

Breast Prediction ML model

Venkata Krishna Uppuluri
Electrical and Computer Engineering
Carleton University
Ottawa, Canada

Sai Deepak Devineni
Electrical and Computer Engineering
Carleton University
Ottawa, Canada

Abstract- Breast cancer is a pervasive and often life-altering disease that affects millions of individuals worldwide. Machine learning models can be useful in predicting the likelihood of breast cancer whether it is malignant or benign, allowing for early intervention and treatment. In this project, we will build a machine learning model using a dataset of patient information, including demographics, medical history, and laboratory test results. We will compare the performance of several classification algorithms, including logistic regression, linear logistic regression, random forest, support vector machines, decision trees, and naive Bayes, and evaluate their accuracy, precision, recall, and F1-score using a confusion matrix. The best-performing model will be used to create an interactive web application that allows users to input their own information and receive a prediction of breast cancer whether it is benign or malignant with a probability from 0 to 1. This project aims to provide a practical tool for healthcare professionals and individuals to make informed decisions about their health in predicting breast cancer.

Key Terms: Machine Learning, Breast Cancer, Benign, Malignant, Python, Node JS, React, Web application, Flask API, Heroku

1. Introduction

1.1 Problem Definition

Breast cancer remains one of the most common and deadly forms of cancer affecting women worldwide. Early detection is crucial for effective treatment and increases the chances of survival significantly. However, traditional diagnostic methods such as mammography, while effective, can sometimes be limited in their accuracy and may also be inaccessible in resource-limited settings. Furthermore, these methods can be time-consuming and costly, and they often require specialized medical expertise for interpretation. There is an urgent need for more efficient, accurate, and accessible diagnostic tools to improve the early detection of breast cancer. This is particularly critical for distinguishing between malignant (cancerous) and benign (non-cancerous) tumors, as this classification greatly influences treatment decisions and patient outcomes. The development of a machine learning (ML) model that can accurately and promptly classify breast cancer tumors as malignant or benign could revolutionize breast cancer diagnostics. Such a model can leverage existing

medical data to provide rapid, cost-effective, and potentially more accurate assessments than traditional methods. This innovation could be particularly transformative in under-resourced healthcare settings, where access to advanced diagnostic tools and specialized medical expertise may be limited.

The overarching goal of this project is to develop and validate a machine learning model capable of effectively distinguishing between malignant and benign breast tumors. This model aims to supplement existing diagnostic methods, thereby enhancing the accuracy of breast cancer diagnoses and improving the treatment and survival prospects for patients globally.

1.2 Background

Relevant Technologies

- Python
- Sklearn Library
- HTML, CSS and JavaScript
- React
- Flask API
- Heroku
- Machine Learning
- Data Preparation Techniques and Visualisation

Tools and Requirements

- Visual Studio Code
- Medical data
- Node JS

1.3 Project Guidance

For help with the technical specifications and advice from a professional, I'd like to enlist the help of Prof. Ali Hassan Abbas in the Robotic Surgery course. Prof. Abbas is a researcher and entrepreneur; he is the co-founder and director of KI design. He has in-depth knowledge of data science, artificial intelligence, and machine learning which is applied to the medical industry and many others, which will help us gain in-depth knowledge of the subject. To effectively complete the project, we would like to collaborate, split the roles, and then integrate the whole. We will also work with members from different teams who are working on similar tasks and coordinate with them to finally integrate the tasks with the E-hospital page.

1.4 Project Context

- We have used the publicly available dataset Breast Cancer Wisconsin and have downloaded it from UCI Machine Learning Repository.
- Once our tasks are completed and tested, we have worked with an integration team to integrate with the E-hospital platform.

2. Design Overview

2.1 Requirements

- Identification of required datasets
- Data Cleaning and modeling
- Data Analysis and Exploration
- Training ML model
- Evaluating and validating ML model
- Deployment of dataset
- Integration of the model with E-hospital

2.2 Detailed Design

The objective of our project is to develop an ML model that can accurately predict the presence of breast cancer in patients based on the inputs given by the patients. Data was collected from various sources including hospitals, medical research institutions, and public databases. The dataset will consist of clinical and demographic features of patients and more about medical attributes like radius mean, texture mean, perimeter mean, area mean etc. The collected data has been preprocessed by removing missing values, handling outliers, and normalizing the data to ensure that the ML model can perform optimally. Different ML models were evaluated, including logistic regression, decision trees, random forests, and support vector machines (SVMs). The performance of each model will be evaluated based on accuracy, precision, recall, F1 score, AUC, etc. The performance of the final ML model was evaluated on an independent test set. The data set was deployed in Heroku for a wide-access range and then the ML model was integrated with E-hospital platform using flask-API.

2.3 Implementation

Building a ML model for Breast Cancer Prediction contains the following steps:

2.3.1 Data Cleaning and Preprocessing

Data Cleaning helps to explore the data and remove incomplete or incorrect values, handle missing data, outlier detection. Data preprocessing to transform the input data into desired format and to train the ML model.

Task 1 (Breast Cancer Prediction)

Data Cleaning

The dataset does not contain null values or duplicates. Also, the data was properly balanced for the presence and absence of breast cancer prediction disease. Statistical distribution of data

was analysed, and the following are the observations.

```
# calculate duplicates
dups = x.duplicated()
# report if there are any duplicates
print(dups.any())
# list all duplicate rows
print(x[dups])
```

Fig 1. Identify rows that contain duplicate records

2.3.2 Data Preprocessing

- Ordinal encoding is a method that converts categorical data into numerical data by assigning a numerical value to each category based on its order or rank. The numerical values assigned are usually integers ranging from 1 to the number of categories in the feature.
- Data Normalization technique MinMax Scaler was used to scale numerical data within a specified range to improve the performance of machine learning algorithm such as SVC, KNN as they are distance-based algorithms.
- Standard Scalar preprocessing technique was used to scale each feature to have a mean 0 and standard deviation 1 and to compare the values with different units or scales.

```
#Data Encoding by Ordinal Encoder for categorical data
from sklearn.preprocessing import OrdinalEncoder
oe=OrdinalEncoder()
features[['Sex','ChestPainType','RestingECG','ExerciseAngina','ST_Slope']] = oe.fit_transform(features[['Sex','ChestPainType','RestingECG','ExerciseAngina','ST_Slope']])

[ ] from sklearn.preprocessing import MinMaxScaler, StandardScaler
normalize = MinMaxScaler()
standard = StandardScaler()
features['tLopok'] = normalize.fit_transform(features[['tLopok']])
features['Age'] = standard.fit_transform(features[['Age']])
features['RestingBP'] = standard.fit_transform(features[['RestingBP']])
features['Cholesterol'] = standard.fit_transform(features[['Cholesterol']])
features['MaxHR'] = standard.fit_transform(features[['MaxHR']])
features.head()
```

Fig 2. Data Preprocessing

Select KBest from sklearn reduces the dimensionality of the data and removes redundant features. The relationship between each feature and the target variable and the K-best features are selected based on the scoring function.

```
from sklearn.feature_selection import SelectKBest
from sklearn.feature_selection import chi2
#Apply SelectKBest and extract top 10 features
best=SelectKBest(score_func=chi2, k=10)
fit=best.fit(X,y)
data_scores=pd.DataFrame(fit.scores_)
data_columns=pd.DataFrame(X.columns)

#Join the two dataframes
scores=pd.concat([data_columns,data_scores],axis=1)
scores.columns=['Feature','Score']
print(scores.nlargest(11,'Score'))
```

Fig 3. K-best features selection

2.3.3 Data Analysis and Exploration

It is the process of figuring out what the data can tell us, and we use Exploratory Data Analysis (EDA) to find patterns, relationships, or anomalies to inform our subsequent analysis. We will look for patterns, differences, and other features that address the questions we are interested in also will try to uncover the relationships between the different variables.

```
ax = sns.countplot(y,label="Count")          # M = 212, B = 357
B, M = y.value_counts()
print('Number of Benign: ',B)
print('Number of Malignant : ',M)
ax.set_ylabel('Number of patients')
bars = ax.patches
half = int(len(bars)/2)
left_bars = bars[:half]
right_bars = bars[half:]
for left, right in zip(left_bars, right_bars):
    height_l = left.get_height()
    height_r = right.get_height()
    total = height_l + height_r
    ax.text(left.get_x() + left.get_width()/2., height_l + 40,
            '{0:.0%}'.format(height_l/total), ha="center")
    ax.text(right.get_x() + right.get_width()/2., height_r + 40,
            '{0:.0%}'.format(height_r/total), ha="center")
```

Fig 4. Data Analysis

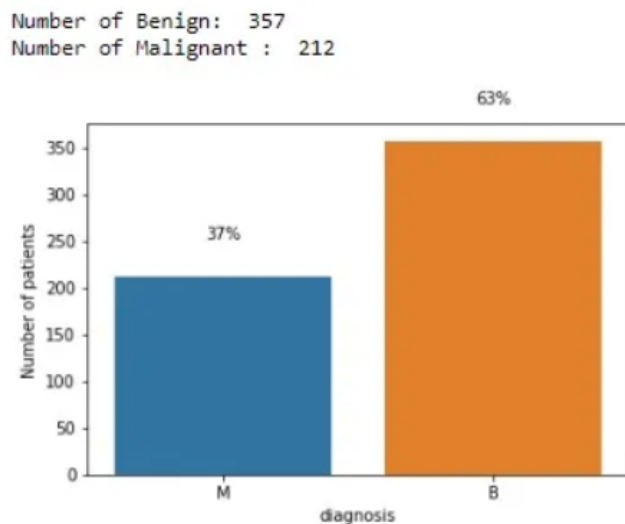


Fig 5. Class Distribution

2.3.4 Data Visualization

Data visualization helps to identify the trends, patterns, and relationships in data. Heatmap displays data using color-coded cells to represent the values of a matrix or table. In order to visualize data, we are going to use seaborn plots. Before plotting our data, we need to normalization or standardization. Because differences between values of features are very high to observe on plot. I plot features in 3 group and each group includes 10 features to observe better.

```
# first ten features
data_dia = y
data = x
# standardization of the data
data_n_2 = (data - data.mean()) / (data.std())

data = pd.concat([y,data_n_2.iloc[:,0:10]],axis=1)
data = pd.melt(data,id_vars="diagnosis",
               var_name="features",
               value_name='value')

plt.figure(figsize=(10,10))
sns.violinplot(x="features", y="value", hue="diagnosis",
               data=data,split=True, inner="quart",palette = "Set2")
plt.xticks(rotation=90)
```

Fig 6. Exploring data using Violinplot

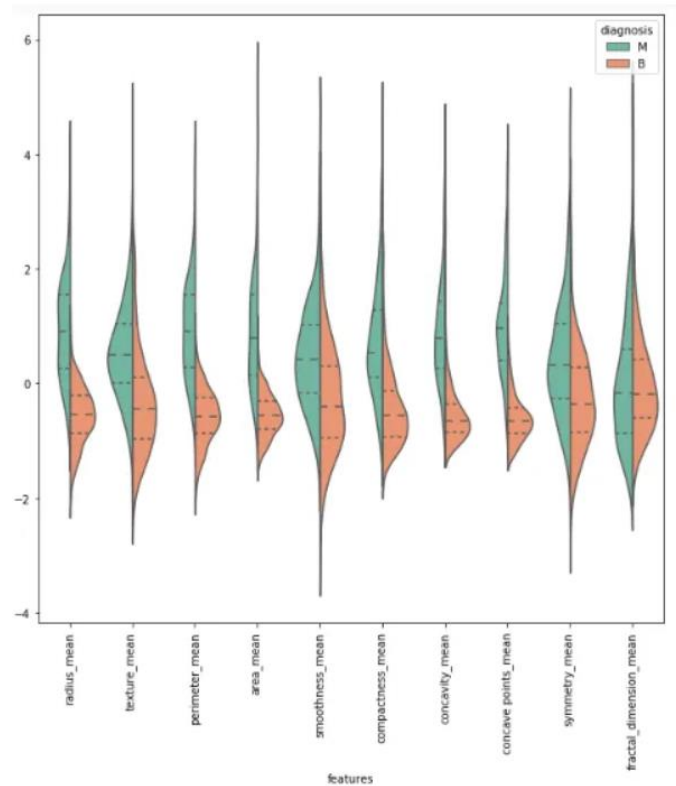


Fig 7. Data Interpretation

Histogram shows the distribution of numerical data by dividing the data into bins and counting the number of data points in each bin.

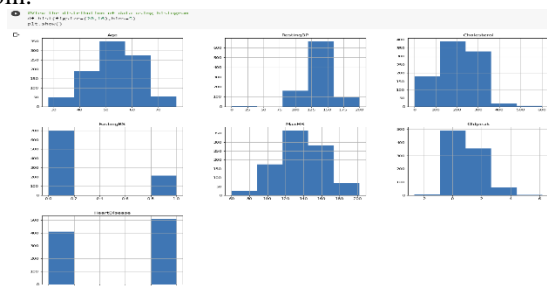


Fig 8. Distribution of numerical data using histograms

Box plot is a type of data visualization that shows the distribution of numerical data using quartiles and outliers. It presents a summary of the range, median, and spread of the data

in a compact and easy-to-understand format. The “box” in the plot represents the interquartile range (IQR), which contains 50% of the data, and the “whiskers” extending from the box show the range of the data within a certain number of standard deviations from the mean. Any data points that fall outside the whiskers are considered outliers and are represented by dots. Overall, the box plot provides a clear and concise summary of the distribution of data and can be a useful tool for identifying patterns and trends in the data.

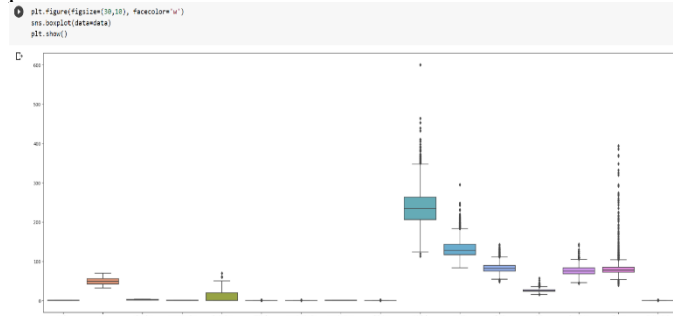


Fig 9. Boxplot using quartiles and outliers.

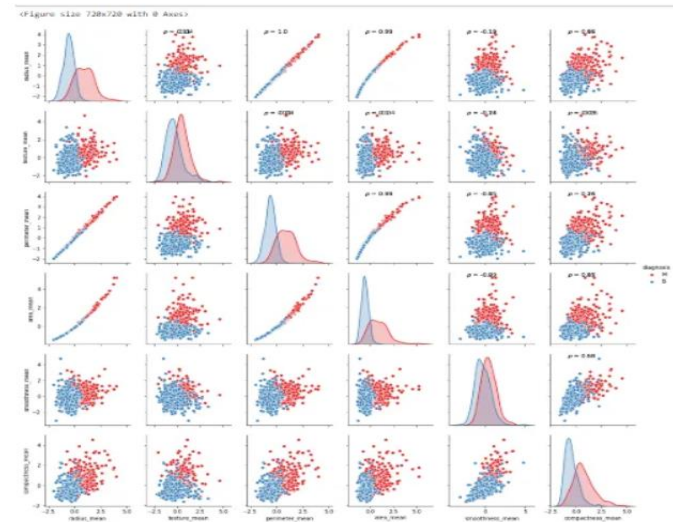


Fig 10. Exploring data using pair plot.

2.3.5 Model Building & Validation

Classification a supervised machine learning technique which involves predicting a categorical or discrete output variable based on input features. The primary objective of classification is to determine a decision boundary that can separate distinct classes within the data. In the case of breast cancer prediction, the goal is to determine whether a person has breast cancer or not based on certain factors other health indicators. If the patient is identified with breast cancer determining the stage of breast cancer like benign or malignant is necessary for the health-care providers to treat the disease effectively. So, presence or absence of breast cancer is identified.

```
In [21]: nonlr_clf_dict = dict()

# 1
from sklearn.linear_model import LogisticRegression
clf = LogisticRegression()
nonlr_clf_dict['Logistic Regression'] = clf

# 2
from sklearn.ensemble import RandomForestClassifier
clf = RandomForestClassifier()
nonlr_clf_dict['Random Forest Classifier'] = clf

# 3
from sklearn.ensemble import GradientBoostingClassifier
clf = GradientBoostingClassifier()
nonlr_clf_dict['Gradient Boosting Classifier'] = clf

# 4
from sklearn.ensemble import ExtraTreesClassifier
clf = ExtraTreesClassifier()
nonlr_clf_dict['Extra Trees Classifier'] = clf

# 5
from xgboost import XGBClassifier
clf = XGBClassifier()
nonlr_clf_dict['XGB Classifier'] = clf

# 6
from sklearn.neighbors import KNeighborsClassifier
clf = KNeighborsClassifier()
nonlr_clf_dict['KNeighbors Classifier'] = clf

# 7
from sklearn.svm import SVC
clf = SVC(kernel='rbf', probability=True)
nonlr_clf_dict['SVM Classifier'] = clf
```

Model Classifiers

Fig 11. Model Classifiers

In this study, 569 patients were randomly divided into a learning set (70%) for model development and a validation set (30%) to assess performance. Seven classifiers (Logistic Regression, Random Forest, Gradient Boosting, Extra Trees, XGB, KNeighbors, SVM) as shown in fig 11 were paired with eight feature selection methods (Correlation, Chi-square, RFE, RFECV, Random Forest, Extra Trees, L1-based, Voted) resulting in 56 distinct models as shown in fig 12 . The goal was to identify the best-performing model based on pre-defined metrics. This comprehensive approach aims to uncover the most effective algorithm-feature selection combination for optimal predictive modeling within our dataset.

```
# 1. Feature selection with correlation (16)
fs_core = ['texture_mean', 'area_mean', 'smoothness_mean', 'concavity_mean', 'symmetry_mean',
           'fractal_dimension_mean', 'texture_se', 'area_se', 'smoothness_se', 'concavity_se',
           'symmetry_se', 'fractal_dimension_se', 'smoothness_worst', 'concavity_worst',
           'symmetry_worst', 'fractal_dimension_worst']

# 2. Univariate feature selection SelectKBest, chi2
fs_chi2 = ['texture_mean', 'area_mean', 'concavity_mean', 'symmetry_mean', 'area_se',
           'concavity_se', 'smoothness_worst', 'concavity_worst', 'symmetry_worst',
           'fractal_dimension_worst']

# 3. Recursive feature elimination (RFE) with random forest
fs_rfe = ['texture_mean', 'area_mean', 'smoothness_mean', 'concavity_mean', 'area_se',
           'smoothness_se', 'concavity_se', 'smoothness_worst', 'symmetry_worst']

# 4. Recursive feature elimination with cross validation(RFECV) with random forest
fs_rfecv = ['texture_mean', 'area_mean', 'smoothness_mean', 'concavity_mean', 'fractal_dimension_mean',
            'area_se', 'concavity_se', 'concavity_worst', 'symmetry_worst']

# 5. Tree based feature selection with random forest classification
fs_rf = ['texture_mean', 'area_mean', 'concavity_mean', 'area_se', 'concavity_se',
         'fractal_dimension_se', 'smoothness_worst', 'concavity_worst', 'symmetry_worst',
         'fractal_dimension_worst']

# 6. ExtraTree based feature selection
fs_extratree = ['texture_mean', 'area_mean', 'concavity_mean', 'fractal_dimension_mean', 'area_se',
               'concavity_se', 'smoothness_worst', 'concavity_worst',
               'symmetry_worst', 'fractal_dimension_worst']

# 7. L1 feature selection (LinearSVC)
fs_l1 = ['texture_mean', 'area_mean', 'area_se']

# 8. Vote based feature selection
fs_voted = ['texture_mean', 'area_mean', 'smoothness_mean', 'concavity_mean',
            'fractal_dimension_mean', 'area_se', 'concavity_se', 'smoothness_worst',
            'concavity_worst', 'symmetry_worst', 'fractal_dimension_worst']
```

Selected features

Fig 12. Selected Features

2.3.5.1 Model Evaluation

Utilizing a 70/30 training-testing data split, our approach integrated 5-fold cross-validation. This entailed partitioning the dataset into five subsets, leveraging four segments for robust model training, and reserving the fifth for gauging model performance and its ability to generalize to new data.

2.3.5.2 Training and testing performance for feature selection method.

After a comprehensive analysis of outputs derived from various methods across multiple classifiers, our study identifies the Logistic Regression model as the most effective among the

evaluated classifiers. This conclusion arises from an evaluation of performance metrics encompassing accuracy, precision, recall, and F1-score across different feature selection techniques. Logistic Regression consistently exhibited superior predictive accuracy and robustness in generalizing to new data compared to alternative classifiers. Additionally, its adeptness in handling both linear and non-linear relationships within the data further supported its superior performance.

The finding accentuates Logistic Regression's potential as a practical and efficient predictive modeling tool in this domain. Further refinements and exploration could potentially unveil additional nuances, enhancing its performance for future research or practical applications.

Training Performance: 5-fold Cross Validation Scores on Training Data							
Algorithm	Feat. Selec.	Accuracy	Average_Precision	F1	Precision	Recall	ROC_AUC
Logistic Regression	Correlation	0.9775	0.9936	0.9700	0.9804	0.9611	0.9939
SVM Classifier	Chi2	0.9648	0.9914	0.9532	0.9670	0.9411	0.9936
SVM Classifier	RFE	0.9626	0.9908	0.9481	0.9739	0.9286	0.9930
Logistic Regression	RFECV	0.9724	0.9933	0.9632	0.9800	0.9477	0.9938
Logistic Regression	RF	0.9774	0.9945	0.9704	0.9742	0.9675	0.9947
Logistic Regression	Extra trees	0.9725	0.9931	0.9635	0.9752	0.9546	0.9931
Logistic Regression	L1	0.8994	0.9514	0.8601	0.9214	0.8088	0.9585
Logistic Regression	Voted	0.9725	0.9932	0.9635	0.9752	0.9546	0.9935

Fig 13. Feature selection Vote based.

Testing Performance							
Algorithm	Feat. Selec.	Accuracy	F1	Precision	Recall	ROC_AUC	
Logistic Regression	Correlation	0.9591	0.9412	0.9492	0.9333	0.9532	
SVM Classifier	Chi2	0.9591	0.9421	0.9344	0.9500	0.9570	
SVM Classifier	RFE	0.9649	0.9492	0.9655	0.9333	0.9577	
Logistic Regression	RFECV	0.9591	0.9402	0.9649	0.9167	0.9493	
Logistic Regression	RF	0.9766	0.9661	0.9828	0.9500	0.9705	
Logistic Regression	Extra trees	0.9649	0.9492	0.9655	0.9333	0.9577	
Logistic Regression	L1	0.9064	0.8621	0.8929	0.8333	0.8896	
Logistic Regression	Voted	0.9591	0.9412	0.9492	0.9333	0.9532	

Fig 14. Feature selection Vote based

In the quest to enhance predictive model accuracy, multiple models were trained and fine-tuned using 16 different sets of features. These features were optimized using specific methods tailored to each model's needs. Each model and its feature set underwent rigorous evaluation through a 5-fold cross-validation, assessing Accuracy, AUC, and sensitivity metrics. Ultimately, the Logistic Regression model stood out, utilizing a refined feature subset of 10 elements derived from Random Forest. The performance table illustrates that Logistic Regression achieved impressive scores of 0.977 for accuracy and 0.971 for AUC when tested against the validation dataset. These results highlight the Logistic Regression model's reliability and effectiveness in making accurate predictions within this specific context.

2.3.5.3 Logistic Regression

Logistic Regression is a machine learning algorithm commonly used for binary classification tasks. It involves analyzing the

relationship between predictor variables and the probability of a specific outcome occurring. In Logistic Regression, a sigmoid function is employed to map the input features to a value that ranges between 0 and 1. This value represents the probability of the output class. To determine the predicted class, the output of the sigmoid function is compared with a threshold value. Logistic regression produces result in binary format which is used to predict the outcome of a categorical dependent variable.

```
[ ] # Logistic Regression Machine Learning Model
from sklearn.linear_model import LogisticRegression
classifier_lr = LogisticRegression(random_state=0, C= 10, penalty= 'l2')
model(classifier_lr)
model_evaluation(classifier_lr)
```

Fig 15. Logistic Regression

2.3.5.4 Hyperparameter Tuning

With Scikit-Learn's RandomizedSearchCV method, we can define a grid containing ranges of hyperparameters. This method then randomly selects combinations of these values from the grid and conducts K-Fold Cross-Validation for each combination. This systematic process allows us to efficiently explore various hyperparameter configurations and select the best combination that optimizes the model's performance based on the specified evaluation metric.

```
from sklearn.ensemble import ExtraTreesClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.svm import SVC
from sklearn.neighbors import KNeighborsClassifier

models1 = {
    'Logistic Regression' : LogisticRegression(),
    'RandomForestClassifier': RandomForestClassifier(),
    'GradientBoostingClassifier': GradientBoostingClassifier(),
    'ExtraTreesClassifier': ExtraTreesClassifier(),
    'KNeighborsClassifier' : KNeighborsClassifier(),
    'SVC': SVC()
}

params1 = {
    'Logistic Regression' : {'penalty' : ['l1','l2'], 'C' :
np.logspace(0, 4, 10), 'solver' :['liblinear']},
    'RandomForestClassifier': { 'n_estimators': [16, 32] },
    'GradientBoostingClassifier': { 'n_estimators': [16, 32],
    'learning_rate': [0.8, 1.0] },
    'ExtraTreesClassifier': { 'n_estimators': [16, 32] },
    'KNeighborsClassifier' : { 'n_neighbors' : [1,2,5,6,7,10], 'weights':
    ['uniform', 'distance']},
    'SVC': {'kernel': ['linear','rbf'], 'C': [1, 10], 'gamma': [0.001,
    0.0001]}
}
```

Fig 16. Define Hyperparameter Grid for RandomizedSearchCV

We've conducted a comprehensive comparison among various strategies employed to enhance performance, highlighting the gains achieved with each approach. The table below presents the conclusive outcomes encompassing all improvements implemented, including those from the third part excluding hyperparameter search. Our analysis indicates that when utilizing ROC_AUC as the scoring metric, there appears to be negligible improvement. However, with accuracy as the scoring metric, there is a discernible but slight enhancement observed.

2.3.6 Model Deployment

Deploying predictive models comes with complexities, involving managing their lifecycle, storage formats, deployment methods, and a wide array of technical choices. In

this scenario, two primary models emerge for deployment: Batch Mode and Real-time Mode. These models represent distinct approaches, each with unique considerations for how models interact with data and provide predictions in various environments and use cases.

2.3.6.1 Local Host (Batch Mode)

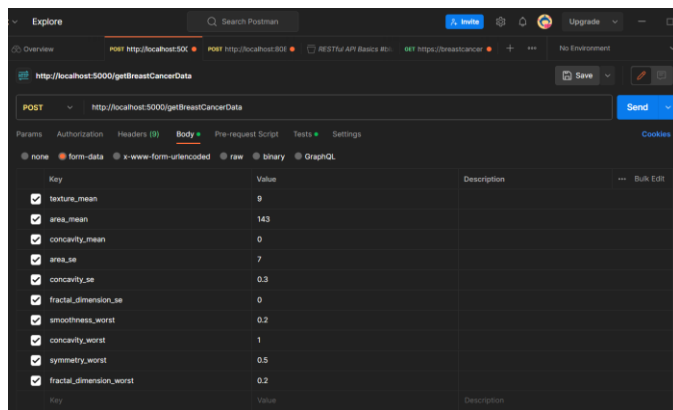
Before deploying the ML model, we rigorously tested it on a local host environment. This preliminary stage allowed us to evaluate the model's behavior within a controlled setup, ensuring its functionality, accuracy, and compatibility with the intended system architecture. Local testing provided crucial insights into the model's performance and behavior, enabling necessary adjustments and refinements before actual deployment, thereby enhancing the model's readiness for production.

```
from sklearn.linear_model import LogisticRegression
import sklearn.model_selection
import pickle
import os
clf_lr = LogisticRegression(C=7.7426, penalty='l2', solver='liblinear')
clf_lr.fit(X, y)
os.chdir('/Users/krish/Desktop/deployment/')
pickle.dump(clf_lr, open('lr_model', 'wb'))
load_model = pickle.load(open(filename, 'rb'))
result = load_model.predict(X_unseenData)
print(result)
```

Fig 17. Local Host Creation

2.3.6.2 Real Time (Post Man)

After deploying the system, we conducted output checks using Postman. This involved sending requests to the deployed API endpoints and inspecting the responses to ensure the system functions as intended. Postman allowed us to simulate various scenarios and verify the accuracy and reliability of the API outputs, ensuring seamless communication and functionality in real-world usage.



Key	Value	Description
texture_mean	9	
area_mean	143	
concavity_mean	0	
area_se	7	
concavity_se	0.3	
fractal_dimension_se	0	
smoothness_worst	0.2	
concavity_worst	1	
symmetry_worst	0.5	
fractal_dimension_worst	0.2	

Fig 18. Post man output

2.3.6.3 Real Time (Flask API)

The code developed processes input data, utilizes it for ML model input, and delivers predictions to end-users. This involves data processing, invoking the model's predict function, and presenting the prediction in JSON format as a response. The

API incorporates the ML model as a pickle file to enable this prediction functionality.

```
import numpy as np
from flask import Flask, request, jsonify, render_template
import pickle
from sklearn.preprocessing import StandardScaler
from flask_cors import CORS

app = Flask(__name__, static_url_path='/static')
CORS(app, resources={r'/*': {'origins': ["http://localhost",
"http://localhost:8080",
"https://myapp.herokuapp.com",
"http://www.e-hospital.ca/gastroImagePrediction",
"http://www.e-hospital.ca",
"http://localhost:5000",
"http://localhost:3000",
"https://e-react-frontend-55dbf7a5897e.herokuapp.com"]}}})

model = pickle.load(open('model.pkl', 'rb'))
scaler = pickle.load(open('scaler.pkl', 'rb'))
```

Fig 19. Flask API Creation

2.3.6.4 Defining API Endpoints

Endpoints for API were defined using Flask Api's decorators, such as @app.get or @app.post. These endpoints accept input data and returns a prediction from the ML model.

```
6 function BreastCancerML() {
7   const [showDiagnose, setShowDiagnose] = useState(false)
8   const [data, setData] = useState({})
9   const location = useLocation();
10  const [prediction, setPrediction] = useState('');
11  const patient_id = location.state?.id;
12  const BASE_URL = 'https://e-react-node-backend-22e986ad9f3.herokuapp.com'; // Update with your node backend app URL
13  //const BASE_URL = 'http://localhost:5000'; // Update with your node backend app URL
14
15  useEffect(() => {
16    // Function to retrieve breast cancer data
17    const getBreastCancerData = async () => {
18      try {
19        console.log("patient found ", patient_id);
20        const response = await axios.post(`${BASE_URL}/getBreastCancerData`, {
21          patient_id,
22        });
23        const { data } = response;
24        setData(data);
25        if (data.error) {
26          console.log(JSON.stringify(data.error));
27        } else {
28          delete data.id;
29          delete data.patient_id;
30          console.log(data);
31          const responsePrediction = await axios.post('https://breastcancerml-717ef42b0eb4.herokuapp.com/predict', data, {
32            headers: {
33              'Content-Type': 'multipart/form-data', // Important: Set the content type to form data
34            },
35          });
36          console.log(responsePrediction.data);
37          setPrediction(responsePrediction.data);
38        } catch (error) {
39          alert('Error: ${error.message}');
40        }
41      }
42    };
43    getBreastCancerData();
44  }, [patient_id]);
45
46  return (
47
```

Fig 20. API end points

2.3.6.5 Integration with E-Hospital database

To seamlessly integrate and archive predictions alongside patient information, dedicated tables and APIs were established within the existing database. For breast cancer predictions, pertinent records reside in the database table. We have created a table (breastcancerdetails) in database to get the breast cancer prediction attributes (like radius mean, texture mean, perimeter mean, area mean etc.) which are used by machine learning algorithm. The primary key is patient id which is used to retrieve the patient records from breast cancer table. Similarly, for breast cancer disease predictions, relevant records are housed in the database table, with prediction results stored in the "Breast_cancer_disease" table.

```
sql = `
SELECT *
FROM breast_cancer_details
WHERE patient_id = "${patient_id}"`
```

Fig 21. Selecting records from breast cancer details table

2.3.7 Interactive Front-End Page with the ML model to Breast Cancer Prediction

Combining JavaScript, HTML, CSS and ReactJS facilitates the development of dynamic web pages enabling user engagement with machine learning models. HTML acts as the interface, collecting user inputs, showcasing prediction results, and offering a user-friendly platform for model interaction. CSS, on the other hand, defines the visual layout, appearance, and formatting of HTML documents, enhancing their presentation and usability. Together, these technologies enable the integration of machine learning features into intuitive web interfaces, streamlining data input and prediction retrieval processes.

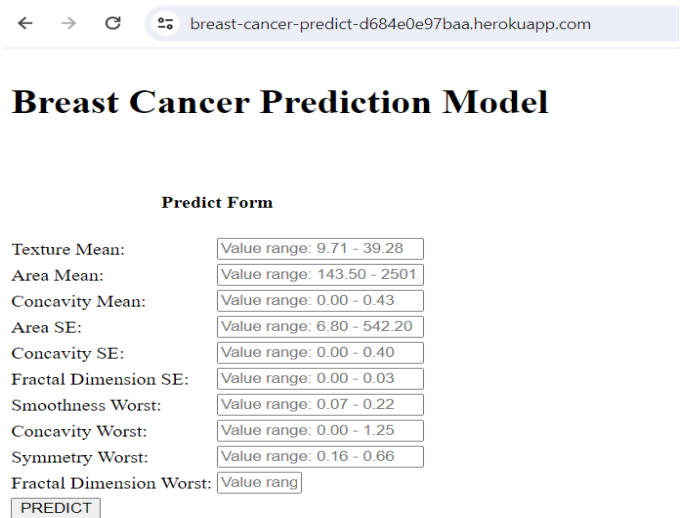
3. Overall Results and Analysis

The project's evaluation showcased promising outcomes from the ML models designed to predict breast cancer and coronary artery disease, boasting notably high accuracy scores. Among these models, the Linear Regression model stood out with an impressive 97.20% accuracy rate in predicting breast cancer. To streamline deployment, a Flask API was crafted, allowing seamless integration of the ML model into cloud platforms like Heroku. Serving as the backend API for the web application, FlaskAPI facilitates smooth coordination with the machine learning model. It efficiently processes user inputs from the interface, directs them to the model for prediction, and retrieves the output for display on the web page. This integration within the e-hospital web application enables users to effortlessly input their data and receive real-time predictions. As a result of this integration, the e-hospital platform successfully generated significant outcomes.

4. Deployment Plan

4.1 Deployment of flask API on Heroku Cloud Platform

The deployment of a Flask API on the Heroku cloud platform entails a structured process. Following configuration setup and dependencies specification within the Flask application, integration with Heroku involves linking the app to a Git repository, enabling seamless code transfer. Leveraging Heroku's Command Line Interface (CLI), code is pushed to the platform's repository, triggering an automated build and deployment sequence. Scaling the application's dynos, adjusting configurations to align with Heroku's environment variables and port settings, and vigilantly managing the deployed API via the Heroku dashboard or CLI are crucial steps in ensuring the API's operational efficiency and reliability within the cloud infrastructure.



← → ↺ breast-cancer-predict-d684e0e97baa.herokuapp.com

Breast Cancer Prediction Model

Predict Form

Texture Mean: Value range: 9.71 - 39.28

Area Mean: Value range: 143.50 - 2501

Concavity Mean: Value range: 0.00 - 0.43

Area SE: Value range: 6.80 - 542.20

Concavity SE: Value range: 0.00 - 0.40

Fractal Dimension SE: Value range: 0.00 - 0.03

Smoothness Worst: Value range: 0.07 - 0.22

Concavity Worst: Value range: 0.00 - 1.25

Symmetry Worst: Value range: 0.16 - 0.66

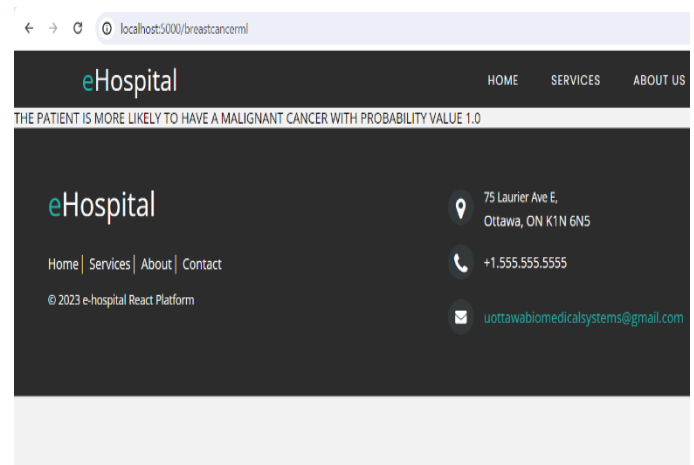
Fractal Dimension Worst: Value rang

PREDICT

Fig 22. API deployed in heroku

4.2 Integrating with the E-Hospital Platform

After obtaining the updated Git repository and integrating necessary changes, initial predictions and outputs were verified locally. Upon thorough review, all modifications were committed and pushed to GitHub. Integration with the E-Hospital platform involved raising a pull request to merge these changes. Post-merger, the predictions were re-evaluated on the E-Hospital website to validate the accuracy and functionality of the integrated output.



← → ↺ localhost:5000/breastcancerm1

eHospital HOME SERVICES ABOUT US

THE PATIENT IS MORE LIKELY TO HAVE A MALIGNANT CANCER WITH PROBABILITY VALUE 1.0

eHospital

75 Laurier Ave E,
Ottawa, ON K1N 6N5

+1.555.555.5555

uottawabiomedicalsystems@gmail.com

© 2023 e-hospital React Platform

Fig 23. Initial results in Local host

Parameters	Values
Area Mean	366.3
Area St	23.55
Convexity Mean	0.89464
Convexity St	0.82287
Convexity Worst	0.239
Fractal Dimension St	0.49823
Fractal Dimension Worst	0.87250
Sensitiveness Worst	0.140
Symmetry Worst	0.2977
Texture Mean	4.36

THE PATIENT IS MORE LIKELY TO HAVE A BENIGN CANCER WITH PROBABILITY VALUE 0.895

Fig 24. Final results integrated in ehospital webpage

5. Conclusions and Future Works

In conclusion, the utilization of machine learning models for breast cancer prediction has demonstrated promising results. Through rigorous analysis and model development, notably high accuracy rates were achieved, with the Linear Regression model standing out with an accuracy of 97.20%. The successful deployment of Flask API on cloud platforms like Heroku facilitated real-time predictions, enabling seamless integration within the e-hospital web application. This endeavor not only showcased the potential of machine learning in healthcare but also highlighted the practical implementation of predictive models in aiding early detection and improved decision-making in breast cancer diagnosis. As advancements continue in the field of machine learning, the integration and refinement of these models hold immense promise for enhancing medical diagnostics and ultimately improving patient care.

5.1 Future Works

Several promising avenues exist for advancing breast cancer prediction through ML models. Exploring additional features or employing advanced feature engineering techniques could refine predictive capabilities, potentially incorporating genetic data or emerging biomarkers. Further investigation into sophisticated algorithms or ensemble methods, such as deep learning or hybrid models, might unveil intricate data patterns, potentially enhancing accuracy. Emphasizing model interpretability and transparency aids comprehension, crucial in healthcare contexts. Developing systems for real-time integration of patient data could facilitate instant predictions for proactive interventions. Rigorous validation studies involving clinical experts are essential to assess model performance across diverse patient cohorts, ensuring real-world effectiveness. Addressing ethical implications around privacy, bias mitigation, and fairness aligns with ethical healthcare practices. Finally, improving the user interface of predictive tools within healthcare systems can enhance usability and promote widespread adoption among healthcare professionals.

Advancements in these areas hold promise for enhancing the accuracy and practicality of ML-based breast cancer prediction models, ultimately contributing to improved patient care and clinical decision-making.

6. References

To work on our project, we have used the following references to understand the project in depth and to make a clear analysis of our work.

- [1] Gupta, P. and Garg, S. (2020) 'Breast cancer prediction using varying parameters of machine learning models', *Procedia Computer Science*, 171, pp. 593–601. doi:10.1016/j.procs.2020.04.064.
- [2] Islam, Md.M. *et al.* (2020) *Breast cancer prediction: A comparative study using machine learning techniques - SN computer science*, SpringerLink. Available at: <https://link.springer.com/article/10.1007/s42979-020-00305-w>.
- [3] Mallika, M. and Suresh Babu, K. (2023) 'Breast cancer prediction using machine learning algorithms', *International Journal of Science and Research (IJSR)*, 12(10), pp. 1235–1238. doi:10.21275/sr231015173828.
- [4] TIWARI, M. *et al.* (2020) 'Breast cancer prediction using Deep Learning and Machine Learning Techniques', *SSRN Electronic Journal* [Preprint]. doi:10.2139/ssrn.3558786.
- [5] Rovshenov, A. and Peker, S. (2022) 'Performance comparison of different machine learning techniques for early prediction of breast cancer using Wisconsin Breast Cancer Dataset', *2022 3rd International Informatics and Software Engineering Conference (IISEC)* [Preprint]. doi:10.1109/iisec56263.2022.9998248.
- [6] Aggarwal, R. (2022) 'An intelligent system for diagnosis and prediction of breast cancer malignant features using machine learning algorithms', *Machine Learning and Deep Learning Techniques for Medical Science*, pp. 143–151. doi:10.1201/9781003217497-8.
- [7] Jiang, X. and Xu, C. (2022) *Improving clinical prediction of later occurrence of breast cancer metastasis using Deep Learning and machine learning with grid search* [Preprint]. doi:10.20944/preprints202206.0394.v1.
- [8] Fatima M, Pasha M. Survey of machine learning algorithms for disease diagnostic. *J Intell Learn Syst Appl.* 2017;91–16. <https://doi.org/10.4236/jilsa.2017.91001>