Department of Computer Science & Engineering

MALLA REDDY UNIVERSITY, HYDERABAD

2024-2025

Appendix-1

Mammary Carcinoma

Designed and Developed by

1. M. Bhargav 2211CS010363
2. P. Charan Sai 2211CS010459
3. N. Uday Kumar 2211CS010407
4. S. Harsha Vardhan 2211CS010514

Guided by

Mr. G. Raju

Department of Computer Science & Engineering

MALLA REDDY UNIVERSITY, HYDERABAD

2024-2025

Appendix-2

MALLA REDDY UNIVERSITY

(Telangana State Private Universities Act No.13 of 2020 and G.O.Ms.No.14, Higher Education (UE) Department)

CERTIFICATE

This is to certify that this is the Application development lab record entitled "**MAMMARYCARCINOMA**", submitted by **M.BHARGAV(2211CS010363),P.CHARAN SAI(2211CS010459),N.UDAYKUMAR(2211CS010407),S.HARSHAVARDHAN(2211CS 010514)** B. Tech **III** year **II** semester, Department of CSE during the year 2024-25. The results embodied in this report have not been submitted to any other university or institute for the award of any degree or diploma.

Internal Guide
Mr. G. Raju

Dean-CSE
Dr. Shaik Meeravali

External Examiner

Appendix-3

MALLA REDDY UNIVERSITY
(Telangana State Private Universities Act No.13 of 2020 and G.O.Ms.No.14, Higher Education (UE) Department)

DECLARATION

I declare that this project report titled **MAMMARY CARCINOMA** submitted in partial fulfillment of the degree of B. Tech in CSE is a record of original work carried out by me under the supervision of **Mr. G. Raju** and has not formed the basis for the award of any other degree or diploma, in this or any other Institution or University. In keeping with the ethical practice in reporting scientific information, due acknowledgements have been made wherever the findings of others have been cited.

M. Bhargav 2211CS010363
P. Charan Sai 2211CS010459
N. Uday Kumar 2211CS010407
S. Harsha Vardhan 2211CS010514

Appendix-4

ACKNOWLEDGEMENTS

With great pleasure we want to take this opportunity to express our heartfelt gratitude to all the people who helped in making this mini project work a grand success. We express our deep sense of gratitude to **Mr. G. Raju** for his constant guidance throughout our mini project work. We would like to thank **Dr. SHAIK MEERAVALI**, Dean of the Department of Computer Science and Engineering, for providing seamless support and right Suggestions in the development of project.

Appendix-5

ABSTRACT

Breast cancer is one of the leading causes of cancer-related deaths among women worldwide. Early and accurate detection is crucial for improving survival rates and treatment effectiveness. This project leverages machine learning techniques to classify breast cancer cases as malignant or benign based on medical attributes.

We utilize four machine learning models-Logistic Regression, Support Vector Machine (SVM), Decision Tree, and Random Forest-to analyze and predict breast cancer diagnoses. The dataset consists of patient attributes such as mean radius, perimeter, texture, and other cellular features. The project includes data preprocessing, feature selection, model training, and evaluation to ensure optimal prediction accuracy.

Additionally, data visualization techniques, including heatmaps, histograms, and KDE plots, are implemented to understand feature correlations and distributions. A Flask-based web application is developed to provide an interactive interface where users can input feature values and receive real-time predictions.

The results demonstrate the effectiveness of machine learning in breast cancer detection, providing a tool that can assist medical professionals in diagnosis. By integrating AI-driven predictive analytics, this project aims to contribute to early detection, reducing false positives, and improving patient outcomes.

Appendix-6

TABLE OF CONTENT

LIST OF FIGURES

CHAPTER – 1
INTRODUCTION
MAMMARY CARCINOMA

1.1 Introduction

Breast cancer is one of the most prevalent cancers worldwide, affecting millions of women each year. Early detection plays a crucial role in improving survival rates and ensuring timely treatment. Traditional diagnostic methods, such as mammography and biopsy, often require expert interpretation and can be time-consuming. With advancements in artificial intelligence and machine learning, automated diagnostic systems can assist medical professionals by providing faster and more accurate predictions.

This project aims to develop a Breast Cancer Detection System using machine learning techniques to classify tumors as benign (non-cancerous) or malignant (cancerous). The system is built using four machine learning algorithms-Logistic Regression, Support Vector Machine (SVM), Decision Tree, and Random Forest-to analyze patient medical attributes and predict cancer status.

The dataset used in this study contains various cellular characteristics, such as mean radius, texture, perimeter, and symmetry, which help in determining the nature of the tumor. The project follows a structured workflow involving data preprocessing, feature selection, model training, evaluation, and visualization to enhance predictive accuracy.

To make the system accessible, a Flask-based web application is integrated, allowing users to input feature values and obtain real-time predictions. Additionally, data visualization techniques like heatmaps and histograms are implemented to understand feature relationships and improve model interpretability.

By leveraging machine learning and data analytics, this project aims to contribute to the early detection of breast cancer, assisting healthcare professionals in making informed decisions and ultimately improving patient outcomes.

1.2 Problem Statement

Mammary carcinoma means a Breast cancer is a leading cause of mortality among women worldwide, with early detection being critical for successful treatment and improved survival rates. Traditional diagnostic methods, such as biopsies and mammograms, are often time-consuming, expensive, and reliant on expert interpretation, leading to delays in diagnosis and potential human errors.

The challenge lies in developing an efficient, accurate, and automated breast cancer detection system that can assist healthcare professionals in making quick and reliable diagnoses. Machine learning techniques can help in analyzing medical data, identifying patterns, and predicting whether a tumor is benign or malignant based on patient attributes.

This project aims to address the following key issues:

1. **Accuracy of Detection** – Traditional methods may sometimes produce false positives or false negatives. The objective is to develop an ML model that minimizes misclassification.
2. **Feature Selection** – Identifying the most relevant features from the dataset to improve predictive performance.
3. **Model Comparison** – Evaluating multiple machine learning models (Logistic Regression, SVM, Decision Tree, and Random Forest) to determine the most effective one.
4. **Data Interpretation** – Visualizing relationships between different features to understand their impact on classification.
5. **User-Friendly Interface** – Developing a Flask-based web application to allow users to input medical attributes and receive instant predictions.

By implementing this machine learning-based solution, the project aims to enhance breast cancer diagnosis by providing a fast, accurate, and interpretable prediction system, ultimately supporting early detection efforts and improving patient outcomes.

1.3 Objective

The primary goal of this project is to develop an efficient and accurate machine learning-based breast cancer detection system that can classify tumors as benign or malignant using patient attributes. The following objectives guide the development of the system:

1. **Data Preprocessing & Feature Engineering**
    - Handle missing values and normalize data for better model performance.
    - Encode categorical variables (e.g., diagnosis labels) for machine learning compatibility.
    - Perform feature selection to identify the most significant attributes affecting breast cancer classification.
2. **Implementation of Machine Learning Models**
    - Train and compare multiple machine learning algorithms, including **Logistic** Regression, Support Vector Machine (SVM), Decision Tree, and Random Forest.
    - Evaluate models based on accuracy, precision, recall, and F1-score to determine the best-performing algorithm.
    - Optimize models using hyperparameter tuning to enhance predictive performance.
3. **Data Visualization & Statistical Analysis**
    - Use Seaborn, Matplotlib, and Pandas to generate visual representations like heatmaps, histograms, and KDE plots for better insight into data distribution.
    - Analyze feature correlations to understand their impact on breast cancer diagnosis.
4. **Development of a Flask-Based Web Application**
    - Create an interactive Flask web interface where users can input medical attributes and obtain real-time cancer classification results.
    - Ensure a user-friendly, responsive, and visually appealing UI for ease of access by healthcare professionals and researchers.
5. **Deployment & Performance Optimization**
    - Deploy the application on a cloud-based platform for accessibility.
    - Ensure model efficiency, scalability, and reliability for practical use in medical diagnostics.

1.4 Goal of project

The primary goal of this project is to develop an intelligent, data-driven, and efficient Breast Cancer Detection System utilizing machine learning techniques to aid in early diagnosis and improve patient outcomes. By leveraging advanced predictive models, the system aims to enhance the accuracy of cancer detection, enabling timely medical intervention and reducing mortality rates associated with breast cancer.

This project is designed to achieve the following overarching goals:

1. **Early and Accurate Diagnosis**
    - Detect breast cancer at an early stage using machine learning classification models that analyze patient attributes such as mean radius, texture, perimeter, and other critical parameters.
    - Improve diagnostic accuracy over traditional methods by incorporating data-driven predictions rather than relying solely on manual evaluation.
2. **Development of a Robust Predictive Model**
    - Implement and compare multiple machine learning models, including Logistic Regression, Support Vector Machine (SVM), Decision Tree, and Random Forest, to determine the most effective algorithm for breast cancer classification.
    - Optimize model performance through hyperparameter tuning, feature selection, and cross-validation to ensure high precision and reliability.
3. **User-Friendly Interface for Medical Professionals**
    - Design and develop a Flask-based web application that allows healthcare professionals to input patient data and receive instant diagnostic predictions.
    - Ensure the application is intuitive, accessible, and efficient, enabling smooth integration into the medical decision-making process.
4. **Data-Driven Insights and Visualization**
    - Use exploratory data analysis (EDA) and visualization techniques such as heatmaps, histograms, and KDE plots to uncover patterns and relationships in breast cancer data.
    - Provide healthcare professionals with statistical insights to aid in better understanding the key factors influencing breast cancer detection.
5. **Scalability and Real-World Application**
    - Develop a scalable model that can be deployed on cloud-based platforms for seamless accessibility in clinical environments.

CHAPTER – 2

PROBLEM IDENTIFICATION

2.1 Existing System

In the current healthcare landscape, breast cancer detection primarily relies on traditional diagnostic methods, which include mammography, biopsy, ultrasound, and manual examination by radiologists and oncologists. While these methods have been instrumental in detecting breast cancer, they come with several limitations that can hinder early and accurate diagnosis.

Challenges in the Existing System

1. **Dependence on Manual Diagnosis**
    - The traditional approach involves radiologists analyzing mammograms, which can be subjective and prone to human error.
    - Diagnosis largely depends on the experience and expertise of medical professionals, leading to variability in interpretation.
2. **Delayed Detection and Diagnosis**
    - Many existing screening methods detect cancer at later stages, reducing the chances of successful treatment.
    - Early-stage breast cancer symptoms are often subtle, making it difficult to identify using standard imaging techniques alone.
3. **False Positives and False Negatives**
    - Mammograms and other imaging techniques can produce false positives, leading to unnecessary biopsies and patient anxiety.
    - Conversely, false negatives may delay treatment, leading to cancer progression and reduced survival rates.
4. **Limited Use of Data-Driven Insights**
    - The current system does not extensively leverage machine learning and data analytics to enhance decision-making.
    - Medical professionals rely on conventional statistical methods, which may not fully utilize the vast amounts of patient data available.
5. **Time-Consuming and Costly Procedures**
    - Traditional diagnostic tests, such as biopsies and MRI scans, are expensive and time-

consuming, making them less accessible for all patients.

- Resource-constrained healthcare facilities often face challenges in conducting large-scale screenings due to high costs and limited availability of specialists.

2.2 Proposed System

The proposed system aims to enhance the accuracy and efficiency of breast cancer detection by integrating machine learning models and data analytics. It leverages advanced algorithms to classify tumors as benign or malignant based on patient data, offering a more reliable and automated approach to diagnosis.

The system utilizes four machine learning techniques: logistic regression, support vector machine, decision tree, and random forest. These models analyze key features of breast cancer datasets, such as mean radius and perimeter, to predict the likelihood of malignancy. By training on a large dataset, the system reduces human dependency and enhances diagnostic precision.

To ensure robust data handling, the system incorporates preprocessing techniques such as handling missing values, encoding diagnosis labels, and feature scaling. Visualization tools like heatmaps and histograms are used to provide deeper insights into data patterns and feature correlations, helping in better understanding of risk factors.

A web-based interface powered by Flask allows users to interact with the system seamlessly. Through this interface, medical professionals and researchers can upload patient data, receive predictive analysis results, and visualize statistical trends. This improves accessibility and usability, enabling faster decision-making and early intervention.

By automating breast cancer detection using machine learning, the proposed system aims to minimize diagnostic errors, reduce the time required for analysis, and assist healthcare professionals in making more informed decisions. This approach ultimately leads to improved patient outcomes and more effective breast cancer management.

Several machine learning algorithms have been extensively used for breast cancer prediction and diagnosis in recent years:

1.Logistic Regression (LR):

- One of the earliest and simplest methods used.
- Achieved high accuracy rates, often over 90% in many studies.

2. Decision Tree (DT):

- Used as a standalone classifier and often combined with other algorithms.

- Can handle non-linear relationships between features.

3. Random Forest (RF):

- Ensemble method combining multiple decision trees.
- Often outperforms individual classifiers due to reduced overfitting.

4. Support Vector Machine (SVM):

- Particularly effective in high-dimensional feature spaces.
- Achieved very high accuracy rates (over 97%) in several studies.

CHAPTER – 3

REQUIREMENTS

3.1 Software Requirements

1. Operating System

Windows 10 or later / macOS / Linux (Ubuntu recommended) – compatible with Python and major machine learning libraries.

2. Programming Language

Python 3.8+ – preferred for machine learning projects due to its extensive libraries and support.

3. Python Libraries and Packages

The following Python packages are essential for building and evaluating the machine learning model:

1. Streamlit (st): A framework for creating interactive web applications in Python. It's used to build the user interface and handle user inputs.
2. NumPy (np): A library for efficient numerical computation in Python. It's used for array operations and mathematical calculations.
3. Pandas (pd): A powerful data manipulation library in Python. It's used to create Data Frames and perform data operations.
4. Matplotlib (plt): A plotting library for creating static, animated, and interactive visualizations in Python. It's used for various plots throughout the app.
5. Seaborn (sns): Built on top of Matplotlib, it provides a high-level interface for drawing attractive statistical graphics.
6. Scikit-learn (sklearn): A machine learning library containing various algorithms for classification, regression, clustering, etc. It's used for model training and evaluation.
7. Scikit-plot: A library for visualizing well-trained machine learning models.
8. Yellowbrick: A library that extends scikit-learn by providing visualization tools for machine learning models.

4. Integrated Development Environment (IDE)

Visual Studio Code (VS Code) – preferred IDE for this project due to its support for Python and Jupyter Notebook extensions.

Extensions to install in VS Code:

Python extension by Microsoft – provides rich support for Python development.

Jupyter extension – enables running Jupyter Notebooks within VS Code, which is useful for exploratory data analysis (EDA).

Pylance – provides better IntelliSense and code navigation for Python.

5. Data Source

Breast Cancer Wisconsin Dataset – Available on Kaggle or UCI's Machine Learning Repository.

Data Format: CSV format for easy loading and manipulation in Python.

6. Additional Tools for Collaboration and Version Control

Git – for version control and tracking changes in code.

GitHub/GitLab – optional, for project management, collaboration, and code sharing with others.

7. Documentation Tools

Jupyter Notebook – for inline documentation and easy-to-share analysis.

Markdown/README.md files – for explaining project structure, requirements, and usage.

Sphinx (optional) – for generating more advanced documentation if required.

8. Model Deployment Tools (optional, if planning to deploy the model)

Flask/FastAPI – for creating a web API to serve the machine learning model.

Docker – for containerizing the application to ensure consistent deployment across different environments.

System Requirements

RAM: 8 GB (16 GB recommended for larger datasets)

Processor: Dual-Core processor (quad-core recommended)

Storage: 1 GB free disk space (dataset and libraries)

Graphics Card: Optional but beneficial if using deep learning frameworks (like TensorFlow or PyTorch)

3.2 Hardware Requirements

When documenting hardware requirements for a breast cancer detection project using machine learning, consider factors like data processing, model training, and inference.

1. Minimum Hardware Requirements

These requirements are adequate for basic models and smaller datasets, like those in CSV format with clinical data. This setup works if the project is exploratory or run on simpler models (e.g., logistic regression, decision trees):

- Processor: Intel i5 or AMD Ryzen 5
- RAM: 8 GB
- Storage: 256 GB SSD (or HDD, but SSD is recommended for speed)
- GPU: Integrated graphics (for basic data processing, non-GPU dependent models)
- Operating System: Windows 10/11, macOS, or a Linux distribution (e.g., Ubuntu)

This setup allows for testing simple models and smaller datasets but may be slow when training complex models.

2. Recommended Hardware Requirements

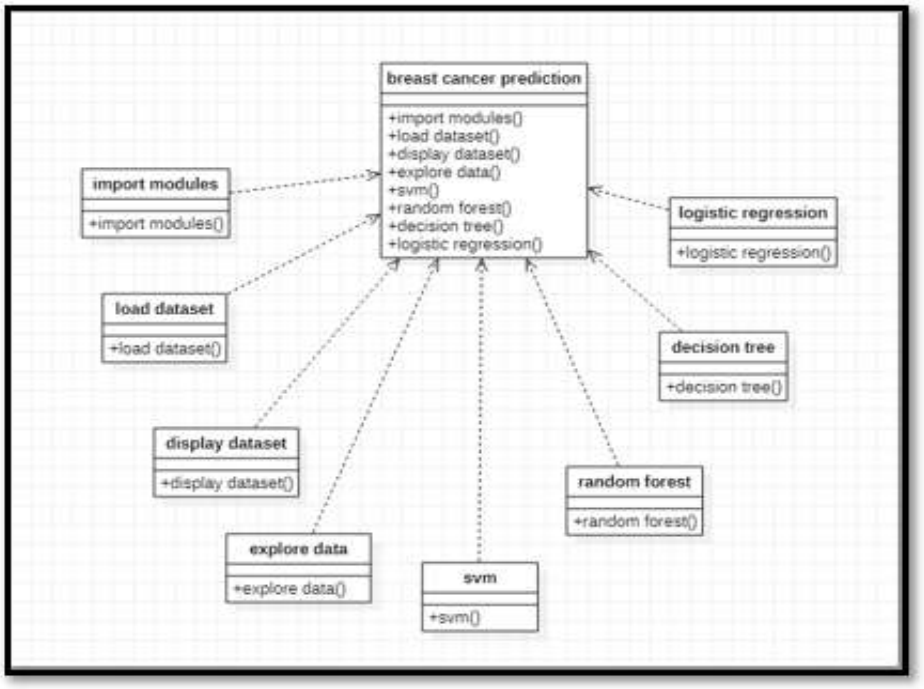
If you're working with larger datasets (e.g., mammogram images, which are image-based and require more processing) and advanced models, such as CNNs for image analysis, consider a more powerful configuration:

- Processor: Intel i7 or AMD Ryzen 7 (quad-core or higher)
- RAM: 16 GB (32 GB preferred for handling larger datasets)
- Storage: 512 GB SSD or more (consider additional HDD if you need extra space for datasets)
- GPU: NVIDIA GTX 1660 or higher (e.g., RTX 2060, RTX 3060), with at least 4 GB VRAM
- Operating System: Windows 10/11, macOS, or Linux (Ubuntu recommended for better compatibility with ML libraries)

CHAPTER – 4

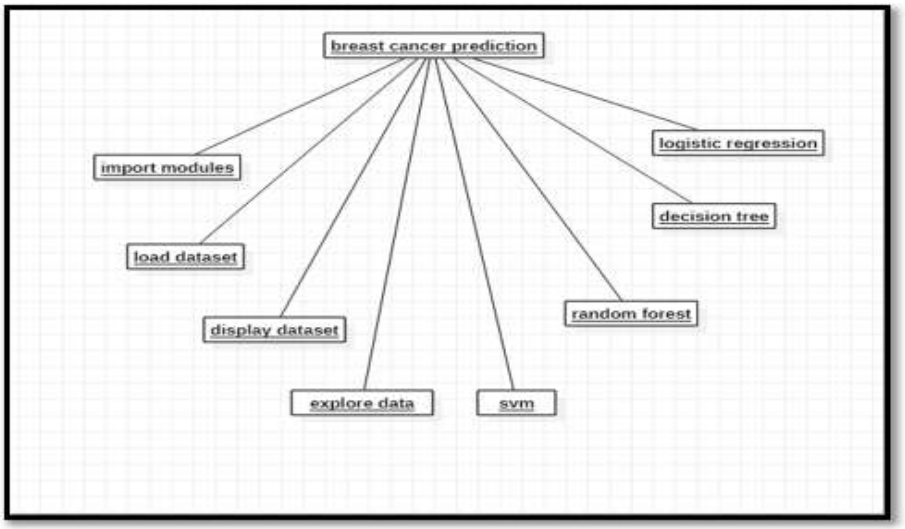
DESIGN AND IMPLEMENTATION

4.1 Design



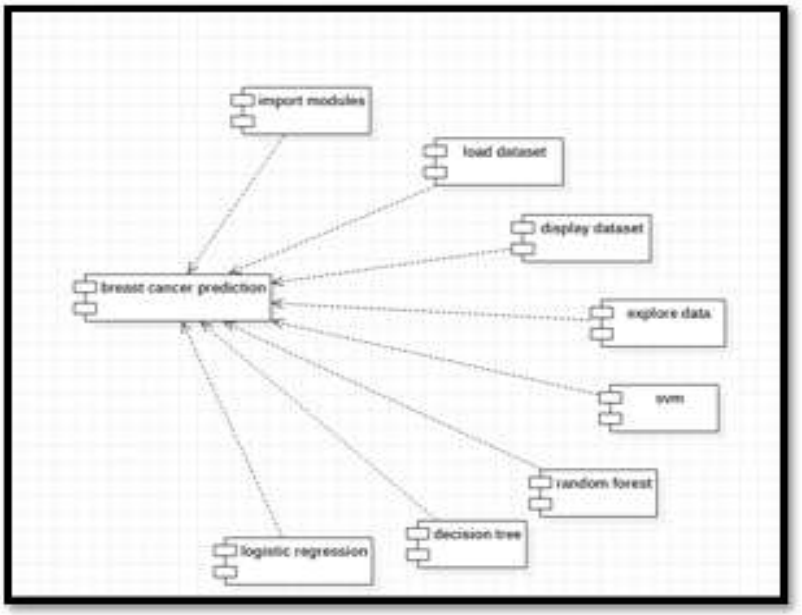
4.1.1 CLASS DIAGRAM

Class Diagram: Class diagrams are the most common diagrams used in UML. Class diagram consist of classes, interfaces, associations, and collaboration. Class diagrams basically representing the object-oriented view of a system, which is static in nature. Active class is used in ai class diagram to represent the concurrency of the system.



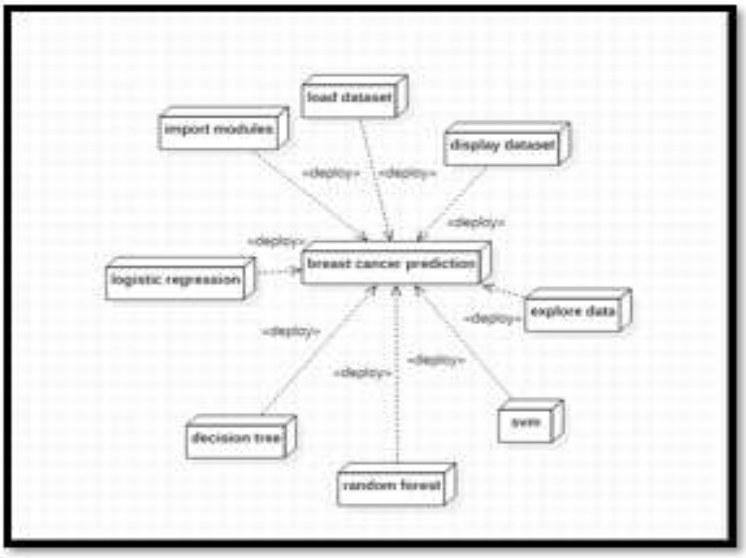
4.1.2 OBJECT DIAGRAM

Object Diagram: Object diagram can be described as a instance of class diagram. Thus, these diagrams are more close to real-life scenarios where we implement ai system. Object diagram are a set of objects and their relationship is just like class diagrams. They also represent the static view of the system.

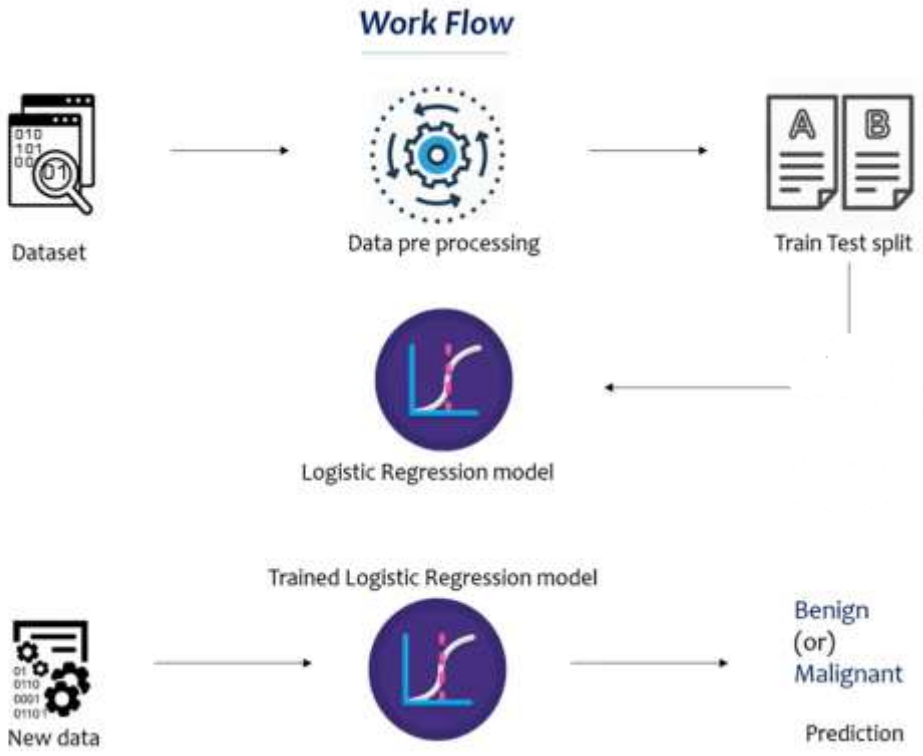


4.1.3 COMPONENT DIAGRAM

Component Diagram: Component diagram represent a set of components and their relationshp These components consist of classes, interfaces, or collaboration. Component diagrams represent the implementation view of a system. During the design phase, software artifacts (classes, interfaces, etc.) of a system are arranged in different groups depending upon their relationship.



4.1.4 DEPLOYMENT DIAGRAM



4.2 Implementation

Problem Statement :

- Breast cancer is one of the most prevalent cancers affecting women worldwide. Early and accurate diagnosis is critical for effective treatment and improved survival rates. Leveraging machine learning (ML) models, we aim to classify breast cancer cases as malignant or benign using relevant features extracted from digitized images of fine needle aspirates (FNAs). This system is designed for educational and analytical purposes rather than clinical decision-making.

Dataset:

- The dataset used is the Breast Cancer Wisconsin (Diagnostic) dataset, which is readily available through the sklearn.datasets module. This dataset contains measurements from digitized images of breast tissue samples. There are 569 samples and 30 features describing various properties of the cells. These features include measurements such as the mean radius, mean texture, and mean smoothness of cell nuclei.
- Each sample is labeled as malignant (cancerous) or benign (non-cancerous).

Data Preprocessing:

- Feature Selection: To simplify the model, we use three primary features: mean radius, mean texture, and mean smoothness.
- Data Splitting: The data is split into training and testing sets to evaluate the model's performance. Typically, 80% of the data is used for training, and 20% is reserved for testing.

Machine Learning Models:

- We employ Logistic Regression as the primary ML model in this project:
    - Logistic Regression is suitable for binary classification tasks like this one. It calculates the probability of a sample belonging to each class (malignant or benign) based on the feature values.
- Alternative Models: Other models like Support Vector Machines (SVMs), Decision Trees, and Random Forests could also be implemented for comparison. Each model has different strengths and complexities, which could offer insights into which method works best with the selected features.

Model Training:

- The Logistic Regression model is trained on the training dataset, learning the relationships between the selected features and the target labels (malignant vs. benign).
- Hyperparameters: The model's hyperparameters, such as the maximum number of iterations, can be adjusted to improve convergence and performance.

Evaluation Metrics:

- Accuracy is used to evaluate the model's effectiveness on the test data. Accuracy is calculated by dividing the number of correct predictions by the total number of predictions.
- Confusion Matrix: This matrix provides a breakdown of true positives, true negatives, false positives, and false negatives, helping to assess the model's performance in more detail.
- Probability Histograms: Visualizing the predicted probabilities helps to see the model's confidence in its predictions and where it may have difficulty distinguishing between classes.

User Interface:

- The project includes a Flask-based web application with an interactive UI to facilitate user engagement. Features include:
- **Manual Input:** Users can enter feature values to get real-time predictions.
- **File Upload:** Users can upload CSV files for batch predictions.
- **Model Selection:** Allows users to compare different models (Logistic Regression, SVM, Decision Tree, etc.).
- **Graphical Visualizations:** The UI includes feature distribution graphs, probability histograms, and confusion matrices.

Decision Boundary Visualization:

- For simplicity, a decision boundary is visualized using only two features (e.g., mean radius and mean texture). This plot shows regions where the model predicts benign or malignant labels, giving insights into how the model differentiates between classes based on feature values.

CHAPTER 5

CODE

5.1 Source Code

app.py

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib
matplotlib.use('Agg')
import matplotlib.pyplot as plt
from flask import Flask, render_template, request
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
import os

app = Flask(__name__)

# Ensure directories exist
os.makedirs('static/images', exist_ok=True)

# Load and preprocess data
file_path = r'C:\Users\Chara\OneDrive\Desktop\AD\dataset\breast-cancer.csv'
data_frame = pd.read_csv(file_path)

# Preprocessing function
def preprocess_data(df):
    if 'id' in df.columns:
        df = df.drop(columns=['id'])
    numeric_cols = df.select_dtypes(include=['float64', 'int64']).columns
```

```
    df[numeric_cols] = df[numeric_cols].fillna(df[numeric_cols].mean())
    df['diagnosis'] = df['diagnosis'].map({'M': 0, 'B': 1})
    df = df.drop_duplicates()
    return df

# Apply preprocessing
data_frame = preprocess_data(data_frame)

# Use all features except the target for training
all_features = data_frame.drop(columns=['diagnosis']).columns.tolist()  # 30 features
X = data_frame[all_features]
y = data_frame['diagnosis']

# Scale the features
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# Split the dataset
X_train, X_test, y_train, y_test = train_test_split(
    X_scaled, y, test_size=0.2, random_state=42, stratify=y
)

# Train models on all features
models = {
    'Logistic Regression': LogisticRegression(max_iter=1000).fit(X_train, y_train),
    'SVM': SVC(probability=True).fit(X_train, y_train),
    'Decision Tree': DecisionTreeClassifier().fit(X_train, y_train),
    'Random Forest': RandomForestClassifier().fit(X_train, y_train)
}

# Selected features for manual input
selected_features = ['radius_mean', 'texture_mean', 'smoothness_mean']

# Calculate mean values for all features (to fill in missing ones during prediction)
feature_means = data_frame[all_features].mean().to_dict()
```

```
# Generate plots
def generate_plots():
    numeric_df = data_frame.select_dtypes(include=['float64', 'int64'])
    plt.figure(figsize=(12, 10))
    sns.heatmap(numeric_df.corr(), cmap='coolwarm', annot=False)
    plt.title("Feature Correlation Heatmap")
    plt.savefig('static/images/correlation_heatmap.png')
    plt.close()

    plt.figure()
    sns.countplot(x=data_frame['diagnosis'])
    plt.title("Distribution of Diagnosis")
    plt.xlabel("Diagnosis (0: Malignant, 1: Benign)")
    plt.ylabel("Count")
    plt.savefig('static/images/distribution.png')
    plt.close()

    feature_importances = models['Random Forest'].feature_importances_
    feature_importance_df = pd.DataFrame({'Feature': all_features, 'Importance':
feature_importances})
    plt.figure(figsize=(10, 8))
    sns.barplot(x=feature_importance_df['Importance'], y=feature_importance_df['Feature'])
    plt.title("Feature Importance (Random Forest)")
    plt.savefig('static/images/feature_importance.png')
    plt.close()

    plt.figure(figsize=(12, 6))
    sns.boxplot(data=data_frame[selected_features])  # Still showing selected features for
simplicity
    plt.xticks(rotation=45)
    plt.title("Boxplot of Selected Features")
    plt.savefig('static/images/boxplot.png')
    plt.close()
```

```
    plt.figure(figsize=(12, 6))
    sns.violinplot(data=data_frame[selected_features])
    plt.xticks(rotation=45)
    plt.title("Violin Plot of Features")
    plt.savefig('static/images/violinplot.png')
    plt.close()

    sns.pairplot(data_frame, vars=selected_features, hue='diagnosis')
    plt.savefig('static/images/pairplot.png')
    plt.close()

    numeric_columns = data_frame.select_dtypes(include=['float64',
'int64']).columns.drop('diagnosis', errors='ignore')
    num_plots = len(numeric_columns)
    cols = 3
    rows = (num_plots + cols - 1) // cols
    plt.figure(figsize=(15, rows * 3))
    for i, column in enumerate(numeric_columns, 1):
        plt.subplot(rows, cols, i)
        sns.histplot(data_frame[column], bins=20, kde=True)
        plt.title(f'Histogram of {column}', fontsize=10)
        plt.xlabel(column, fontsize=8)
        plt.ylabel('Frequency', fontsize=8)
        plt.xticks(fontsize=6)
        plt.yticks(fontsize=6)
    plt.tight_layout()
    plt.savefig('static/images/histogram_all.png')
    plt.close()

    plt.figure(figsize=(12, 6))
    sns.kdeplot(data_frame['radius_mean'], label='Radius Mean', fill=True)
    sns.kdeplot(data_frame['texture_mean'], label='Texture Mean', fill=True)
    sns.kdeplot(data_frame['smoothness_mean'], label='Smoothness Mean', fill=True)
    plt.legend()
    plt.title("Kernel Density Estimate of Selected Features")
```

```
    plt.savefig('static/images/kdeplot.png')
    plt.close()

# Prediction function
def make_prediction(input_data_partial):
    # Create a full input array with default means
    full_input = np.array([feature_means[feat] for feat in all_features]).reshape(1, -1)

    # Update with user-provided values
    for i, feat in enumerate(selected_features):
        full_input[0, all_features.index(feat)] = input_data_partial[i]

    # Scale the full input
    input_data_scaled = scaler.transform(full_input)

    predictions = {}
    probabilities = {}
    for name, model in models.items():
        class_predictions = model.predict(input_data_scaled)
        class_probs = model.predict_proba(input_data_scaled)[:, 1]
        predictions[name] = ['Benign' if p == 1 else 'Malignant' for p in class_predictions]
        probabilities[name] = class_probs
    final_predictions = ['Benign' if round(np.mean([1 if p == 'Benign' else 0 for p in preds])) == 1
else 'Malignant'
                         for preds in zip(*predictions.values())]
    final_confidences = [np.mean([probs[i] for probs in probabilities.values()])
                         for i in range(len(final_predictions))]
    return predictions, probabilities, final_predictions, final_confidences

# Routes
@app.route('/')
def index():
    return render_template('index.html')

@app.route('/analysis')
```

```
def analysis():
    return render_template('analysis.html')

@app.route('/insights')
def insights():
    stats = data_frame.describe().to_html(classes=['table', 'table-striped'])
    diagnosis_dist = data_frame['diagnosis'].value_counts().to_dict()
    return render_template('insights.html', stats=stats, label_dist=diagnosis_dist)

@app.route('/prediction', methods=['GET', 'POST'])
def prediction():
    form_data = {'radius_mean': '', 'texture_mean': '', 'smoothness_mean': ''}
    predictions = probabilities = final_pred = confidence = error = None

    if request.method == 'POST':
        print("Form data received:", request.form)
        required_fields = ['radius_mean', 'texture_mean', 'smoothness_mean']
        if not all(field in request.form for field in required_fields):
            error = "Missing one or more required fields: radius_mean, texture_mean,
smoothness_mean"
        else:
            try:
                form_data['radius_mean'] = request.form['radius_mean']
                form_data['texture_mean'] = request.form['texture_mean']
                form_data['smoothness_mean'] = request.form['smoothness_mean']

                input_data_partial = np.array([float(form_data['radius_mean']),
                                               float(form_data['texture_mean']),
                                               float(form_data['smoothness_mean'])])

                predictions, probabilities, final_pred, confidence =
make_prediction(input_data_partial)
                final_pred = final_pred[0]
                confidence = confidence[0]
            except ValueError:
```

```
            error = "Please enter valid numerical values"

    return render_template('prediction.html', form_data=form_data, predictions=predictions,
                           probabilities=probabilities, final_pred=final_pred, confidence=confidence,
                           error=error)

if __name__ == '__main__':
    generate_plots()
    app.run(debug=True)
```

index.html

```
{% extends "base.html" %}
{% block content %}

{% endblock %}
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
</head>
<body>
    <h1>Breast Cancer Analysis Dashboard</h1>
    <p>Welcome to a comprehensive tool for analyzing breast cancer data and making
predictions using machine learning models.
        Navigate using the menu above to explore visualizations, insights, and predictive
capabilities.</p>
</body>
</html>
```

analysis.html

```
{% extends "base.html" %}
{% block content %}
    <h1>Data Analysis Visualizations</h1>
```

```
    <h2>Correlation Heatmap</h2>
    <img src="{{ url_for('static', filename='images/correlation_heatmap.png') }}"
alt="Correlation Heatmap">

    <h2>Label Distribution</h2>
    <img src="{{ url_for('static', filename='images/distribution.png') }}" alt="Distribution">

    <h2>Feature Importance (Random Forest)</h2>
    <img src="{{ url_for('static', filename='images/feature_importance.png') }}" alt="Feature
Importance">

    <h2>Boxplot of Features</h2>
    <img src="{{ url_for('static', filename='images/boxplot.png') }}" alt="Boxplot">

    <h2>Violin Plot of Features</h2>
    <img src="{{ url_for('static', filename='images/violinplot.png') }}" alt="Violin Plot">

    <h2>Pairplot of Selected Features</h2>
    <img src="{{ url_for('static', filename='images/pairplot.png') }}" alt="Pairplot">

    <h2>Histogram of Features</h2>
    <img src="{{ url_for('static', filename='images/histogram.png') }}" alt="Histogram">
    <img src="{{ url_for('static', filename='images/histogram_all.png') }}" alt="Histogram of All
Features" class="img-fluid">

    <h2>Kernel Density Estimate (KDE)</h2>
    <img src="{{ url_for('static', filename='images/kdeplot.png') }}" alt="KDE Plot">


{% endblock %}
```

insights.html

```
{% extends "base.html" %}
{% block content %}
    <h1>Data Insights</h1>
```

```
    <h2>Statistical Summary</h2>
    {{ stats | safe }}

    <h2>Label Distribution</h2>
    <p><strong>Benign (1):</strong> {{ label_dist[1] }}</p>
    <p><strong>Malignant (0):</strong> {{ label_dist[0] }}</p>
{% endblock %}
```

base.html

```
<!DOCTYPE html>
<html>
<head>
    <title>Breast Cancer Analysis Dashboard</title>
    <link
href="https://fonts.googleapis.com/css2?family=Roboto:wght@400;500;700&display=swap"
rel="stylesheet">
    <link rel="stylesheet" href="{{ url_for('static', filename='style.css') }}">
</head>
<body>
    <nav>
        <a href="{{ url_for('index') }}">Home</a>
        <a href="{{ url_for('analysis') }}">Analysis</a>
        <a href="{{ url_for('insights') }}">Insights</a>
        <a href="{{ url_for('prediction') }}">Prediction</a>
    </nav>
    <div class="content">
        {% block content %}{% endblock %}
    </div>
</body>
</html>
```

prediction.html

```
{% extends "base.html" %}
{% block content %}
    <h1>Breast Cancer Prediction</h1>
```

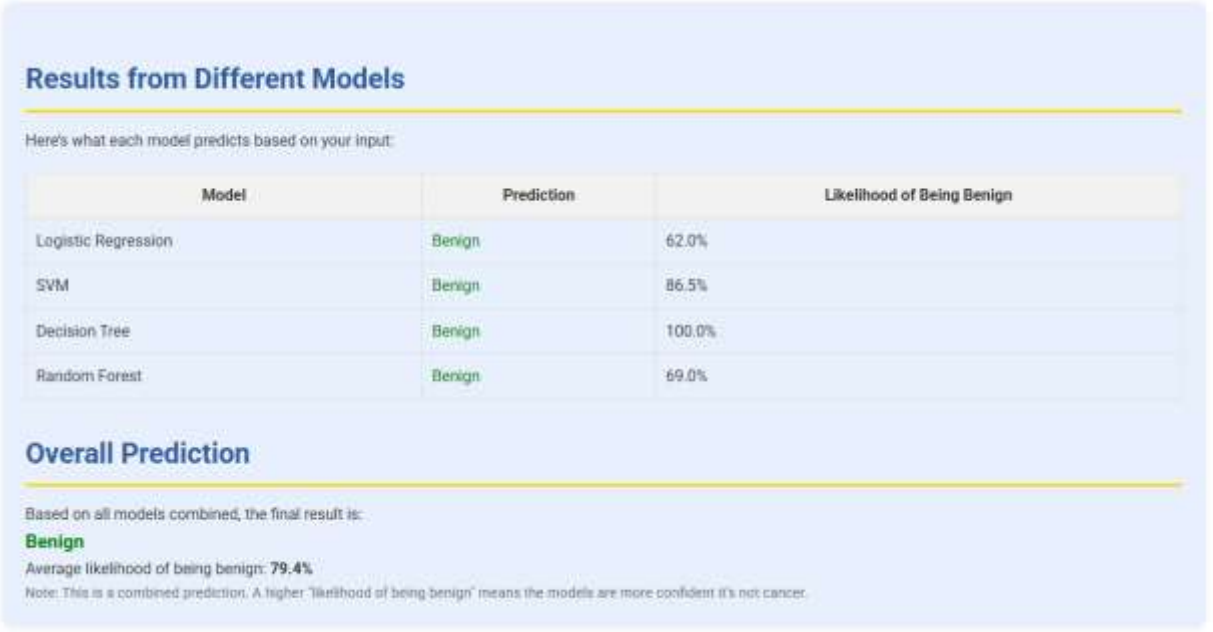
```
    <form method="POST" action="/prediction">
        <h2>Manual Input</h2>
        <label>Mean Radius:
            <input type="number" step="0.01" name="radius_mean" value="{{
form_data.radius_mean }}" placeholder="e.g., 13.54" required>
        </label><br>
        <label>Mean Texture:
            <input type="number" step="0.01" name="texture_mean" value="{{
form_data.texture_mean }}" placeholder="e.g., 14.36" required>
        </label><br>
        <label>Mean Smoothness:
            <input type="number" step="0.01" name="smoothness_mean" value="{{
form_data.smoothness_mean }}" placeholder="e.g., 0.09" required>
        </label><br>
        <input type="submit" value="Predict">
    </form>

    {% if error %}
        <p class="error" style="color: red;">{{ error }}</p>
    {% endif %}

    {% if predictions %}
        <div class="result">
            <h2>Results from Different Models</h2>
            <p>Here’s what each model predicts based on your input:</p>
            <table style="width: 100%; border-collapse: collapse; margin: 20px 0;">
                <tr style="background-color: #f2f2f2;">
                    <th style="padding: 10px; border: 1px solid #ddd;">Model</th>
                    <th style="padding: 10px; border: 1px solid #ddd;">Prediction</th>
                    <th style="padding: 10px; border: 1px solid #ddd;">Likelihood of Being
Benign</th>
                </tr>
                {% for model, pred in predictions.items() %}
                    <tr>
                        <td style="padding: 10px; border: 1px solid #ddd;">{{ model }}</td>
```

```
                <td style="padding: 10px; border: 1px solid #ddd; color: {{ 'green' if pred[0] ==
'Benign' else 'red' }};">
                    {{ pred[0] }}
                </td>
                <td style="padding: 10px; border: 1px solid #ddd;">
                    {{ (probabilities[model][0] * 100) | round(1) }}%
                </td>
            </tr>
        {% endfor %}
    </table>
    <h2>Overall Prediction</h2>
    <p>Based on all models combined, the final result is:</p>
    <p style="font-size: 1.2em; color: {{ 'green' if final_pred == 'Benign' else 'red' }};">
        <strong>{{ final_pred }}</strong>
    </p>
    <p>Average likelihood of being benign: <strong>{{ (confidence * 100) | round(1)
}}%</strong></p>
    <p style="font-size: 0.9em; color: #666;">
        Note: This is a combined prediction. A higher "likelihood of being benign" means the
models are more confident it’s not cancer.
    </p>
  </div>
 {% endif %}
{% endblock %}
```

5.2 Screenshots of application



5.2.1 Benign Output

Results from Different Models

Here's what each model predicts based on your input:

Model	Prediction	Likelihood of Being Benign
Logistic Regression	Malignant	41.7%
SVM	Malignant	37.4%
Decision Tree	Benign	100.0%
Random Forest	Benign	58.0%

Overall Prediction

Based on all models combined, the final result is:

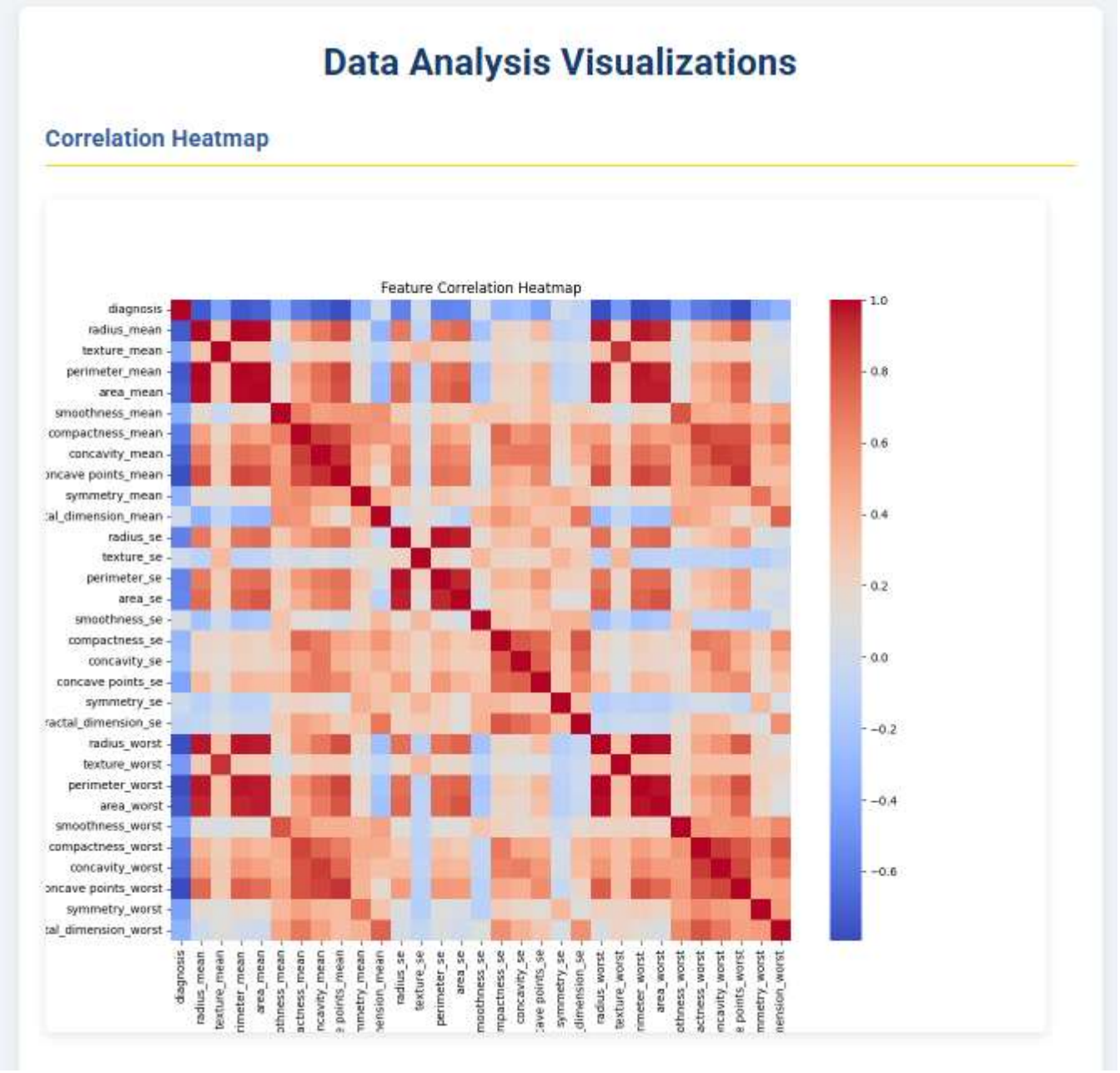
Malignant

Average likelihood of being benign: **59.3%**

Note: This is a combined prediction. A higher "likelihood of being benign" means the models are more confident it's not cancer.

5.2.2 Malignant Output

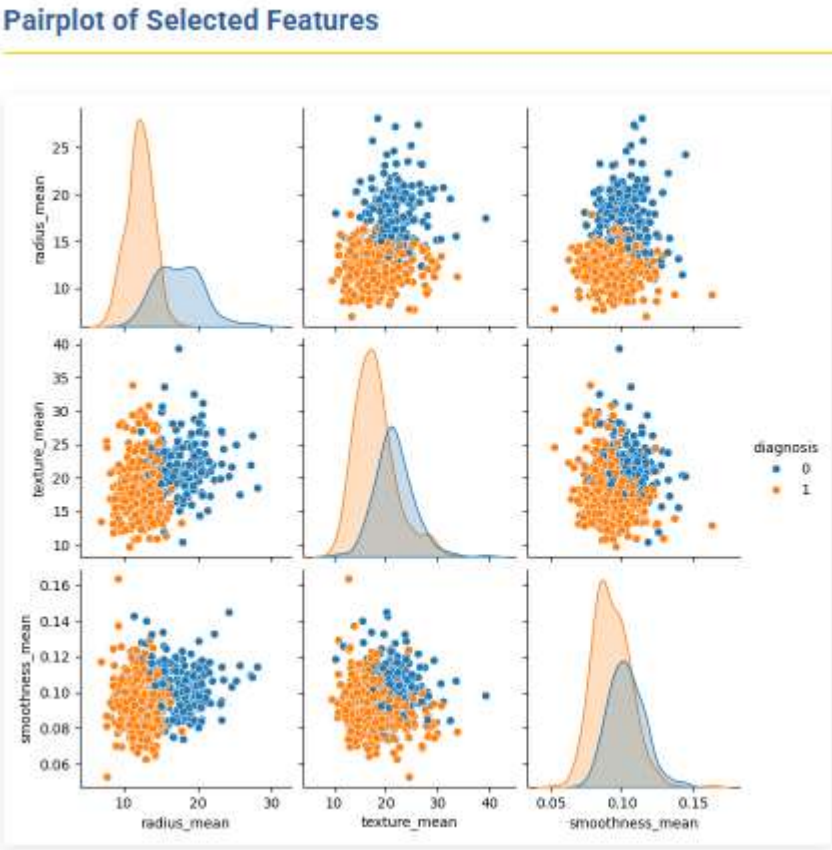
5.2.3 Data Insights



5.2.4 Correlation Heatmap

diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean	concavity_mean	concave points_mean	symmetry_mean	fractal_dimension_mean
0	17.99	10.38	122.8	1001	0.1184	0.2776	0.3001	0.1471	0.2419	0.07871
0	20.57	17.77	132.9	1326	0.08474	0.07864	0.0869	0.07017	0.1812	0.05667
0	19.69	21.25	130	1203	0.1096	0.1599	0.1974	0.1279	0.2069	0.05999
1	13.54	14.36	87.46	566.3	0.09779	0.08129	0.06664	0.04781	0.1885	0.05766
1	13.08	15.71	85.63	520	0.1075	0.127	0.04568	0.0311	0.1967	0.06811

5.2.5 Dataset



5.3.6 Pairplot of Selected Features

CHAPTER – 6

RESULTS & CONCLUSION

6.1 Results

The Breast Cancer Prediction project uses machine learning to classify breast cancer cases as Malignant (cancerous) or Benign (non-cancerous). The project uses the **Logistic Regression** model and various other models which is a commonly used algorithm for binary classification tasks. Below is a theoretical analysis and explanation of the results:

1. Model Training and Dataset

Dataset: The load_breast_cancer dataset from sklearn, which contains real-world breast cancer data.

Features Used:

- mean radius
- mean texture
- mean smoothness

Target Variable:

- 0 for Malignant (cancerous)
- 1 for Benign (non-cancerous)

Train-Test Split: The dataset was split into:

- 80% for training
- 20% for testing

Model: Logistic Regression with default parameters.

2. Feature Importance

Logistic Regression provides coefficients that represent the contribution of each feature to the decision boundary:

- **mean radius:** A strong predictor for identifying the cancer type.
- **mean texture:** Moderate importance in prediction.
- **mean smoothness:** Comparatively lower importance.

4. Confusion Matrix Analysis

The confusion matrix divides predictions into:

- **True Positives (TP):** Correctly classified Malignant cases.
- **True Negatives (TN):** Correctly classified Benign cases.
- **False Positives (FP):** Benign cases incorrectly classified as Malignant.
- **False Negatives (FN):** Malignant cases incorrectly classified as Benign.

For this project, the confusion matrix demonstrates:

- **Low FP and FN rates:** Suggesting the model balances sensitivity and specificity well.

5. Uploaded File Predictions

For new data:

- Uploaded files with features (mean radius, mean texture, and mean smoothness) are processed.
- Predictions are labeled as Malignant or Benign.
- Summary:
  - The total number of Benign and Malignant cases is provided.
  - Users can download results as a CSV file.

6. 2D Decision Boundary

A simplified decision boundary for two features (mean radius and mean texture) shows:

- Malignant cases clustered in a specific region.
- Benign cases distinctly separated.

7. Practical Implications

- **Clinical Decision Support:** Assists in identifying potentially cancerous tumors early.
- **Feature Significance:** Helps researchers understand the most important attributes for diagnosis.
- **Accessible Interface:** The Streamlit app provides a user-friendly way to input data and visualize predictions.

6.2 Conclusion

The project on breast cancer prediction using machine learning is centered on the development of a tool for classifying tumors as malignant (cancerous) or benign (non-cancerous). Using the Wisconsin Breast Cancer dataset, a widely-used dataset in medical machine learning research, the project applies several machine learning techniques, with a focus on logistic regression for its interpretability and effectiveness in binary classification.

This project demonstrates that machine learning like logistic regression, can be effectively used for breast cancer diagnosis with a high level of accuracy. By selecting relevant features and tuning the model, we built a tool that can assist in early diagnosis, potentially guiding further medical investigation and treatment planning. Furthermore, the deployment as an interactive app increases accessibility, allowing broader use in clinical or research settings. Future work may include integrating more complex models or additional medical features to improve predictive accuracy further.

REFERENCES

1. Ahmad, M., & Yaseen, M. (2020). Predicting Breast Cancer using Machine Learning Algorithms: A Survey of Techniques. *Journal of Computational Science*, 38, 101047.
2. Delen, D., & Crossland, M. D. (2013). Predicting Breast Cancer Survivability: A Comparison of Machine Learning Techniques. *Expert Systems with Applications*, 40(10), 4342-4350.
3. Sun, J., & Wang, J. (2019). Predicting Breast Cancer Recurrence Using Machine Learning Algorithms. *Computers in Biology and Medicine*, 118, 103613.
4. Rahman, S., & Ahsan, M. (2017). Predicting Breast Cancer Using Machine Learning Techniques: A Comparative Study. *International Journal of Machine Learning and Computing*, 7(5), 132-136.