Early Detection of Lung Cancer Using Machine Learning Techniques

Authors:
Shmeleva Mariia, email: shmeleva.mm@edu.spbstu.ru
Khrestianovskii Daniil, email: hrestyanovskij.dd@edu.spbstu.ru
Kartalieva Farida, email: kartalieva.fi@edu.spbstu.ru
Miheeva Alina, email: miheeva.ad@edu.spbstu.ru

Additional Contributors:
Espinola Rivera Holger Elias, email: espinola_rh@spbstu.ru
Pak Vadim Gennadievich, email: pak_vg@spbstu.ru
Zaborovsky Vladimir Sergeevich, email: vladimir.zaborovsky@spbstu.ru

Abstract

Lung cancer remains one of the deadliest diseases worldwide, primarily due to its late diagnosis and limited treatment options at advanced stages. Traditional diagnostic techniques, such as radiological imaging and biopsies, are expensive and time-consuming. This research explores the use of machine learning (ML) techniques to provide fast and automated lung cancer detection based on patient-reported medical data.

We utilize the Survey Lung Cancer dataset, consisting of 284 samples, 15 features, and 2 target classes: positive and negative diagnoses of lung cancer. The features include various patient attributes, such as demographic information, lifestyle factors, and symptom-based indicators. We implement rigorous preprocessing techniques, including data cleaning, feature engineering, and class balancing using the Synthetic Minority Oversampling Technique (SMOTE). The study evaluates two classification models: k-Nearest Neighbors (kNN) and Support Vector Machine (SVM), optimizing their hyperparameters during the training phase using GridSearchCV. Model performance is compared using standard metrics such as accuracy, precision, recall, and F1-score. SVM outperformed kNN across all metrics, achieving higher accuracy (0.97 vs. 0.94), precision (0.97 vs. 0.94), recall (0.97 vs. 0.94), and F1-score (0.97 vs. 0.94). Notably, SVM achieved a recall of 0.98 for cancer-positive cases, compared to 0.92 with kNN—an essential improvement in medical diagnosis where minimizing false negatives is critical. These results highlight SVM as a more reliable model for early lung cancer detection.

Additionally, we develop a web-based diagnostic system, where healthcare professionals can input patient features to receive real-time predictions of cancer condition. The application backend is implemented in Flask and integrated with a PostgreSQL database, ensuring secure storage and efficient retrieval of diagnostic data.

1 Introduction

Lung cancer is one of the most lethal diseases worldwide, accounting for millions of deaths annually. The high mortality rate is primarily due to late-stage detection, where treatment options become limited, and the survival rate drops significantly. Early diagnosis of lung cancer is crucial for increasing life expectancy and improving treatment efficacy. However, the traditional diagnostic approaches, such as computed tomography (CT) scans, positron emission tomography (PET) scans, and biopsy analysis, require highly specialized medical expertise, advanced imaging equipment, and are often costly and time-consuming (Saif, Tzannou, Makrilia Syrigos, 2010). The rise of artificial intelligence (AI) and machine learning (ML) has revolutionized the field of medical diagnostics, offering new opportunities to automate and enhance disease detection (Huang, Li, Jiang, Yang & Li, 2024). ML models are capable of learning complex patterns from structured datasets, enabling early diagnosis and reducing the dependency on expensive imaging techniques. Specifically, ML-based classification models can process patient-reported symptoms and demographic data to estimate lung cancer risk levels, assisting healthcare professionals in identifying high-risk individuals.

Despite the potential benefits of ML in medical diagnostics, several challenges need to be addressed before these models can be effectively deployed in clinical environments. Medical datasets often suffer from class imbalance, where the number of positive cases (cancer patients) is significantly lower than the negative cases (healthy individuals). This imbalance can lead to biased models that favor the majority class, thereby increasing false negatives and reducing the model's ability to detect cancer patients correctly (Li, Liu & Hu, 2010). Additionally, medical datasets are often relatively small due to privacy concerns and the difficulty of collecting large-scale patient records. Thus, robust preprocessing techniques, feature engineering, and data augmentation strategies, such as the Synthetic Minority Oversampling Technique (SMOTE), are necessary to improve the reliability and fairness of ML models (Ishwaran & O'Brien, 2021).

This study investigates the effectiveness of two widely used machine learning classification models: k-Nearest Neighbors (kNN) and Support Vector Machine (SVM), for lung cancer detection. These models were selected based on their ability to handle structured tabular data and their interpretability, which is essential in medical applications (Nabeel et al., 2024). The dataset used in this study contains patient demographic information, lifestyle habits, and symptom-related attributes, providing a holistic view of potential risk factors associated with lung cancer. The dataset is preprocessed using feature normalization, categorical encoding, and SMOTE to balance class distribution.

The performance of the models is evaluated using key classification metrics such as accuracy, precision, recall, and F1-score. In medical applications, recall is particularly important as it measures the ability of the model to correctly identify positive cases. A high recall value ensures that the number of false negatives (missed cancer cases) is minimized, making the model more suitable for early diagnosis and intervention.

Beyond model development, this study also focuses on the practical implementation of the trained ML model into a user-friendly web-based application. The application is designed to provide medical professionals with a fast and reliable tool for assessing lung cancer risk based on patient input. The backend of the application is developed using Flask, which serves as an API for handling prediction requests and managing patient data. The frontend is built using React.js, ensuring an intuitive interface for healthcare professionals. The system is connected to a PostgreSQL database, which securely stores patient records and diagnostic predictions, allowing for seamless integration with medical record systems.

This research contributes to the growing body of work on AI-driven medical diagnostics by demonstrating the feasibility of using ML models for lung cancer prediction. By integrating these models into a web-based platform, we aim to enhance accessibility, scalability, and usability in real-world medical settings through deployment of intelligent medical system.

This study not only demonstrates the potential of machine learning in lung cancer detection but also bridges the gap between theoretical research and practical application by implementing a fully functional diagnostic intelligent system. The results of this research could serve as a foundation for future developments in AI-assisted medical diagnostics, ultimately contributing to early cancer detection and improved patient outcomes.

2 Methodology

2.1 Data Processing and Preprocessing

The quality and reliability of machine learning models depend significantly on the preprocessing of input data. In medical applications, where datasets often contain noise, missing values, and imbalanced class distributions, robust preprocessing techniques are essential for achieving high predictive perfor-

mance. This section details the comprehensive data preprocessing pipeline used in this study, covering data cleaning, feature selection, encoding methods, handling missing values, class balancing techniques, and feature scaling.

The dataset used in this study is the Survey Lung Cancer dataset, which consists of 309 patient records with 16 attributes, including demographic information, lifestyle factors, and symptom-based indicators. The attributes include:

Feature	Description
GENDER	Patient's gender (Male/Female)
AGE	Age of the patient in years
SMOKING	Indicates if the patient is a smoker
YELLOW FINGERS	Presence of nicotine stains on fingers
ANXIETY	Reported anxiety levels
PEER PRESSURE	Influence of social environment
CHRONIC DISEASE	Presence of chronic medical conditions
FATIGUE	Symptoms of exhaustion
ALLERGY	History of allergies
WHEEZING	Presence of wheezing in breathing
ALCOHOL CONSUMPTION	Regular alcohol consumption
COUGHING	Persistent coughing symptom
SHORTNESS OF BREATH	Difficulty breathing
SWALLOWING DIFFICULTY	Issues with swallowing food
CHEST PAIN	Chest pain associated with lung conditions
LUNG CANCER	Binary classification (Cancer/No Cancer)

Table 1: Dataset Features and Descriptions

Data cleaning is a crucial step in machine learning pipelines, particularly in medical datasets where inconsistencies and missing values can affect model accuracy. The dataset used in this study was derived from survey responses, leading to several data quality issues.

The dataset presented multiple challenges:

- Self-reported bias: Patients may have provided inaccurate information on symptoms and medical history.
- Inconsistent labeling: Variations in categorical data representation.

Several features, including age and chronic disease status, contained missing values. The following strategies were used:

- Numerical Features: Missing values were imputed using the median to reduce the effect of outliers.
- Categorical Features: Missing values were replaced with the most frequent category.
- Removing highly incomplete records: Entries missing more than 50% of features were discarded, though minimal removal was applied due to the small dataset size.

While these methods improved data completeness, they introduced potential biases, as imputed values may not accurately reflect real patient conditions.

Since machine learning models require numerical input, categorical variables were transformed into numerical representations using the following encoding method: Binary categorical variables such as "SMOKING" and "GENDER" were converted into 0 and 1 values.

Feature scaling was applied using min-max normalization to bring numerical features into a similar range, improving convergence for models like SVM that rely on distance-based computations.

One of the significant challenges in medical datasets is class imbalance, where the number of patients diagnosed with lung cancer is significantly lower than those without the disease. A model trained on an imbalanced dataset tends to favor the majority class, resulting in high accuracy but poor recall for minority class predictions. To address this, the Synthetic Minority Oversampling Technique (SMOTE) was applied.

SMOTE works by selecting existing minority class samples and generating new synthetic data points along the feature space between a sample and its nearest neighbors. This approach helps improve model learning without simply duplicating existing cases, making the classifier more generalizable.

Algorithm 1. Synthetic Minority Oversampling Technique (SMOTE):

1. Choose the instances from the minority class for which synthetic samples need to be generated.

2. For each selected minority class instance, find its k nearest neighbors in the feature space. The value of k is a hyperparameter that can be adjusted.

3. Create synthetic samples by interpolating between the selected instance and its nearest neighbors. The synthetic sample x_{synt} is generated as:

$$x_{\text{synt}} = x_i + \lambda \cdot (x_j - x_i)$$

   where x_i is the original instance, x_j is one of its nearest neighbors, and λ is a random value between 0 and 1. This process adds variation and generates new, plausible data points.

4. Repeat the process until the desired balance between the minority and majority class is achieved. This can be done by generating enough synthetic instances to balance the class distribution.

The data distribution before and after applying SMOTE was analyzed to observe its balancing effect:

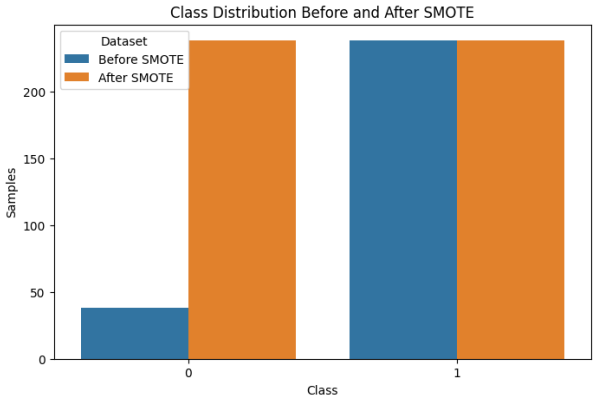

Figure 1: Class Distribution Before And After SMOTE

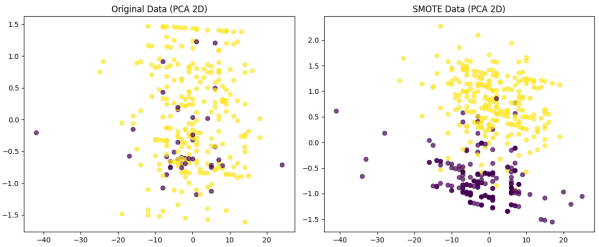

Figure 2: PCA-Based Visualization of Class Distribution Before and After SMOTE

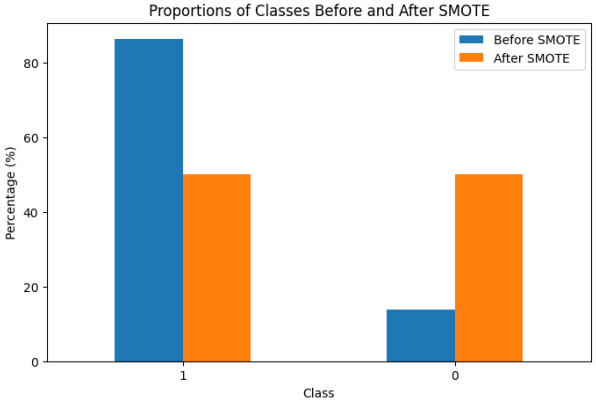

Figure 3: Proportions of Classes Before and After SMOTE

Feature selection is crucial for improving model efficiency and reducing overfitting. Highly correlated features introduce redundancy, making the model more complex without adding predictive power. To address this, a correlation heatmap was generated to visualize feature relationships, and redundant features with correlation coefficients above 0.5 were considered for removal.

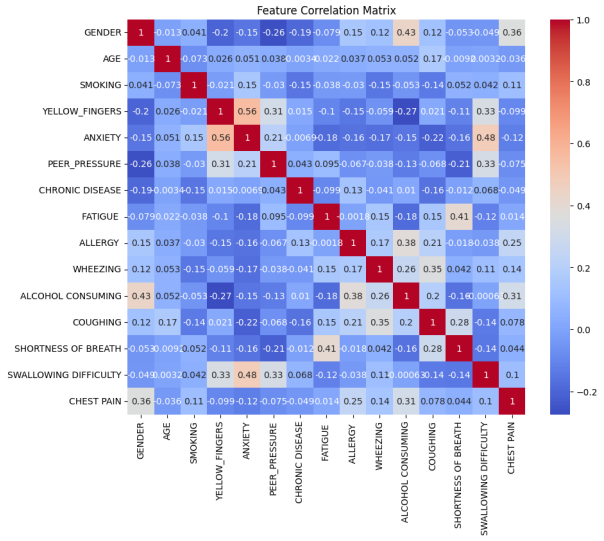

Figure 4: Feature Correlation Matrix Before Removal

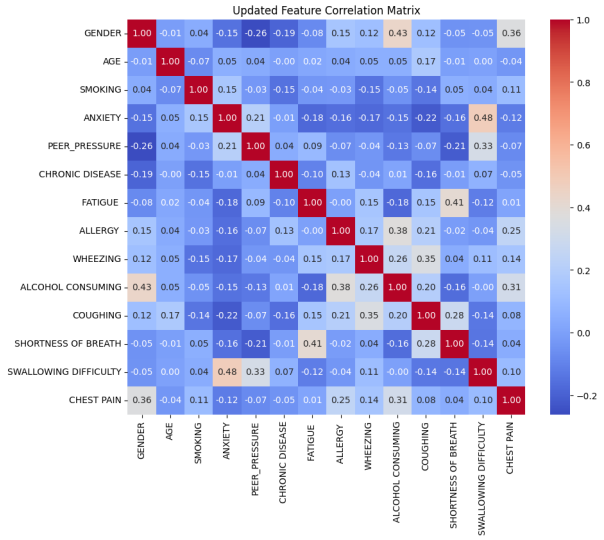

Figure 5: Feature Correlation Matrix After Removal

After preprocessing, the final dataset was split into training and testing sets in a 75:25 ratio. Stratified sampling was used to ensure that both classes were proportionally represented in the training and testing data, preventing data leakage and improving generalization.

This section detailed the data processing and preprocessing steps essential for training an effective lung cancer prediction model. The dataset was cleaned, transformed, and balanced using SMOTE to address class imbalance. Feature selection and scaling ensured that the input data was optimized for machine learning models. The next section will explore the specific machine learning models used in this study and their implementation.

2.2 Machine Learning Models

The selection of appropriate machine learning models plays a crucial role in ensuring accurate and reliable predictions in medical diagnostics. In this study, we considered several classification algorithms and ultimately selected k-Nearest Neighbors (kNN) and Support Vector Machine (SVM) due to their robustness in handling structured medical data, ease of interpretability, and effectiveness in small-to-moderate-sized datasets.

2.2.1 kNN

kNN is a simple yet powerful instance-based learning algorithm that classifies a given data point by finding the k most similar instances in the training set and assigning the majority class label among them. It relies on the assumption that similar instances exist in close proximity in feature space.

Given a dataset $D = \{(x_1, y_1), (x_2, y_2), ..., (x_n, y_n)\}$, where x_i represents the feature vector and y_i is the corresponding class label, the kNN algorithm operates as follows:

Algorithm 2. k-Nearest Neighbors (kNN)

1. Compute the distance $d(x, x_i)$ between the test point x and all training samples using a chosen distance metric.

2. Select the k nearest neighbors with the smallest distances.

3. Assign the majority class label among the selected neighbors to the test point x.

kNN is simple to implement and requires minimal tuning. It works well with well-separated classes. Also, it has no explicit training phase, making it efficient for small datasets.

However, kNN is computationally expensive for large datasets due to distance calculations (Raikwal & Saxena, 2012). It is sensitive to irrelevant or highly correlated features. In addition, it requires careful selection of k to avoid overfitting or underfitting.

For hyperparameter tuning, we used GridSearchCV and identified the optimal kNN parameters as follows:

Hyperparameter	Value
Algorithm	ball_tree
Metric	Manhattan
Number of neighbors	5
Weights	Distance-based

Table 2: Optimal hyperparameters for kNN

2.2.2 SVM

Support Vector Machine (SVM) is a supervised learning algorithm that constructs an optimal hyperplane to maximize class separation. It is particularly effective for high-dimensional datasets and small sample sizes, making it a suitable choice for medical diagnostics.

Given a binary classification problem, SVM aims to find a hyperplane defined as:

$$f(x) = w^T x + b$$

where w is the weight vector, x is the input feature vector, and b is the bias term. The optimal hyperplane maximizes the margin between the two classes:

$$\max \frac{1}{\|w\|} \quad \text{subject to} \quad y_i(w^T x_i + b) \geq 1, \quad \forall i$$

where $y_i \in \{-1, +1\}$ are the class labels.

The SVM algorithm operates as follows:

Algorithm 3. Support Vector Machine (SVM):

1. Input: Training data $\{(x_i, y_i)\}_{i=1}^n$, with $x_i \in \mathbb{R}^d$, $y_i \in \{-1, +1\}$.

2. Choose regularization parameter C and kernel function $K(x_i, x_j)$.

3. Formulate the optimization problem:

$$\min_{w,b,\xi} \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{n} \xi_i \quad \text{s.t.} \quad y_i(w^T x_i + b) \geq 1 - \xi_i,\ \xi_i \geq 0$$

4. Apply kernel trick (if necessary) to map inputs into a higher-dimensional space:

$$K(x_i, x_j) = \phi(x_i)^T \phi(x_j)$$

5. Identify support vectors and compute bias term b.

6. Construct decision function:

$$f(x) = \sum_{i \in SV} \alpha_i y_i K(x_i, x) + b$$

7. Predicted label: $\hat{y} = \text{sign}(f(x))$

SVM handles both linear and non-linear decision boundaries using kernel functions. It is robust to overfitting and performs well on small, complex datasets. However, it can be computationally expensive on large datasets (Platt, 1999) and requires careful hyperparameter tuning.

Hyperparameter tuning for SVM was performed using GridSearchCV, and the best parameters were identified as follows:

Hyperparameter	Value
C (Regularization parameter)	0.1
Class weight	Balanced
Degree	2
Gamma	Auto
Kernel	Polynomial

Table 3: Optimal hyperparameters for SVM

3 Experimental Results

3.1 ML Operations

To compare the performance of kNN and SVM, we conducted experiments using multiple evaluation metrics, emphasizing recall due to its importance in minimizing false negatives in medical applications.

In this study, class 1 represents patients with lung cancer (positive cases), while class 0 represents patients without lung cancer (negative cases).

The dataset was split into training and testing sets with a 75% / 25% ratio, where the distribution of samples per class was as follows:

- Training set: Class 1 – 179 samples, Class 0 – 178 samples
- Testing set: Class 1 – 59 samples, Class 0 – 60 samples

Model	Accuracy	Precision	Recall	F1-score
kNN	0.94	0.94	0.94	0.94
SVM	0.97	0.97	0.97	0.97

Table 4: Performance Comparison of kNN and SVM for overall data

KNN Classification Report	Precision	Recall	F1-score	Support
Class 0	0.92	0.97	0.94	60
Class 1	0.96	0.92	0.94	59
Accuracy			0.94	119
Macro avg	0.94	0.94	0.94	119
Weighted avg	0.94	0.94	0.94	119

Table 5: KNN Classification Report by class

SVM Classification Report	Precision	Recall	F1-score	Support
Class 0	0.98	0.97	0.97	60
Class 1	0.97	0.98	0.97	59
Accuracy			0.97	119
Macro avg	0.97	0.97	0.97	119
Weighted avg	0.97	0.97	0.97	119

Table 6: SVM Classification Report by class

SVM outperformed kNN across all evaluation metrics, achieving higher accuracy (0.97 vs. 0.94), precision (0.97 vs. 0.94), recall (0.97 vs. 0.94), and F1-score (0.97 vs. 0.94). However, the most crucial difference is in recall for class 1 (cancer-positive cases), where SVM achieved 0.98 compared to 0.92 for kNN. Since recall measures the ability to correctly identify positive cases, this improvement is critical in medical applications, where false negatives (misclassifying a cancer patient as healthy) must be minimized (Hicks et al., 2022). A recall of 0.98 means that 98% of cancer-positive patients were correctly identified by the model, making SVM the more reliable choice for lung cancer detection. Given that early and accurate diagnosis is essential in medical decision-making, these results strongly support the use of SVM over kNN for this task.

3.2 Web Application Development

Deploying the trained machine learning model into a real-world clinical environment requires an intuitive, scalable, and secure web-based system. The primary goal of the web application is to provide healthcare professionals with an easy-to-use tool for inputting patient data, obtaining lung cancer risk predictions, and securely storing medical records. This section describes the architecture, technologies, and implementation details of the web application.

3.2.1 System Architecture

The web application follows a client-server architecture, where the frontend interacts with the backend API to send user input and receive predictions. The overall system is divided into three primary components:

- **Backend (Flask)** – Handles request processing, ML model inference, and communication with the database;
- **Frontend (React.js)** – Provides an interactive user interface for medical professionals to enter patient details and view predictions;
- **Database (PostgreSQL)** – Stores patient data and predictions for future reference and analysis.

The system operates as follows:

1. **User Interaction (Frontend)**: The user inputs medical details into a React.js web form, which collects fields such as age, smoking habits, and other health-related factors. Upon submission, the form sends a JSON request to the backend.
2. **Data Processing & Prediction (Backend)**: The Flask backend validates the received data, converts categorical variables

into numerical format (e.g., boolean values to 0/1), and sends the processed data to the machine learning model.

3. **Machine Learning Model (SVM)**: The pre-trained Support Vector Machine (SVM) model, stored as a `.pkl` file, performs classification and returns:
   - **predicted_class**: 0 (No lung cancer) or 1 (Lung cancer detected).
   - **probability_class**: Confidence score for both classes.
4. **Data Persistence (PostgreSQL)**: The backend stores user details, medical symptoms, and prediction results in the PostgreSQL database.

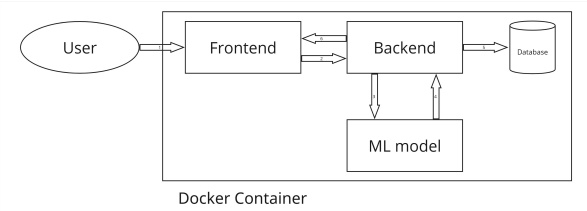

Figure 6: Integration of Intelligent System Components

3.2.2 Frontend

The frontend of the application is developed using React.js, which provides a modular, maintainable, and dynamic interface. It enables users to input patient data and receive real-time predictions in a seamless manner. The interface is designed to be intuitive, ensuring smooth interaction with the system while maintaining a professional and user-friendly experience.

To enhance the usability and appearance of the application, Material-UI (MUI) is used as the primary styling framework. It offers a collection of pre-built components that ensure consistency in design. Navigation between different sections of the application is managed using React Router, allowing users to move effortlessly between pages. Since predictions require a short processing time, a loading animation is implemented with the React Loader Spinner library to indicate system activity and prevent confusion.

Following a component-based architecture, the frontend consists of several key elements. The Prediction Form serves as the primary input interface, where users enter medical data, such as symptoms and health indicators, before submitting the request for analysis. Once the backend processes the data and returns a result, the Prediction Results component dynamically displays the classification outcome, ensuring clarity and readability. A Navigation Bar is integrated to provide structured access to different sections of the application, offering a streamlined user experience. Additionally, a Loader Component enhances responsiveness by visually indicating when predictions are being processed.

React's state management ensures efficient handling of user inputs and interaction flows, allowing for a smooth and uninterrupted experience. Combined with the flexibility of Material-UI, the frontend achieves both high performance and a clean, professional design.

3.2.3 Backend

The backend, implemented with Flask, processes user requests, handles ML model inference, and interacts with the database.

REST API Endpoints:

- **POST /predict** – Accepts JSON-formatted patient data and returns a lung cancer prediction.
- **GET /patients** – Retrieves stored patient records from the database.
- **POST /register** – Registers new users with authentication.
- **POST /login** – Handles user authentication.
- **GET /prediction-history** – Fetches past predictions for a user.

The SVM model is loaded at backend initialization, ensuring efficient inference without reloading the model for each request.

3.2.4 Database

PostgreSQL serves as the primary database for storing user data, medical records, and prediction history. The database schema includes:

- **users** – Table which stores user credentials.
- **medical_results** – Table which contains demographic information and symptom data.

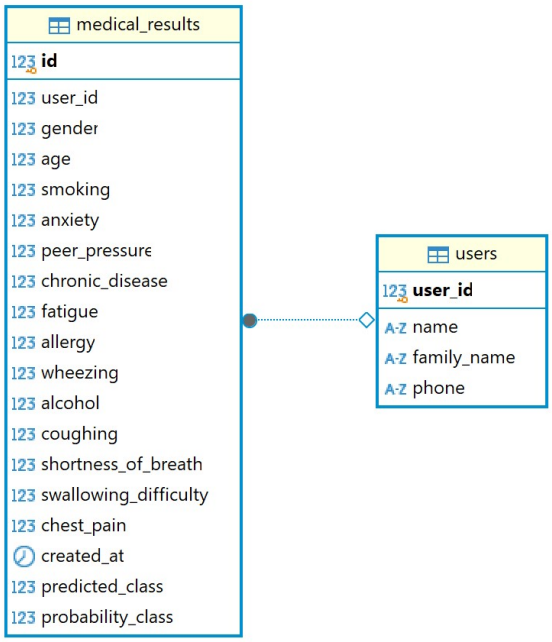

Figure 7: Database Schema

3.2.5 Deployment with Docker

To ensure consistency, scalability, and ease of deployment, the entire system is deployed inside Docker containers. Each component runs in its own isolated Docker environment, preventing dependency conflicts and enabling straightforward management. The frontend, developed in React.js, operates within a Node.js Docker container, while the backend, powered by Flask, runs in a Python Docker container. The PostgreSQL database is housed in a separate container, ensuring efficient storage and retrieval of patient records and prediction history.

By utilizing Docker Compose, the system orchestrates these services seamlessly, facilitating smooth communication between them. This setup not only simplifies deployment but also enhances flexibility, allowing the system to be deployed across different environments with minimal configuration. The containerized architecture ensures that each component remains independent, making updates and scalability straightforward. New services can be integrated effortlessly by modifying the Docker Compose configuration, reinforcing the system's adaptability to future improvements and expansions.

4 Conclusion and Future Work

The research presented in this paper demonstrates the potential of machine learning techniques for the early detection of lung cancer based on patient-reported symptoms. By leveraging structured survey data, we developed a predictive system that aids in identifying high-risk patients, potentially enabling earlier medical intervention and improving survival rates. The integration of a web-based diagnostic system further enhances accessibility, allowing medical professionals to efficiently assess lung cancer risk through an intuitive user interface.

Key contributions of this study include:

- A comprehensive data preprocessing pipeline, including missing value imputation, feature engineering, and class balancing using SMOTE.
- Machine learning experiments with k-Nearest Neighbors (kNN) and Support Vector Machine (SVM) for lung cancer detection, demonstrating the superior recall performance of SVM in medical diagnostics.
- The development and deployment of a full-stack web application incorporating Flask, PostgreSQL, and React.js to facilitate real-time predictions.

The results indicate that SVM achieves higher recall and F1-score, making it the preferred choice for medical applications where minimizing false negatives is crucial. The web application successfully bridges the gap between AI research and clinical implementation, providing a scalable and user-friendly platform for medical professionals.

Despite these achievements, several challenges and limitations were encountered. The dataset used in this study is relatively small, which may limit the generalizability of the models. Additionally, the reliance on self-reported symptoms introduces potential biases and inconsistencies in the data. Future enhancements are necessary to improve the system's robustness and clinical applicability, including the integration of a multimodal data layer that combines images and tabular data.

Collecting a larger and more diverse dataset from multiple sources, including hospital records and electronic health records (EHRs), will improve the generalizability of the model. Additionally, integrating real-world clinical data from CT scans, genetic markers, and laboratory test results could enhance the model's predictive power.

The current system is designed as a web application. Future iterations should include a mobile-friendly version to allow broader accessibility, especially in regions with limited healthcare infrastructure. Additionally, integrating wearable devices for real-time patient monitoring could provide continuous health assessments, improving early detection capabilities.

To ensure seamless adoption in clinical practice, the web application should be integrated with existing hospital information systems. This would allow automatic retrieval and storage of patient data, streamlining workflows for healthcare providers.

Future work should include longitudinal studies where the model is tested on real patient cases over time. This would provide insights into its real-world effectiveness, potential biases, and areas for improvement.

This research highlights the potential of AI-driven diagnostic tools for improving early lung cancer detection. By combining machine learning, data science, and web technologies, we have developed an accessible and scalable solution that can assist medical professionals in identifying high-risk patients. While challenges remain, ongoing advancements in AI and medical data integration will further enhance the accuracy, usability, and clinical adoption of such systems.

5 References

[1] Saif, M. W., Tzannou, I., Makrilia, N., & Syrigos, K. (2010). Role and cost effectiveness of PET/CT in management of patients with cancer. *The Yale Journal of Biology and Medicine, 83*(2), 53.

[2] Huang, D., Li, Z., Jiang, T., Yang, C., & Li, N. (2024). Artificial intelligence in lung cancer: current applications, future perspectives, and challenges. *Frontiers in Oncology, 14*, 1486310.

[3] Li, D. C., Liu, C. W., & Hu, S. C. (2010). A learning method for the class imbalance problem with medical data sets. *Computers in Biology and Medicine, 40*(5), 509–518.

[4] Ishwaran, H., & O'Brien, R. (2021). Commentary: The problem of class imbalance in biomedical data. *The Journal of Thoracic and Cardiovascular Surgery, 161*(6), 1940-1941.

[5] Nabeel, S. M., Bazai, S. U., Alasbali, N., Liu, Y., Ghafoor, M. I., Khan, R., & Por, L. Y. (2024). Optimizing lung cancer classification through hyperparameter tuning. *Digital Health, 10*, 20552076241249661.

[6] Raikwal, J. S., & Saxena, K. (2012). Performance evaluation of SVM and k-nearest neighbor algorithm over medical data set. *International Journal of Computer Applications, 50*(14).

[7] Platt, J. C. (1999). Fast training of support vector machines using sequential minimal optimization. In *Advances in Kernel Methods* (pp. 185-208). MIT Press.

[8] Hicks, S. A., Strümke, I., Thambawita, V., Hammou, M., Riegler, M. A., Halvorsen, P., & Parasa, S. (2022). On evaluation metrics for medical applications of artificial intelligence. *Scientific Reports, 12*(1), 5979.