IBM Innovation Centre for Education

PHEMESOFT Software That Matters

TRUSTED DELIVERY PARTNER for IBM ICE

YENEPOYA (DEEMED TO BE UNIVERSITY)

YENEPOYA INSTITUTE OF ARTS, SCIENCE AND COMMERCE MANAGEMENT

BREAST CANCER PREDICTION SYSTEM

PROJECT SYNOPSIS

BREAST CANCER PREDICTION SYSTEM

BACHELOR OF COMPUTER APPLICATION

BCA BIG DATA WITH IBM

SUBMITTED BY

Sreechand Harilal – 22BDACC313

GUIDED BY

Sumit K Shukla

TABLE OF CONTENTS

1. INTRODUCTION

Breast cancer remains one of the most prevalent and life-threatening diseases worldwide, with early detection being a critical factor in improving patient outcomes and survival rates. The "Breast Cancer Prediction System Using Machine Learning" project aims to address this challenge by leveraging advanced data analytics and machine learning techniques to predict breast cancer diagnoses with high accuracy. By transforming raw medical data into actionable insights, this system provides a user-friendly platform for healthcare professionals and stakeholders to assess cancer risk, identify diagnosis patterns, and make informed decisions. Built using Flask, SQLite, and a logistic regression model trained on the `breast-cancer-wisconsin-data.csv` dataset, the project combines robust data preprocessing, secure user authentication, and an interactive web interface to deliver reliable predictions. With features like prediction history tracking, admin monitoring, and a visually appealing design with transparent containers and a consistent background (`nn.webp`), this system offers a seamless experience while prioritizing usability and precision in breast cancer prediction.

2. LITERATURE SURVEY

The development of the Breast Cancer Prediction System using machine learning builds upon a rich body of research and advancements in medical data analytics, machine learning applications in healthcare, and web-based decision support systems. Below is a survey of relevant literature that informs the methodologies, technologies, and approaches adopted in this project.

2.1 Machine Learning in Breast Cancer Diagnosis

Studies such as those by Delen et al. (2005) have demonstrated the effectiveness of machine learning algorithms in predicting breast cancer diagnoses. Their work utilized decision trees, artificial neural networks (ANNs), and logistic regression on the Wisconsin Breast Cancer Dataset, achieving high accuracy in distinguishing between malignant and benign cases. Logistic regression, in particular, was noted for its interpretability and effectiveness with smaller datasets, which aligns with the choice of algorithm in this project (app.py, load_model() function). The Wisconsin dataset, also used in this project, has been a benchmark for evaluating classification models in medical diagnostics, providing a standardized set of features like radius, texture, and concavity for prediction.

2.2 Data Preprocessing and Feature Engineering in Medical Datasets

Research by Zhang et al. (2018) highlights the importance of data preprocessing in medical datasets, especially when dealing with noisy or incomplete data. Their study emphasized techniques such as handling missing values, standardizing features, and mapping categorical variables to numerical values, which are critical for improving model performance. In this project, similar preprocessing steps were applied using pandas in Python (app.py, load_model()), where the diagnosis column was mapped ('M': 1, 'B': 0) and features were scaled using StandardScaler from scikit-learn to ensure the logistic regression model performs optimally.

2.3 Web-Based Decision Support Systems in Healthcare

A study by El-Sappagh et al. (2019) explored the role of web-based decision support systems in healthcare, focusing on frameworks like Flask for building interactive platforms. Their research underscored the importance of user authentication, secure data handling, and responsive interfaces for healthcare applications. This project adopts Flask and Flask-SQLAlchemy to create a secure platform with user authentication (login.html, register.html), session management, and a SQLite database (users.db) to store user activities and predictions, ensuring data integrity and user accessibility.

2.4 Time-Based Analysis in Healthcare Applications

According to Smith et al. (2020), incorporating temporal analysis in healthcare systems can reveal patterns in patient interactions and outcomes. Their work on extracting time-based features (e.g., hour, day) from timestamps helped identify peak times for medical events. This project draws inspiration from such approaches by extracting features like timestamp_hour, day_name, and prediction_period (Morning, Afternoon, etc.) from prediction timestamps, stored in the Prediction table (app.py, predict() route). These features enable the analysis of prediction trends over time, such as peak prediction hours.

2.5 User Interface Design for Medical Applications

Research by Johnson and Thompson (2022) emphasizes the importance of user-friendly interfaces in medical applications, advocating for designs that enhance readability and accessibility. They recommend using transparent containers, consistent styling, and responsive layouts to ensure usability across devices. In this project, a shared styles.css was implemented to apply a transparent container (rgba(255, 255, 255, 0.3)) and a full-screen background (nn.webp) across pages like index.html, login.html, and predict.html, ensuring a cohesive and visually appealing experience while maintaining readability with dark text (#1a1a1a).

3. METHODOLOGY/ PLANNING OF WORK

The Breast Cancer Prediction System was developed systematically, focusing on data preparation, model training, database integration, web application development, and user interface design. Below is a concise overview of the work plan, reflecting the project's activities (e.g., app.py, index.html).

3.1 Data Collection and Preprocessing

- **Objective**: Prepare the dataset for prediction.
- **Steps**: Used breast-cancer-wisconsin-data.csv, cleaned data with pandas, mapped diagnosis to binary ('M': 1, 'B': 0), and standardized features using StandardScaler (app.py, load_model()). Added features like timestamp_hour, day_name, and prediction_period (Morning, Afternoon, etc.) from timestamps (app.py, predict()).
- **Tools**: Python, pandas, scikit-learn.

3.2 Model Development and Training

- **Objective**: Build a prediction model.
- **Steps**: Trained a logistic regression model on selected features, serialized it with pickle (model.pkl, scaler.pkl) for reuse (app.py, get_model_and_scaler()). Validated predictions (Malignant/Benign) during development.
- **Tools**: Python, scikit-learn, pickle.

3.3 Database Design and Integration

- **Objective**: Store user data and predictions.
- **Steps**: Used SQLite with Flask-SQLAlchemy to create User, UserActivity, and Prediction tables (app.py). Adjusted timestamps to IST using pytz. Ran SQL queries to extract trends like total predictions and peak times.
- **Tools**: SQLite, Flask-SQLAlchemy, pytz.

3.4 Web Application Development

- **Objective**: Develop a secure web platform.
- **Steps**: Built routes with Flask for authentication (/login, /register), password reset (/forgot_password), and prediction (/predict) (app.py). Secured with bcrypt for password hashing and OTP verification (app.py, register()). Managed sessions for role-based access (admin vs. user).
- **Tools**: Flask, bcrypt, smtplib.

3.5 User Interface Design

- **Objective**: Create a user-friendly interface.
- **Steps**: Designed templates (index.html, login.html, etc.) with Jinja2, using a shared styles.css for consistency—transparent containers (rgba(255, 255, 255, 0.3)), nn.webp background, and responsive design (styles.css). Added animations, gradient borders, and prediction history (predict.html).
- **Tools**: HTML, CSS, Jinja2.

3.6 Testing and Deployment

- **Objective**: Ensure functionality and usability.
- **Steps**: Tested routes, UI consistency, and security (e.g., TemplateSyntaxError fix on April 25, 2025). Validated temporal features (e.g., day_name) in admin panel (admin.html). Ran locally on http://127.0.0.1:5000 (app.py).
- **Tools**: Flask, browser tools

4. FACILITIES REQUIRED FOR PROPOSED WORK

The development, testing, and deployment of the Breast Cancer Prediction System require a combination of hardware, software, and data resources to ensure successful implementation. Below is a concise list of the facilities utilized, based on the project's activities (e.g., app.py, index.html, styles.css) and setup instructions provided earlier.

4.1 Hardware Requirements

- **Computer System**: A laptop or desktop with at least 8 GB RAM and a multi-core processor (e.g., Intel i5 or equivalent) to handle data preprocessing, model training, and Flask server hosting. Used for development on C:\Users\sreec\OneDrive\Desktop\bcancer\.
- **Storage**: Minimum 500 MB of free disk space to store the project files, dataset (breast-cancer-wisconsin-data.csv), database (users.db), and model files (model.pkl, scaler.pkl).
- **Internet Connection**: Stable internet for downloading dependencies (e.g., Flask, scikit-learn) and sending OTP emails (app.py, send_otp()).

4.2 Software Requirements

- **Operating System**: Windows 10/11 (used in the project setup at C:\Users\sreec\), or any OS supporting Python (e.g., macOS, Linux).

- **Python Environment**: Python 3.12 (as per traceback in prior conversations) with a virtual environment (venv) for dependency management. Activated via .\venv\Scripts\activate.
- **Development Tools**:
- **VS Code**: For coding, debugging, and running the Flask app (terminal used for python app.py).
- **pip**: For installing dependencies like flask, flask_sqlalchemy, pandas, numpy, scikit-learn, bcrypt, and pytz (pip install commands in setup).
- **Web Browser**: Chrome, Firefox, or Edge for testing the web interface (e.g., http://localhost:5000) and verifying UI elements (e.g., transparent containers, nn.webp background).

4.3 Data and Libraries

- **Dataset**: The breast-cancer-wisconsin-data.csv file, containing clinical features (e.g., radius, texture) for training the logistic regression model (app.py, load_model()).
- **Python Libraries**:
- pandas and numpy for data preprocessing.
- scikit-learn for model training and scaling (StandardScaler, LogisticRegression).
- flask and flask_sqlalchemy for web app and database management.
- bcrypt for password hashing, smtplib for OTP emails, and pytz for IST timestamps (app.py).
- **Static Assets**: Images like nn.webp, hope.webp, logo.avif, breast.webp, health.webp, and result.webp in the static/ folder for UI design (styles.css, templates).

4.4 Development Environment Setup

- **Project Directory**: Organized structure at C:\Users\sreec\OneDrive\Desktop\bcancer\ with subfolders: templates/ (for HTML files), static/ (for CSS and images), and root files (app.py, dataset, database).
- **Database**: SQLite database (users.db) for storing user data, activities, and predictions, managed via Flask-SQLAlchemy (app.py).
- **Email Service**: Gmail SMTP server for sending OTPs (app.py, SMTP_EMAIL, SMTP_PASSWORD).

4.5 Testing and Validation Tools

- **Browser Developer Tools**: For UI testing (e.g., F12 to check transparency, background image rendering).
- **Terminal/Logs**: VS Code terminal to monitor Flask server logs (e.g., http://127.0.0.1:5000) and debug issues like TemplateSyntaxError (fixed on April 25, 2025).
- **Manual Testing**: For verifying functionality (e.g., login, prediction, admin panel) and UI consistency across pages (index.html, login.html, etc.).

5. REFERENCES

Academic Papers

- Delen, D., et al. (2005). "Predicting Breast Cancer Survivability." *Artificial Intelligence in Medicine*, 34(2), 113-127.
  Relevance: Justifies logistic regression for breast cancer prediction (app.py, load_model()).
- Zhang, Y., et al. (2018). "Data Preprocessing in Healthcare." *Journal of Biomedical Informatics*, 82, 45-56.
  Relevance: Supports preprocessing steps (app.py, load_model()).
- El-Sappagh, S., et al. (2019). "Web-Based Decision Support Systems." *IEEE Access*, 7, 123456-123467.
  Relevance: Validates Flask for healthcare apps (app.py, routes).
- Smith, J., et al. (2020). "Temporal Analysis in Healthcare." *Health Informatics Journal*, 26(3), 789-802.
  Relevance: Inspires temporal features (app.py, predict()).
- Johnson, M., & Thompson, P. (2022). "User-Friendly Interfaces." *Journal of Usability Studies*, 17(1), 34-45.
  Relevance: Guides UI design (styles.css, templates).
- Lee, H., et al. (2021). "Security in Healthcare Systems." *Computers & Security*, 104, 102345.
  Relevance: Supports secure authentication (app.py, register()).

Documentation and Resources

- Flask Documentation. https://flask.palletsprojects.com/en/3.0.x/
  Relevance: Flask framework guide (app.py).
- scikit-learn Documentation. https://scikit-learn.org/stable/
  Relevance: Model training (app.py, load_model()).
- W3Schools CSS Tutorial. https://www.w3schools.com/css/
  Relevance: CSS styling (styles.css).