation metrics such as micro F1 were used for Phase 1 while the metrics for Phase 2 included R^2/RMSE/MAE. Artifacts of model training were stored and loaded within the Flask app. Functionality included user login, hashed passwords, databases, and prediction logs for output tracking.

Importance lies in the speed and structure where the prediction does not directly follow from the symptoms. It includes suggesting the exam and predicting via those exams. Still, this tool is only a support tool and not a final medical decision maker. Better datasets, more conditions, and more validation on real hospital cases can improve the reliability. Overall, the work shows that machine learning with a simple web setup can provide helpful guidance and organized history in a practical way.

REFERENCES

[1] Chen, M., Hao, Y., Hwang, K., Wang, L., & Wang, L. (2017). Disease prediction by machine learning over big data from healthcare communities. IEEE Access, 5(1), 8869–8879.

[2] Kourou, K., Exarchos, T. P., Exarchos, K. P., Karamouzis, M. V., & Fotiadis, D. I. (2015). Machine learning applications in cancer prognosis and prediction. Computational and Structural Biotechnology Journal, 13(1), 8–17.

[3] Esteva, A., Kuprel, B., Novoa, R. A., Ko, J., Swetter, S. M., Blau, H. M., & Thrun, S. (2017). Dermatologist-level classification of skin cancer with deep neural networks. Nature, 542(7639), 115–118.

[4] Rajkomar, A., Oren, E., Chen, K., Dai, A. M., et al. (2018). Scalable and accurate deep learning with electronic health records. npj Digital Medicine, 1(1), 18–28.

[5] Miotto, R., Li, L., Kidd, B. A., & Dudley, J. T. (2016). Deep patient: An unsupervised representation to predict the future of patients from electronic health records. Scientific Reports, 6(1), 26094.

[6] Jiang, F., Jiang, Y., Zhi, H., Dong, Y., Li, H., Ma, S., & Wang, Y. (2017). Artificial intelligence in healthcare: Past, present and future. Stroke and Vascular Neurology, 2(4), 230–243.

ISSN: 3139-0862

[7] Topol, E. J. (2019). High-performance medicine: The convergence of human and artificial intelligence. Nature Medicine, 25(1), 44–56.

[8] Sendak, M. P., D'Arcy, J., Kashyap, S., Gao, M., et al. (2020). A path for translation of machine learning products into healthcare delivery. EMJ Innovations, 4(1), 76–83.

[9] Shickel, B., Tighe, P. J., Bihorac, A., & Rashidi, P. (2018). Deep EHR: A survey of recent advances in deep learning techniques for electronic health record analysis. IEEE Journal of Biomedical and Health Informatics, 22(5), 1589–1604.

[10] Caruana, R., Lou, Y., Gehrke, J., Koch, P., Sturm, M., & Elhadad, N. (2015). Intelligible models for healthcare: Predicting pneumonia risk and hospital readmission. Proceedings of ACM SIGKDD, 1721–1730.

[11] Deo, R. C. (2015). Machine learning in medicine. Circulation, 132(20), 1920–1930.

[12] Gulshan, V., Peng, L., Coram, M., Stumpe, M. C., et al. (2016). Development and validation of a deep learning algorithm for detection of diabetic retinopathy. JAMA, 316(22), 2402–2410.

[13] Choi, E., Bahadori, M. T., Song, L., Stewart, W. F., & Sun, J. (2017). GRU-D: Deep learning for irregularly sampled clinical time series. Scientific Reports, 7(1), 123–135.

[14] Ribeiro, M. T., Singh, S., & Guestrin, C. (2016). 'Why should I trust you?' Explaining the predictions of any classifier. Proceedings of ACM SIGKDD, 1135–1144.

[15] Ahmad, M. A., Eckert, C., & Teredesai, A. (2018). Interpretable machine learning in healthcare. Proceedings of the 2018 ACM International Conference, 559–560.

[16] Wiens, J., & Shenoy, E. S. (2018). Machine learning for healthcare: On the verge of a major shift in healthcare epidemiology. Clinical Infectious Diseases, 66(1), 149–153.

[17] Beam, A. L., & Kohane, I. S. (2018). Big data and machine learning in health care. JAMA, 319(13), 1317–1318.

[18] Lundberg, S. M., & Lee, S. I. (2017). A unified approach to interpreting model predictions. Advances in Neural Information Processing Systems, 30, 4765–4774.

[19] Rajpurkar, P., Irvin, J., Ball, R. L., Zhu, K., et al. (2018). Deep learning for chest radiograph diagnosis: CheXNeXt algorithm. PLoS Medicine, 15(11), e1002686.

[20] Tomasev, N., Glorot, X., Rae, J. W., Zielinski, M., et al. (2019). A clinically applicable approach to continuous prediction of future acute kidney injury. Nature, 572(7767), 116–119.