Project Report: Breast Cancer Classification Using Machine Learning

Project Overview

This project aims to build a robust machine learning model to classify breast cancer tumors as benign or malignant based on diagnostic features. The workflow includes data preprocessing, exploratory analysis, model training and tuning, and deployment via a Flask web application.

Dataset Summary

- **Source**: Breast cancer diagnostic dataset (e.g., Wisconsin Diagnostic Breast Cancer)
- **Features**: 30 numerical attributes derived from digitized images of fine needle aspirate (FNA) of breast mass
- **Target**: Diagnosis (M = Malignant, B = Benign)

Data Science Workflow

1. Data Cleaning

- Removed missing values and duplicates
- Dropped irrelevant columns (e.g., id)
- Verified class balance and feature integrity

2. Feature Engineering

- Encoded target labels using LabelEncoder
- Scaled features using StandardScaler to normalize input for model training

3. Exploratory Data Analysis (EDA)

- Visualized class distribution using seaborn
- Generated correlation heatmaps to identify feature relationships
- Assessed feature importance post-modeling

4. Model Building

- Split data into training and test sets (80/20)
- Trained a RandomForestClassifier using GridSearchCV for hyperparameter tuning
- Parameters tuned:

- n_estimators: [50, 100, 200]
- max_depth: [None, 10, 20]

5. Model Evaluation

- **Test Accuracy**: 96.5%
- **Classification Report**:
  - Precision: 0.96 (Benign), 0.98 (Malignant)
  - Recall: 0.99 (Benign), 0.93 (Malignant)
  - F1-Score: 0.97 (Benign), 0.95 (Malignant)
- **Confusion Matrix**:

  ```
  [[70  1]
   [ 3 40]]
  ```

- **Cross-Validation Scores**: [0.989, 0.945, 0.978, 0.956, 0.945]
- **Mean CV Score**: 96.3%

6. Model Saving

- Saved the tuned model using joblib:

  ```
  joblib.dump(grid.best_estimator_, 'breast_cancer_model.pkl')
  ```

Deployment Workflow (Flask + Render)

Flask Setup

- Created app.py to serve predictions
- Built HTML form in templates/index.html for user input
- Loaded model and scaler for real-time prediction

Render Deployment

- Pushed project to GitHub
- Created requirements.txt
- Deployed via Render's web service with dynamic port binding

Outcome

- Successfully deployed a high-performing breast cancer classifier
- Achieved strong generalization and interpretability
- Enabled real-time predictions via a user-friendly web interface