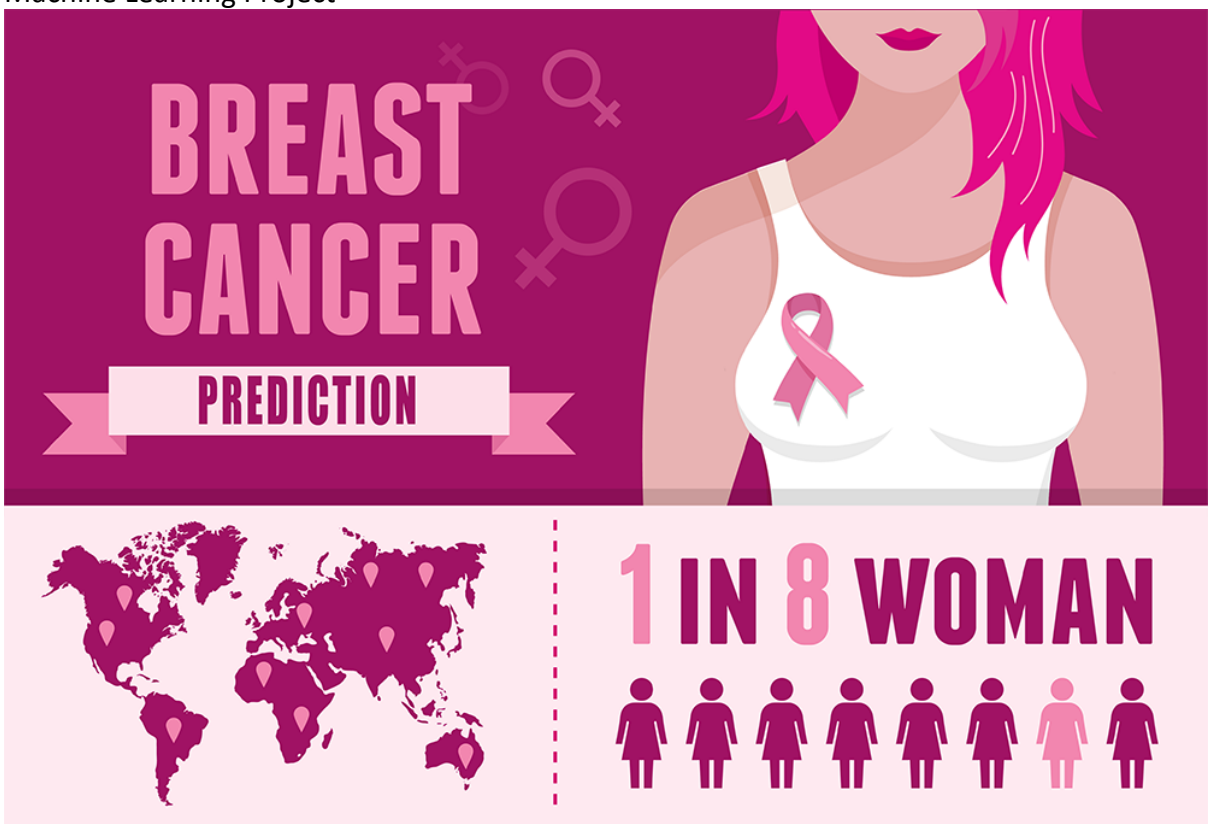
Shreya Bhattacharjee
6861
Machine Learning Project



Shreya Bhattacharjee
6861
Machine Learning Project

MAHATMA EDUCATION SOCIETY'S

PILLAI COLLEGE OF ARTS, COMMERCE & SCIENCE
(Autonomous) NEW PANVEL

PROJECT REPORT ON

"Breast Cancer Risk Prediction Model"

IN PARTIAL FULFILLMENT OF

MASTER OF DATA ANALYTICS

SEMESTER III– 2024-25

PROJECT GUIDE Name: Omkar Sherkhane

SUBMITTED BY: Shreya Bhattacharjee
ROLL NO: 6861

Shreya Bhattacharjee
6861
Machine Learning Project

Mahatma Education Society's

PILLAI COLLEGE OF ARTS,COMMERCE & SCIENCE

(Autonomous)

Re-accredited "A" Grade by NAAC (3rd Cycle)

Project Completion Certificate

THIS IS TO CERTIFY THAT

Shreya Bhattacharjee

of **M.Sc. Data Analytics Part - II** has completed the project titled **Breast Cancer Prediction Model** of subject **Machine Learning** under our guidance and supervision during the academic year 2024-25 in the department of Master of Data Analytics.

Project Guide Course Coordinator Head of the Department

Shreya Bhattacharjee
6861
Machine Learning Project

Introduction

Breast cancer is one of the most prevalent forms of cancer affecting women worldwide, contributing significantly to global cancer mortality rates. According to statistics from the World Health Organization (WHO), breast cancer accounts for nearly 15% of all cancer-related deaths among women. Early detection of breast cancer can greatly improve survival rates, as it allows for timely intervention and treatment before the disease progresses to advanced stages. The early identification of whether a tumor is benign or malignant plays a crucial role in this process, enabling healthcare professionals to devise effective treatment plans that can potentially save lives.

The classification of breast cancer tumors—whether benign or malignant—is typically based on the analysis of cell characteristics derived from medical tests such as biopsies. Various attributes, such as clump thickness, uniformity in cell size and shape, marginal adhesion, and mitosis, are critical in determining the nature of the tumor. Traditionally, this task has been performed manually by pathologists, but advancements in machine learning techniques now offer the potential to automate and improve the accuracy of these classifications. Machine learning algorithms can learn patterns in complex datasets and provide reliable predictions, which could augment the efforts of medical professionals, reduce human error, and facilitate faster decision-making.

This project aims to leverage machine learning algorithms to build a prediction system that can classify breast tumors as benign or malignant based on several input features. We will use a dataset containing relevant characteristics of cell samples, including features commonly observed in biopsy reports, to train and test a machine learning model. The key features used in this dataset include clump thickness, uniformity of cell size, uniformity of cell shape, marginal adhesion, single epithelial cell size, bare nuclei, bland chromatin, normal nucleoli, and mitoses. By developing a robust classification model, this project seeks to contribute to the broader efforts in enhancing early detection mechanisms and improving patient outcomes.

The prediction system will be built using a combination of data preprocessing techniques, machine learning algorithms, and a web-based interface developed using the Flask framework. The Flask web framework will provide a user-friendly platform for medical practitioners to input biopsy data and receive real-time predictions about the likelihood of malignancy.

Shreya Bhattacharjee
6861
Machine Learning Project

Technologies and Libraries Used

This project integrates a range of powerful technologies and libraries that are essential for data preprocessing, machine learning model development, and deployment. Below is an overview of the key tools and libraries employed in this project:

1. **NumPy**: NumPy is a fundamental library for numerical operations in Python, especially for working with arrays and matrices. It provides efficient functions for handling large datasets, performing linear algebra operations, and executing fast computations, which are essential for the preprocessing steps in this project.
2. **Pandas**: Pandas is a widely-used data manipulation and analysis library that simplifies the process of cleaning and organizing data. In this project, Pandas is used to load the dataset, perform exploratory data analysis (EDA), handle missing values, and manipulate the data into a suitable format for model training.
3. **Matplotlib & Seaborn**: These two libraries are instrumental for data visualization. Matplotlib provides basic plotting capabilities such as line plots, bar charts, and histograms. Seaborn, built on top of Matplotlib, offers more sophisticated statistical graphics and styling options. Together, they are used for visualizing key data insights, plotting histograms, correlation matrices, and other graphs that aid in understanding relationships within the dataset.
4. **Scikit-learn**: Scikit-learn is the primary machine learning library used in this project. It offers a comprehensive range of tools for data preprocessing, model training, and evaluation. In particular, it provides:
    - **Machine Learning Models**: The project utilizes several classifiers from Scikit-learn, including Decision Tree Classifier, K-Nearest Neighbors (KNN), Naive Bayes (GaussianNB), and Support Vector Machine (SVC), to predict whether a breast tumor is benign or malignant.
    - **Preprocessing Tools**: Scikit-learn's utilities such as `StandardScaler` help standardize features, improving the performance of machine learning algorithms.
    - **Model Evaluation**: Methods like `cross_val_score` are used to validate model performance using cross-validation. Additionally, `train_test_split` is utilized to divide the dataset into training and testing sets to evaluate model accuracy.

- **Hyperparameter Tuning**: The `GridSearchCV` utility is employed for optimizing model performance by searching for the best hyperparameters.

5. **Flask**: Flask is a lightweight web framework for Python, enabling the creation of web applications. In this project, Flask is used to build a user-friendly web interface where users can input biopsy data and receive predictions from the trained machine learning model in real time. It facilitates the deployment of the machine learning model in a practical, accessible manner.
6. **Pickle**: The trained machine learning model is serialized using Pickle, allowing it to be saved and loaded efficiently for use in the Flask application. Pickle ensures that the model's learned parameters can be preserved and accessed later without retraining, making real-time predictions faster and more efficient.

Work flow of the Project

1. **Data Collection and Preprocessing**:

The dataset, such as the **Wisconsin Breast Cancer Dataset**, contains features describing cell nuclei properties.

Libraries (NumPy, Pandas, Matplotlib) are used for data manipulation and visualization.

Data is cleaned and scaled using **StandardScaler** to ensure all features are on the same scale.

2. **Machine Learning Model Development**:

Several models are used: **Decision Tree Classifier**, **K-Nearest Neighbors (KNN)**, **Naive Bayes (GaussianNB)**, and **Support Vector Machine (SVC)**.

Models are evaluated using metrics like **confusion matrix**, **classification report**, and **accuracy score**. **K-Fold Cross-Validation** ensures generalization.

3. **Model Selection and Tuning**:

GridSearchCV optimizes model hyperparameters.

The best-performing model is saved using **Pickle** for future predictions.

4. **Web Interface Using Flask**:

A **Flask** web app allows users to input cell features and get predictions (benign or malignant) in real-time from the trained model.

How Prediction works

The prediction process is straightforward and user-friendly:

1. **User Input**: The user enters the following cell features into a web form provided by the Flask app:
    - Clump Thickness
    - Uniform Cell Size
    - Uniform Cell Shape
    - Marginal Adhesion
    - Single Epithelial Cell Size
    - Bare Nuclei
    - Bland Chromatin
    - Normal Nucleoli
    - Mitoses
2. **Model Prediction**: These input values are fed into the pre-trained machine learning model. The model, which was trained to recognize patterns in cell features, makes a prediction on whether the tumor is benign or malignant based on the input.
    - If the model predicts a value of **4**, it indicates a **malignant** tumor (i.e., breast cancer).
    - Any other prediction value suggests the tumor is **benign** (non-cancerous).
3. **Result Display**: The predicted result is displayed on the webpage, informing the user whether the patient is likely to have breast cancer or not. This provides immediate feedback for diagnostic purposes.

Shreya Bhattacharjee
6861
Machine Learning Project

Code and Output

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score
from sklearn.model_selection import train_test_split
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import KFold
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import GridSearchCV
from sklearn.svm import SVC
import seaborn as sns

%matplotlib inline
```

Exploratory Analysis

Read and load Dataset

```
# Load Data
df = pd.read_csv('/content/data.csv')
df.head()
```

	id	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean	concavity_mean
0	842302	M	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.3001
1	842517	M	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.0869
2	84300903	M	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.1974
3	84348301	M	11.42	20.38	77.58	386.1	0.14250	0.28390	0.2414
4	84358402	M	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.1980

5 rows x 33 columns

Shreya Bhattacharjee
6861
Machine Learning Project

```
#Shape of the Dataset
df.shape
```

(569, 33)

```
df.describe()
```

	id	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean	concavity_mean	concave points_mean
count	5.690000e+02	569.000000	569.000000	569.000000	569.000000	569.000000	569.000000	569.000000	569.00000
mean	3.037183e+07	14.127292	19.289649	91.969033	654.889104	0.096360	0.104341	0.088799	0.04891
std	1.250206e+08	3.524049	4.301036	24.298981	351.914129	0.014064	0.052813	0.079720	0.03880
min	8.670000e+03	6.981000	9.710000	43.790000	143.500000	0.052630	0.019380	0.000000	0.00000
25%	8.692180e+05	11.700000	16.170000	75.170000	420.300000	0.086370	0.064920	0.029560	0.02031
50%	9.060240e+05	13.370000	18.840000	86.240000	551.100000	0.095870	0.092630	0.061540	0.03350
75%	8.813129e+06	15.780000	21.800000	104.100000	782.700000	0.105300	0.130400	0.130700	0.07400
max	9.113205e+08	28.110000	39.280000	188.500000	2501.000000	0.163400	0.345400	0.426800	0.20120

8 rows × 32 columns

Data pre-processing

```
#set the ID column to be the index of the dataframe
df = df.set_index('id')
```

```
df.drop(['Unnamed: 32'],axis=1,inplace= True)
```

```
# Columns in the dataset
df.columns
```

```
Index(['diagnosis', 'radius_mean', 'texture_mean', 'perimeter_mean',
       'area_mean', 'smoothness_mean', 'compactness_mean', 'concavity_mean',
       'concave points_mean', 'symmetry_mean', 'fractal_dimension_mean',
       'radius_se', 'texture_se', 'perimeter_se', 'area_se', 'smoothness_se',
       'compactness_se', 'concavity_se', 'concave points_se', 'symmetry_se',
       'fractal_dimension_se', 'radius_worst', 'texture_worst',
       'perimeter_worst', 'area_worst', 'smoothness_worst',
       'compactness_worst', 'concavity_worst', 'concave points_worst',
       'symmetry_worst', 'fractal_dimension_worst'],
      dtype='object')
```

Shreya Bhattacharjee
6861
Machine Learning Project

Encoding Categorical Data

```
#Enumerate the diagnosis column such that M = 1, B = 0
df['diagnosis'] = df['diagnosis'].apply(lambda x: '1'
if x == 'M' else '0')
df['diagnosis'] = df['diagnosis'].astype("float64")
```

```
#The number of Benign and Maglinant cases from the
dataset.
print(df.groupby('diagnosis').size())
```

```
diagnosis
0.0    357
1.0    212
dtype: int64
```

```
df.info()
```

Shreya Bhattacharjee
6861
Machine Learning Project

```
Data columns (total 31 columns):
 #   Column                   Non-Null Count  Dtype
---  ------                   --------------  -----
 0   diagnosis                569 non-null    float64
 1   radius_mean              569 non-null    float64
 2   texture_mean             569 non-null    float64
 3   perimeter_mean           569 non-null    float64
 4   area_mean                569 non-null    float64
 5   smoothness_mean          569 non-null    float64
 6   compactness_mean         569 non-null    float64
 7   concavity_mean           569 non-null    float64
 8   concave points_mean      569 non-null    float64
 9   symmetry_mean            569 non-null    float64
 10  fractal_dimension_mean   569 non-null    float64
 11  radius_se                569 non-null    float64
 12  texture_se               569 non-null    float64
 13  perimeter_se             569 non-null    float64
 14  area_se                  569 non-null    float64
 15  smoothness_se            569 non-null    float64
 16  compactness_se           569 non-null    float64
 17  concavity_se             569 non-null    float64
 18  concave points_se        569 non-null    float64
 19  symmetry_se              569 non-null    float64
 20  fractal_dimension_se     569 non-null    float64
 21  radius_worst             569 non-null    float64
 22  texture_worst            569 non-null    float64
 23  perimeter_worst          569 non-null    float64
 24  area_worst               569 non-null    float64
 25  smoothness_worst         569 non-null    float64
 26  compactness_worst        569 non-null    float64
```

```
# get the number of missing data points per column
missing_values_count = df.isnull().sum()
missing_values_count[0:10]
```

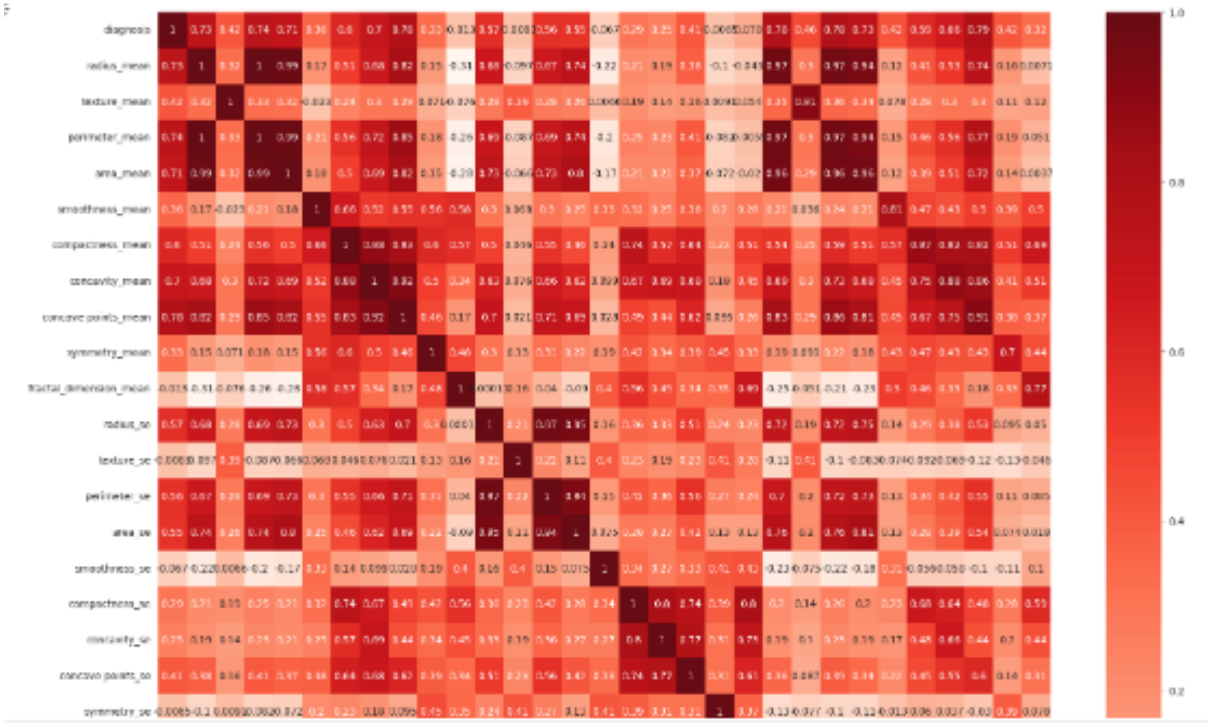
	0
diagnosis	0
radius_mean	0
texture_mean	0
perimeter_mean	0
area_mean	0
smoothness_mean	0
compactness_mean	0
concavity_mean	0
concave points_mean	0
symmetry_mean	0

dtype: int64

Data Visualization

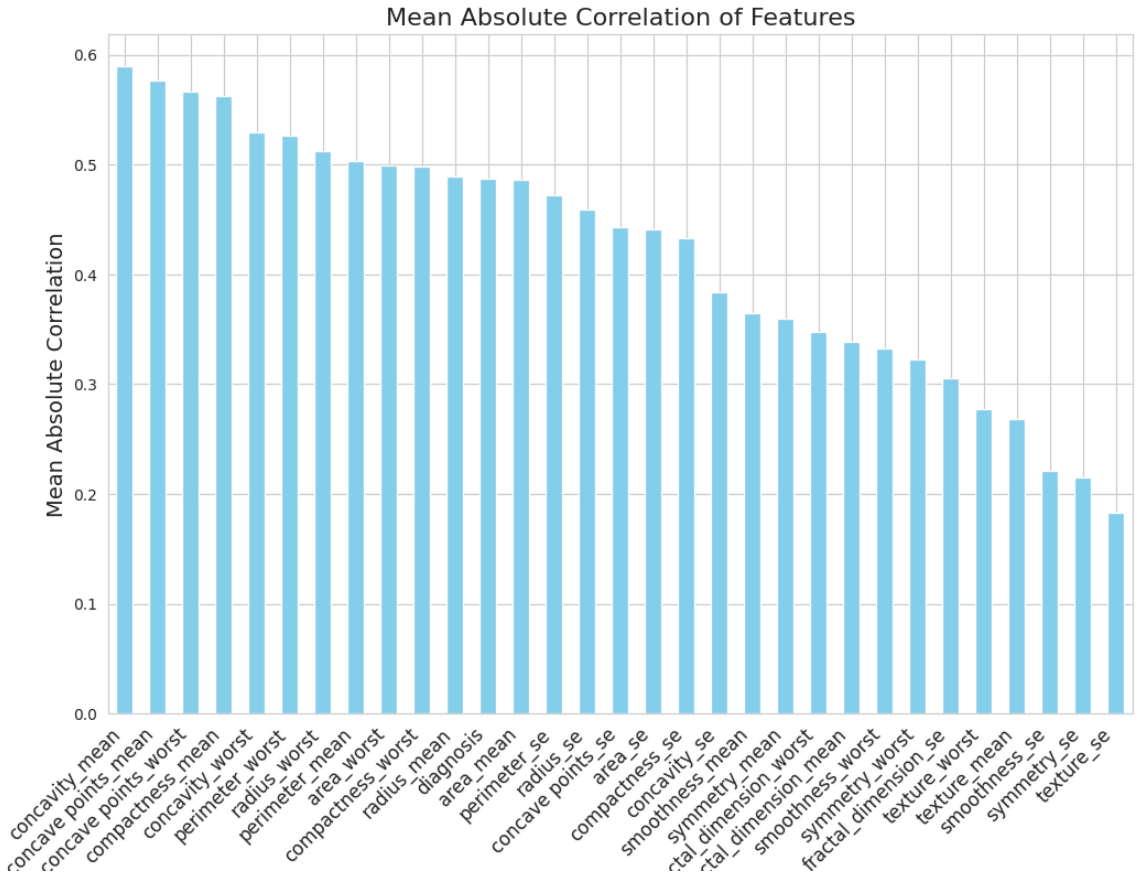
```
plt.figure(figsize=(30,20))
cor = df.corr()
sns.heatmap(cor, annot=True, cmap=plt.cm.Reds)
plt.show()
```

Shreya Bhattacharjee
6861
Machine Learning Project

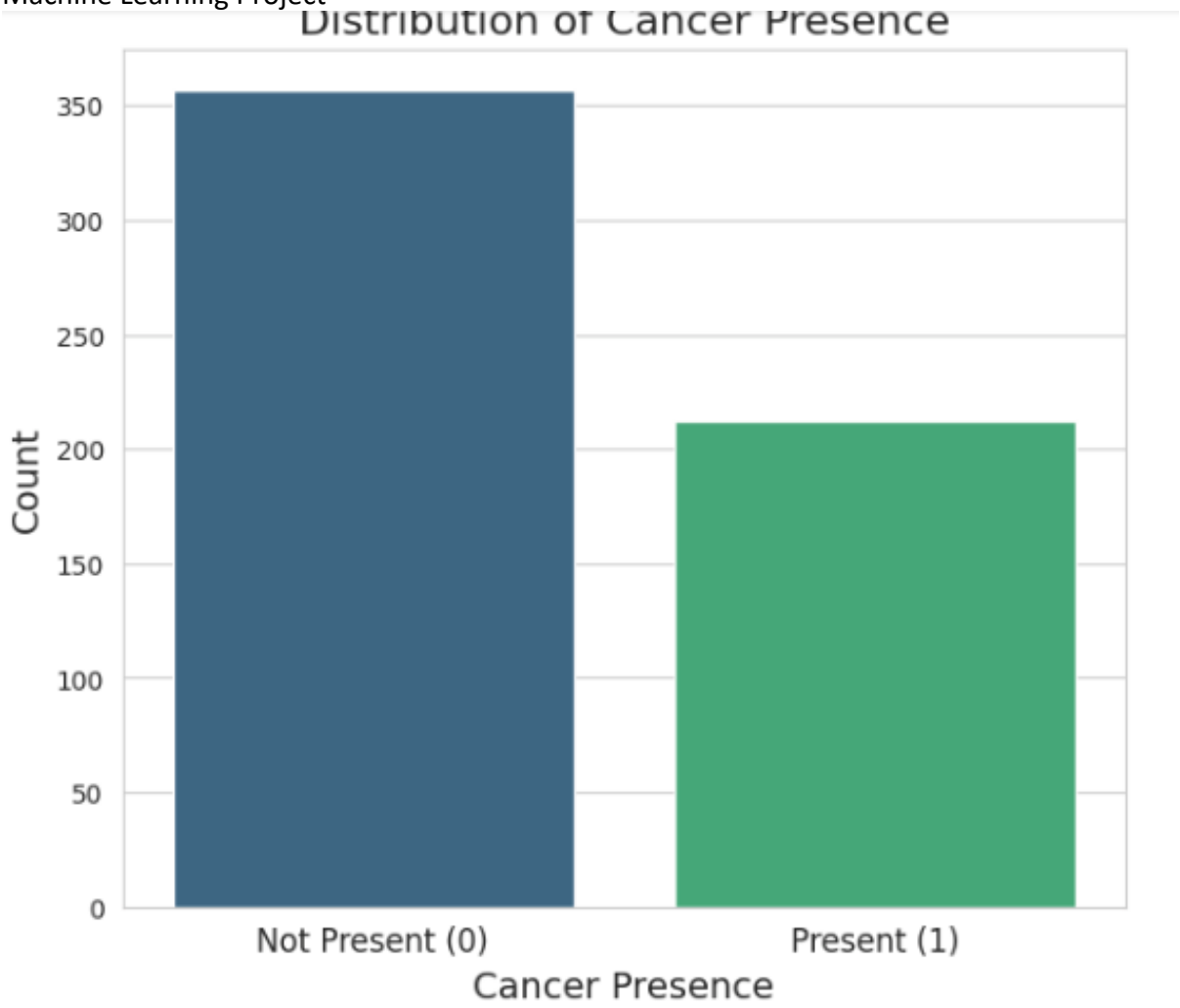


```
# Additional Visualization: Bar plot of mean
correlations
mean_corr =
cor.abs().mean().sort_values(ascending=False)
plt.figure(figsize=(12, 8))
mean_corr.plot(kind='bar', color='skyblue')
plt.title('Mean Absolute Correlation of Features',
fontsize=16)
plt.xlabel('Features', fontsize=14)
plt.ylabel('Mean Absolute Correlation', fontsize=14)
plt.xticks(rotation=45, ha='right', fontsize=12)
plt.show()
```

Shreya Bhattacharjee
6861
Machine Learning Project

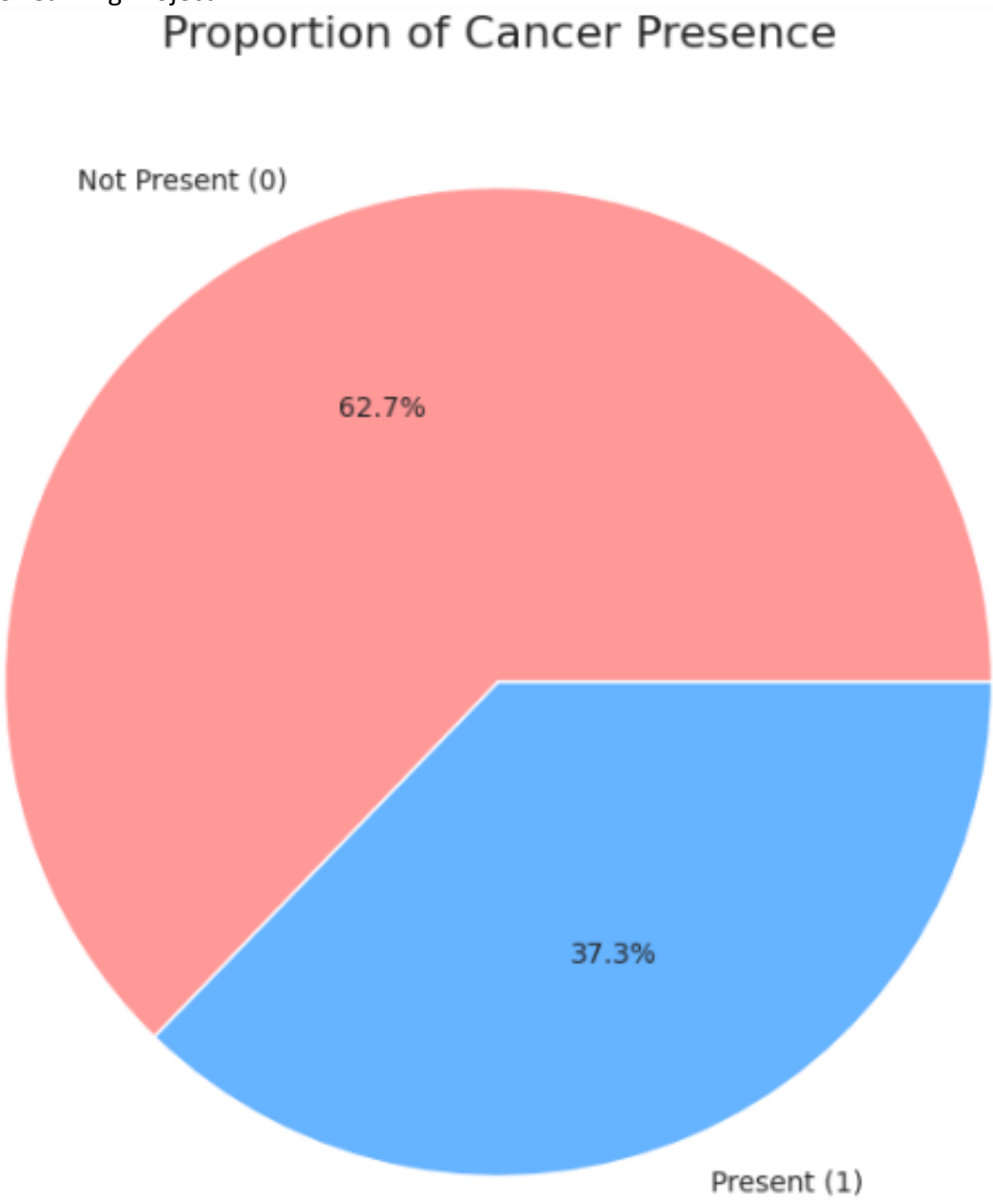



```
plt.figure(figsize=(10,6))
sns.countplot(x='diagnosis', data=df,
palette='viridis')
plt.title('Distribution of Cancer Presence',
fontsize=16)
plt.xlabel('Cancer Presence', fontsize=14)
plt.ylabel('Count', fontsize=14)
plt.xticks(ticks=[0, 1], labels=['Not Present (0)',
'Present (1)'], fontsize=12)
plt.show()
```



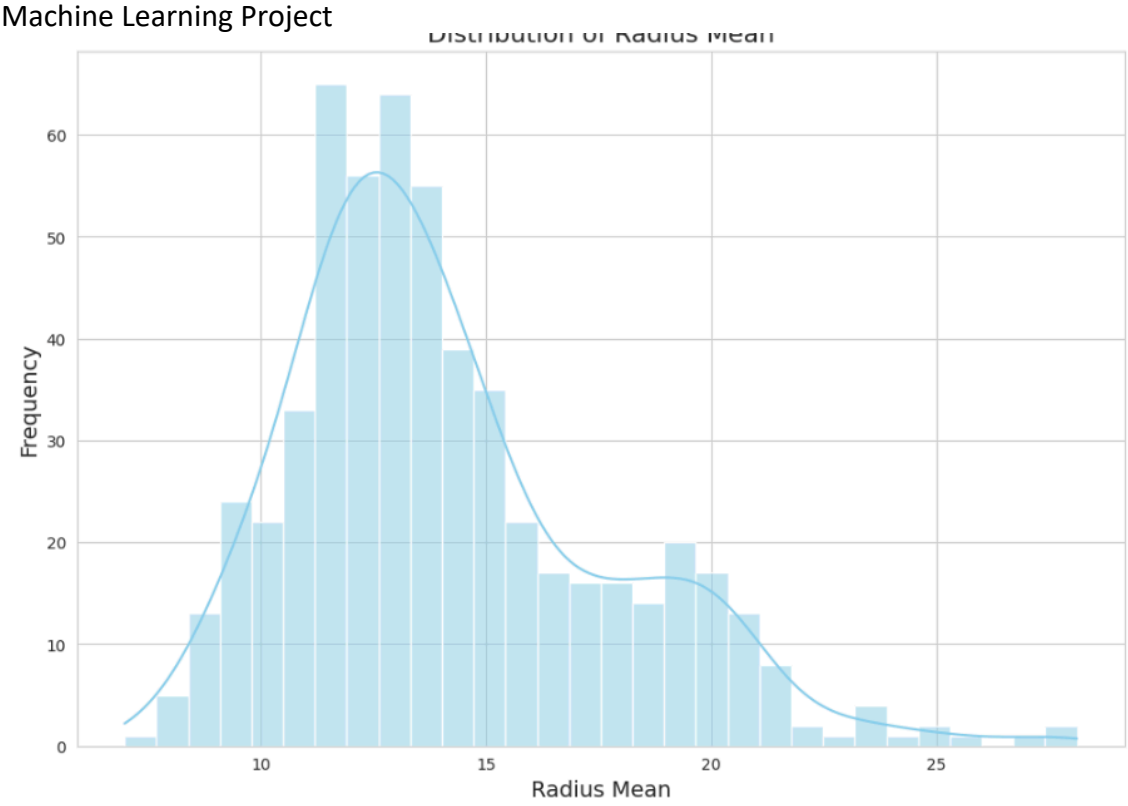

```
plt.figure(figsize=(8,8))
cancer_counts = df['diagnosis'].value_counts()
plt.pie(cancer_counts, labels=['Not Present (0)',
'Present (1)'], autopct='%1.1f%%',
colors=['#ff9999','#66b3ff'])
plt.title('Proportion of Cancer Presence', fontsize=16)
plt.show()
```

Shreya Bhattacharjee
6861
Machine Learning Project



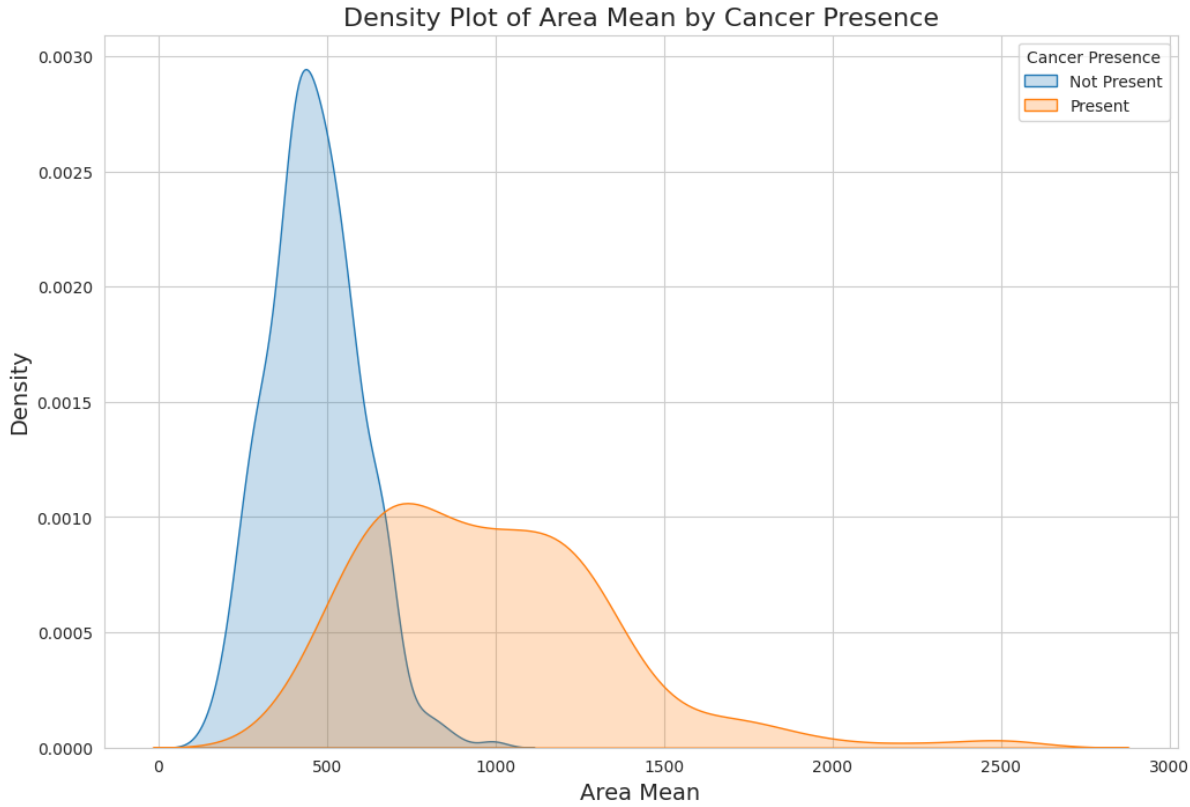

```
plt.figure(figsize=(12,8))
sns.histplot(df['radius_mean'], bins=30, kde=True,
color='skyblue')
plt.title('Distribution of Radius Mean', fontsize=16)
plt.xlabel('Radius Mean', fontsize=14)
plt.ylabel('Frequency', fontsize=14)
plt.show()
```

Shreya Bhattacharjee
6861
Machine Learning Project



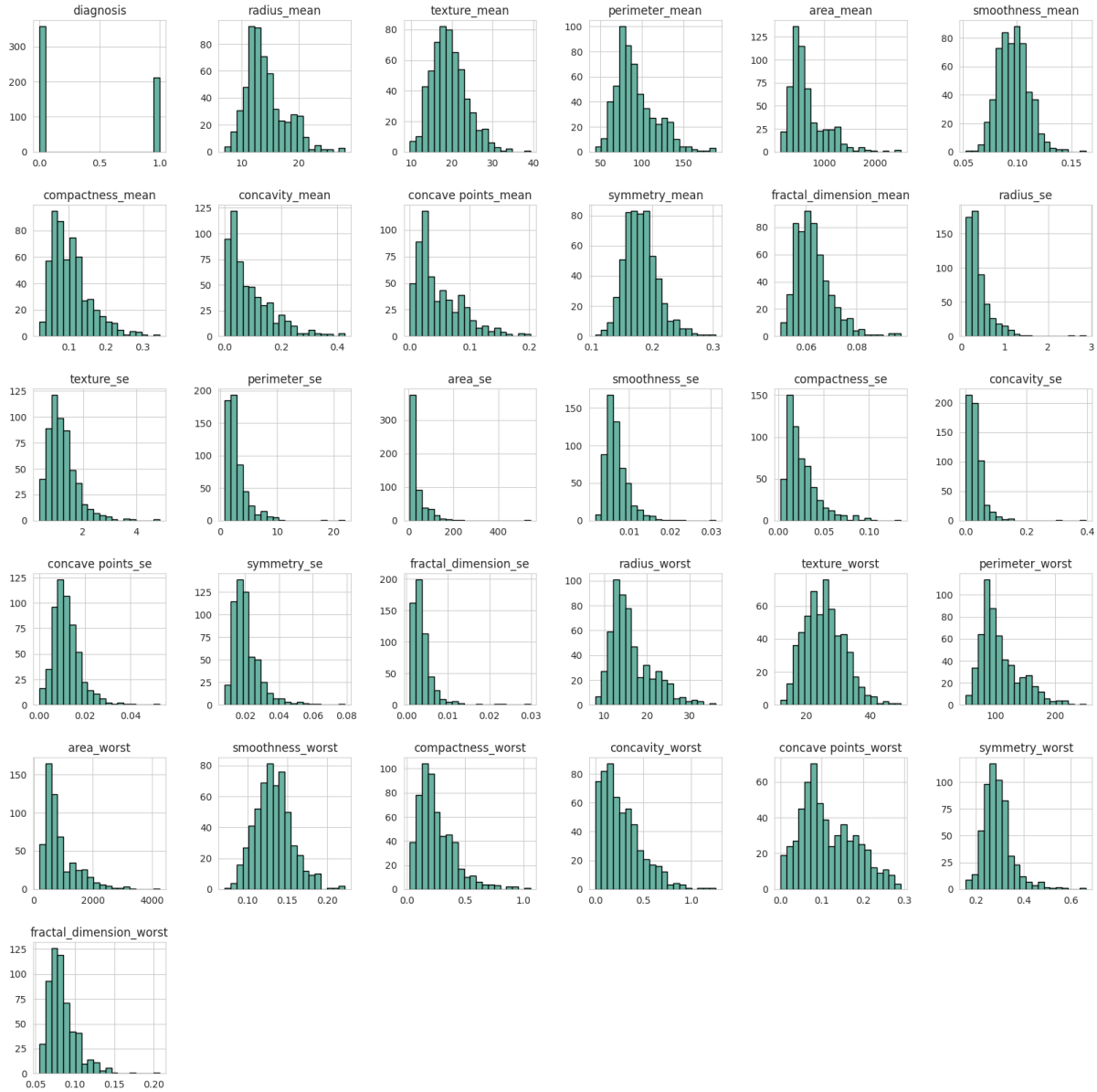

```
plt.figure(figsize=(12,8))
sns.kdeplot(df[df['diagnosis'] == 0]['area_mean'],
label='Not Present', shade=True)
sns.kdeplot(df[df['diagnosis'] == 1]['area_mean'],
label='Present', shade=True)
plt.title('Density Plot of Area Mean by Cancer
Presence', fontsize=16)
plt.xlabel('Area Mean', fontsize=14)
plt.ylabel('Density', fontsize=14)
plt.legend(title='Cancer Presence')
plt.show()
```

Shreya Bhattacharjee
6861
Machine Learning Project




```
import seaborn as sns
import matplotlib.pyplot as plt

# Set Seaborn style and color palette
sns.set_style('whitegrid')  # Change grid style
(lighter background)
custom_palette = sns.color_palette("coolwarm",
as_cmap=True)  # Custom color palette

# Create histograms for each variable
df.hist(figsize=(20, 20), color='#69b3a2',
edgecolor='black', bins=20)

# Add spacing between subplots
plt.subplots_adjust(hspace=0.4, wspace=0.4)

# Display the plot
plt.show()
```

Shreya Bhattacharjee
6861
Machine Learning Project




```
#Correlation with output variable
cor_target = abs(cor["diagnosis"])
#Selecting highly correlated features
relevant_features = cor_target[cor_target>0.7]
relevant_features
```

Shreya Bhattacharjee
6861
Machine Learning Project

	diagnosis
diagnosis	1.000000
radius_mean	0.730029
perimeter_mean	0.742636
area_mean	0.708984
concave points_mean	0.776614
radius_worst	0.776454
perimeter_worst	0.782914
area_worst	0.733825
concave points_worst	0.793566

dtype: float64

Train and Test Model

```
#Split the data into predictor variables and target
variable, following by breaking them into train and
test sets.

Y = df['diagnosis'].values
X = df.drop('diagnosis', axis=1).values

X_train, X_test, Y_train, Y_test = train_test_split (X,
Y, test_size = 0.20, random_state=21)
```

Model Selection

```
# Testing Options
scoring = 'accuracy'
```

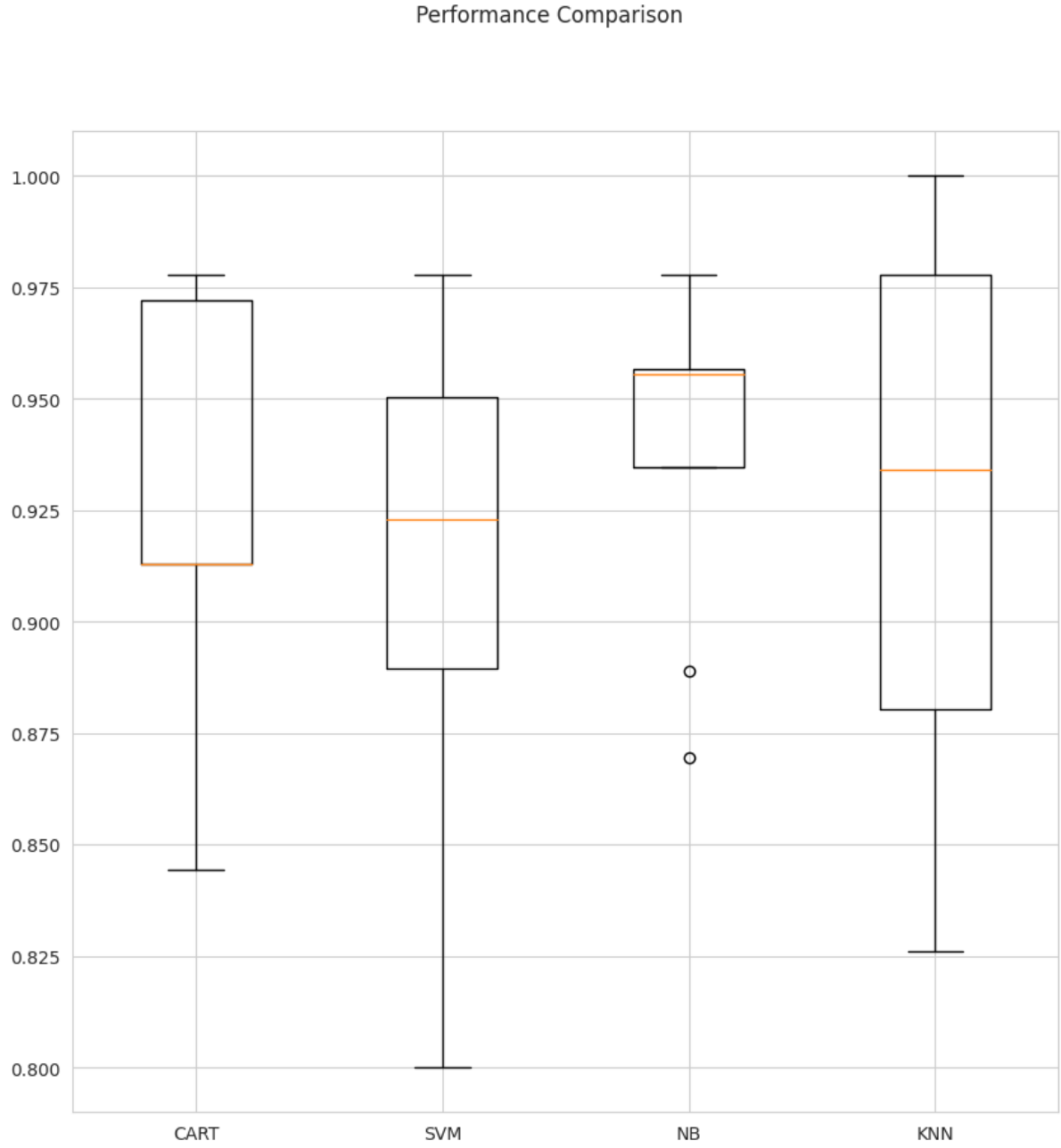
```
# Define models to train
models= []
```

```
models.append(('CART', DecisionTreeClassifier()))
models.append(('SVM', SVC()))
models.append(('NB', GaussianNB()))
models.append(('KNN', KNeighborsClassifier()))

# evaluate each model in turn
results = []
names = []

for name, model in models:
    kfold = KFold(n_splits=10)
    cv_results = cross_val_score(model, X_train,
Y_train, cv=kfold, scoring=scoring)
    results.append(cv_results)
    names.append(name)
    msg = "For %s Model:Mean accuracy is %f (Std
accuracy is %f)" % (name, cv_results.mean(),
cv_results.std())
    print(msg)
```

```
For CART Model:Mean accuracy is 0.925507 (Std accuracy is 0.044070)
For SVM Model:Mean accuracy is 0.907681 (Std accuracy is 0.054723)
For NB Model:Mean accuracy is 0.940773 (Std accuracy is 0.033921)
For KNN Model:Mean accuracy is 0.927729 (Std accuracy is 0.055250)
```

```
fig = plt.figure(figsize=(10,10))
fig.suptitle('Performance Comparison')
ax = fig.add_subplot(111)
plt.boxplot(results)
ax.set_xticklabels(names)
plt.show()
```

Shreya Bhattacharjee
6861
Machine Learning Project



Evaluation of algorithm on Standardised Data

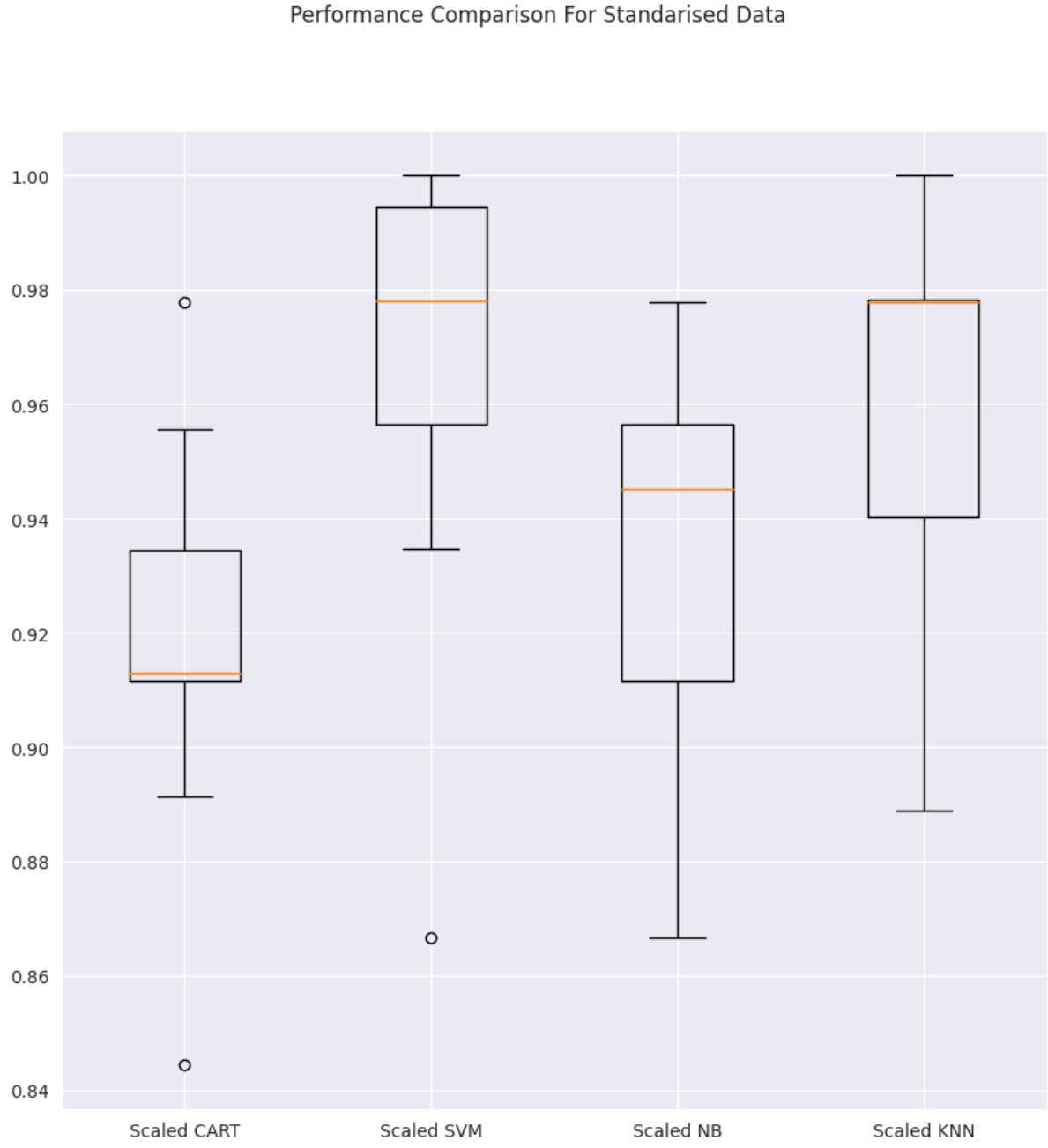
```
# Standardize the dataset
import warnings
pipelines = []

pipelines.append(('Scaled CART', Pipeline([('Scaler',
StandardScaler()),('CART',
DecisionTreeClassifier())])))
```

```
pipelines.append(('Scaled SVM', Pipeline([('Scaler',
StandardScaler()),('SVM', SVC( ))])))
pipelines.append(('Scaled NB', Pipeline([('Scaler',
StandardScaler()),('NB', GaussianNB())])))
pipelines.append(('Scaled KNN', Pipeline([('Scaler',
StandardScaler()),('KNN', KNeighborsClassifier())])))

results = []
names = []

kfold = KFold(n_splits= 10)
for name, model in pipelines:
  cv_results = cross_val_score(model, X_train, Y_train,
cv=kfold, scoring='accuracy')
  results.append(cv_results)
  names.append(name)
  print( "For %s Model: Mean Accuracy is %f (Std
Accuracy is %f)" % (name, cv_results.mean(),
cv_results.std()))
```

```
For Scaled CART Model: Mean Accuracy is 0.918744 (Std Accuracy is 0.034263)
For Scaled SVM Model: Mean Accuracy is 0.964879 (Std Accuracy is 0.038621)
For Scaled NB Model: Mean Accuracy is 0.931932 (Std Accuracy is 0.038625)
For Scaled KNN Model: Mean Accuracy is 0.958357 (Std Accuracy is 0.038595)
```

```
fig = plt.figure(figsize=(10,10))
fig.suptitle('Performance Comparison For Standarised
Data')
ax = fig.add_subplot(111)
plt.boxplot(results)
ax.set_xticklabels(names)
plt.show()
```

Shreya Bhattacharjee
6861
Machine Learning Project




```
# Make predictions on validation dataset

for name, model in models:
    scaler = StandardScaler().fit(X_train)
    X_train_scaled = scaler.transform(X_train)
    model.fit(X_train_scaled, Y_train)
    X_test_scaled = scaler.transform(X_test)
    predictions = model.predict(X_test_scaled)
    print("\nModel:",name)
```

```
    print("Accuracy score:" % accuracy_score(Y_test,
predictions))
    print("Classification
report:\n",classification_report(Y_test, predictions))
    print("Confusion
Matrix:\n",confusion_matrix(Y_test, predictions))


# Accuracy - ratio of correctly predicted observation
to the total observations.
# Precision - (false positives) ratio of correctly
predicted positive observations to the total predicted
positive observations
# Recall (Sensitivity) - (false negatives) ratio of
correctly predicted positive observations to the all
observations in actual class - yes.
# F1 score - F1 Score is the weighted average of
Precision and Recall. Therefore, this score takes both
false positives and false
```

Shreya Bhattacharjee
6861
Machine Learning Project

```
Model: CART
Accuracy score:
Classification report:
              precision    recall  f1-score   support

         0.0       0.97      0.95      0.96        75
         1.0       0.90      0.95      0.92        39

    accuracy                           0.95       114
   macro avg       0.94      0.95      0.94       114
weighted avg       0.95      0.95      0.95       114

Confusion Matrix:
 [[71  4]
 [ 2 37]]

Model: SVM
Accuracy score:
Classification report:
              precision    recall  f1-score   support

         0.0       0.99      0.99      0.99        75
         1.0       0.97      0.97      0.97        39

    accuracy                           0.98       114
   macro avg       0.98      0.98      0.98       114
weighted avg       0.98      0.98      0.98       114

Confusion Matrix:
 [[74  1]
 [ 1 38]]

Model: NB
Accuracy score:
Classification report:
              precision    recall  f1-score   support

         0.0       0.95      0.96      0.95        75
         1.0       0.92      0.90      0.91        39

    accuracy                           0.94       114
   macro avg       0.93      0.93      0.93       114
weighted avg       0.94      0.94      0.94       114

Confusion Matrix:
 [[72  3]
 [ 4 35]]
```

Shreya Bhattacharjee
6861
Machine Learning Project

```
Model: KNN
Accuracy score:
Classification report:
              precision    recall  f1-score   support

         0.0       0.97      1.00      0.99        75
         1.0       1.00      0.95      0.97        39

    accuracy                           0.98       114
   macro avg       0.99      0.97      0.98       114
weighted avg       0.98      0.98      0.98       114

Confusion Matrix:
 [[75  0]
 [ 2 37]]
```

App.py
Flask code

```
import numpy as np
import pandas as pd
from flask import Flask, request, render_template
import pickle

app = Flask(__name__)
model = pickle.load(open('model.pkl', 'rb'))

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

@app.route('/predict',methods=['POST'])
def predict():
  input_features = [int(x) for x in request.form.values()]
  features_value = [np.array(input_features)]

  features_name = ['clump_thickness', 'uniform_cell_size',
'uniform_cell_shape',
       'marginal_adhesion', 'single_epithelial_size', 'bare_nuclei',
       'bland_chromatin', 'normal_nucleoli', 'mitoses']

  df = pd.DataFrame(features_value, columns=features_name)
  output = model.predict(df)

  if output == 4:
      res_val = "Breast cancer"
  else:
      res_val = "no Breast cancer"


  return render_template('index.html', prediction_text='Patient has
```

```
{}'.format(res_val))

if __name__ == "__main__":
  app.run()
```

index.html

```
<!--GUI for Breast Cancer Detection Application using SVM-->
<!DOCTYPE html>
<html>

<head>
  <meta charset="UTF-8">
  <!-- Bootstrap CSS -->
  <link rel="stylesheet"
href="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/css/bootstrap.min.
css" integrity="sha384-
ggOyR0iXCbMQv3Xipma34MD+dH/1fQ784/j6cY/iJTQUOhcWr7x9JvoRxT2MZw1T"
crossorigin="anonymous">
  <title>Breast Cancer Detection</title>
  <style>
    /*just bg and body style*/
    body {
      margin: 40px;
      background-color: #808080;
      background-image: linear-gradient(315deg, #de5499 19%, #a0c5ba 85%);
    }

    .container {
      border-radius: 5px;
      text-align: center;
    }

    .btn-container {
      background: white;
      box-shadow: 0 19px 38px rgba(0, 0, 0, 0.30), 0 15px 12px rgba(0, 0,
0, 0.22);
      border-radius: 5px;
      padding: 10px;
    }

    .head {
      font-weight: bolder;
    }

    .btn-primary {
      border-color: #ff33f !important;
      color: #ffffff;
      text-shadow: 0 -1px 0 rgba(0, 0, 0, 0.25);
      background-color: #de5499 !important;
      border-color: #de5499 !important;
      padding: 5px;
    }

    label {
```

```
      width: 50%;
    }

    #predict {
      display: none;
    }

    .form-group {
      padding: 2px;
    }
  </style>
  <!--Font Awesome-->
  <script src="https://kit.fontawesome.com/a076d05399.js"></script>
  <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/font-
awesome/4.7.0/css/font-awesome.min.css" integrity="sha384-
wvfXpqpZZVQGK6TAh5PVlGOfQNHSoD2xbE+QkPxCAFlNEevoEH3Sl0sibVcOQVnN"
crossorigin="anonymous">
  <script
src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></sc
ript>
  <!-- jQuery first, then Popper.js, then Bootstrap JS -->
  <script
src="https://cdnjs.cloudflare.com/ajax/libs/jquery/3.5.1/jquery.min.js"
integrity="sha512-
bLT0Qm9VnAYZDflyKcBaQ2gg0hSYNQrJ8RilYldYQ1FxQYoCLtUjuuRuZo+fjqhx/qtq/1itJ0C
2ejDxltZVFg==" crossorigin="anonymous"></script>
  <script src="https://code.jquery.com/jquery-3.5.1.slim.min.js"
integrity="sha384-
DfXdz2htPH0lsSSs5nCTpuj/zy4C+OGpamoFVy38MVBnE+IbbVYUew+OrCXaRkfj"
crossorigin="anonymous"></script>
  <script
src="https://cdn.jsdelivr.net/npm/popper.js@1.16.0/dist/umd/popper.min.js"
integrity="sha384-
Q6E9RHvbIyZFJoft+2mJbHaEWldlvI9IOYy5n3zV9zzTtmI3UksdQRVvoxMfooAo"
crossorigin="anonymous"></script>
  <script
src="https://stackpath.bootstrapcdn.com/bootstrap/4.5.0/js/bootstrap.min.js
" integrity="sha384-
OgVRvuATP1z7JjHLkuOU7Xw704+h835Lr+6QL9UvYjZE3Ipu6Tp75j7Bh/kR0JKI"
crossorigin="anonymous"></script>
</head>

<body>
  <div class="container">
    <div class="row">
      <div class="col-md-12">
        <h1 class="head">Breast Cancer Detection</h1>
      </div>
    </div>
    <div class="row">
      <div class="col-md-12">
        <div class="btn-container">

          <!-- Main Input For Receiving Query to our ML -->
          <form action="{{ url_for('predict')}}" method="post" class="form-
inline">
            <div class="row">
              <div class="col-md-6">
                <img class="image" src="{{url_for('static',
filename='img.jpeg')}}" width="100%" height="100%" />
```

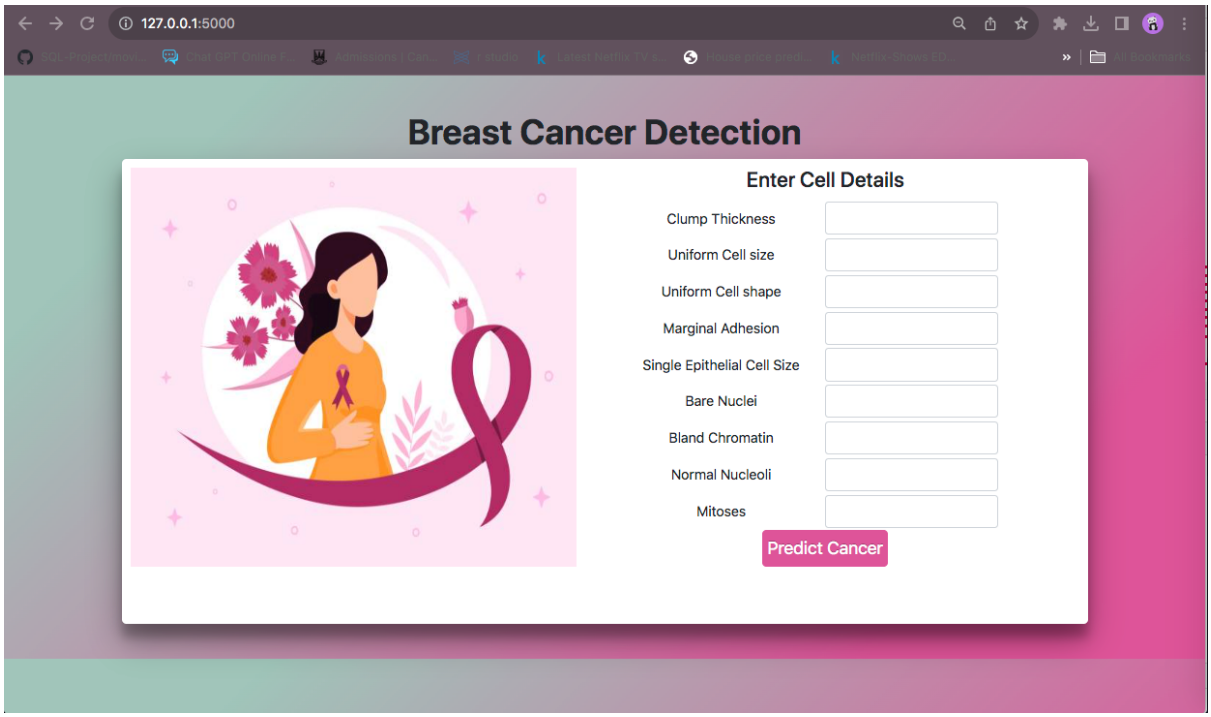
```
                </div>
                <div class="col-md-6">
                  <div class="container">
                    <h4>Enter Cell Details</h4>
                    <div class="form-group">
                      <label for="clump_thickness">Clump Thickness </label>
                      <input type="text" class="form-control"
name="clump_thickness" required="required">
                    </div>
                    <div class="form-group">
                      <label for="uniform_cell_size">Uniform Cell
size</label>
                      <input type="text" class="form-control"
name="uniform_cell_size" required="required">
                    </div>
                    <div class="form-group">
                      <label for="uniform_cell_shape">Uniform Cell
shape</label>
                      <input type="text" class="form-control"
name="uniform_cell_shape" required="required" />
                    </div>
                    <div class="form-group">
                      <label for="marginal_adhesion">Marginal
Adhesion</label>
                      <input type="text" class="form-control"
name="marginal_adhesion" required="required" />
                    </div>
                    <div class="form-group">
                      <label for="single_epithelial_size">Single Epithelial
Cell Size</label>
                      <input type="text" class="form-control"
name="single_epithelial_size" required="required" />
                    </div>
                    <div class="form-group">
                      <label for="bare_nuclei">Bare Nuclei</label>
                      <input type="text" class="form-control"
name="bare_nuclei" required="required" />
                    </div>
                    <div class="form-group">
                      <label for="bland_chromatin">Bland Chromatin</label>
                      <input type="text" class="form-control"
name="bland_chromatin" required="required" />
                    </div>
                    <div class="form-group">
                      <label for="normal_nucleoli">Normal Nucleoli</label>
                      <input type="text" class="form-control"
name="normal_nucleoli" required="required" />
                    </div>

                    <div class="form-group">
                      <label for="mitoses">Mitoses</label>
                      <input type="text" class="form-control" name="mitoses"
required="required" />
                    </div>
                    <button type="submit" class="btn btn-primary btn-
lg">Predict Cancer</button>
                  </div>
                </div>
              </div>
            </form>
```

Shreya Bhattacharjee
6861
Machine Learning Project

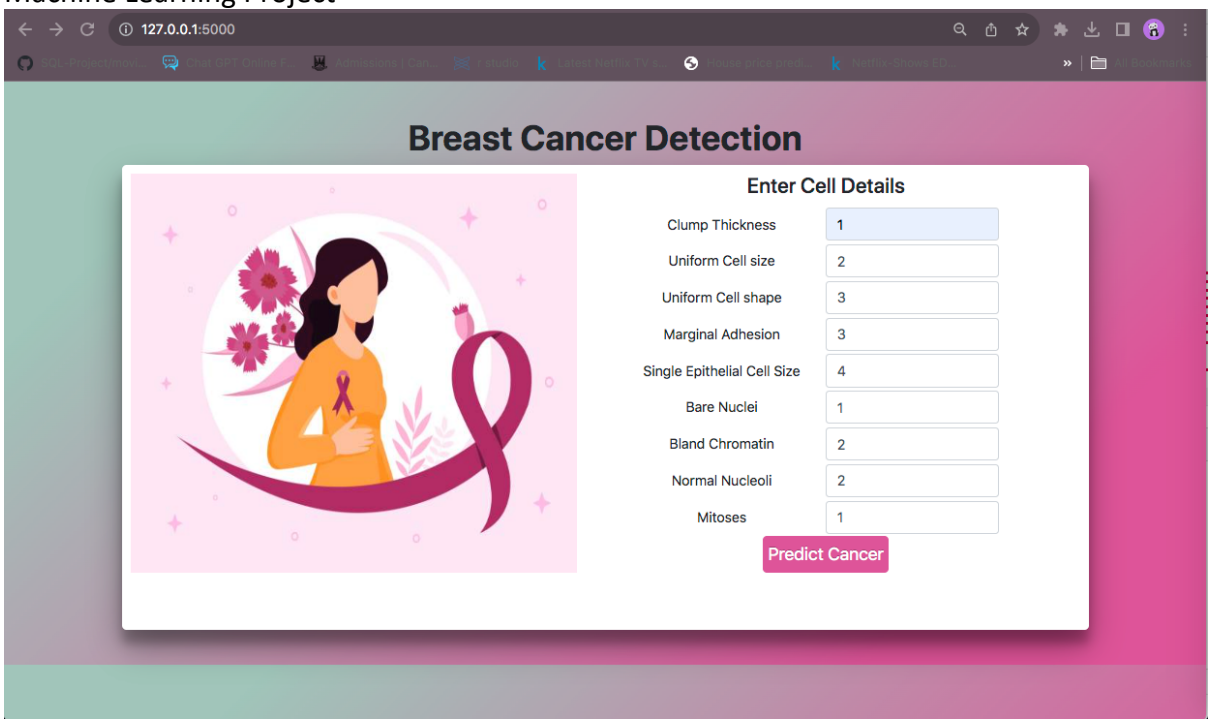
```
        <br />
        <center>
          <h1 style="background:#de5499">{{prediction_text}}</h1>
        </center>
        <br />
</body>

</html>
```

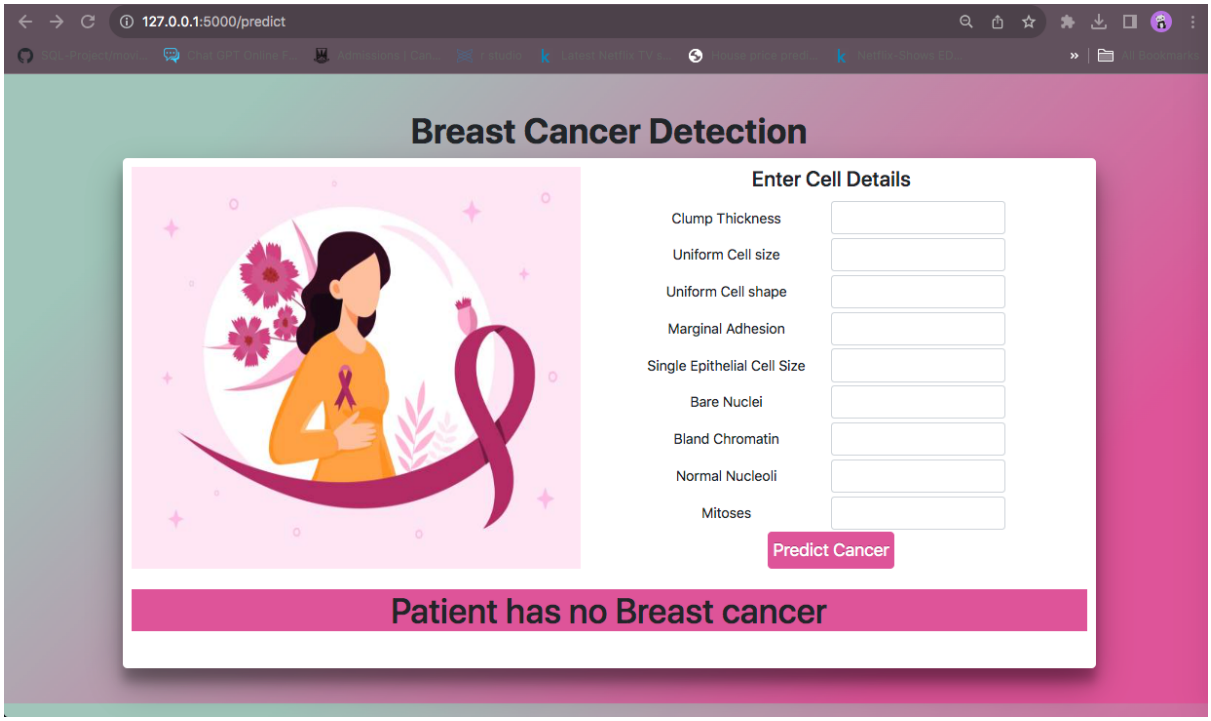
Output

```
 * Serving Flask app 'app'
 * Debug mode: off
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Running on http://127.0.0.1:5000
Press CTRL+C to quit
127.0.0.1 - - [18/Sep/2024 21:24:40] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [18/Sep/2024 21:24:40] "GET /static/img.jpeg HTTP/1.1" 304 -
/Users/shreyabhattacharjee/PycharmProjects/Breast Cancer Prediction/venv/lib/python3.10/site-packages/sklearn/base.py:486: UserWarning: X has feature
 names, but SVC was fitted without feature names
```

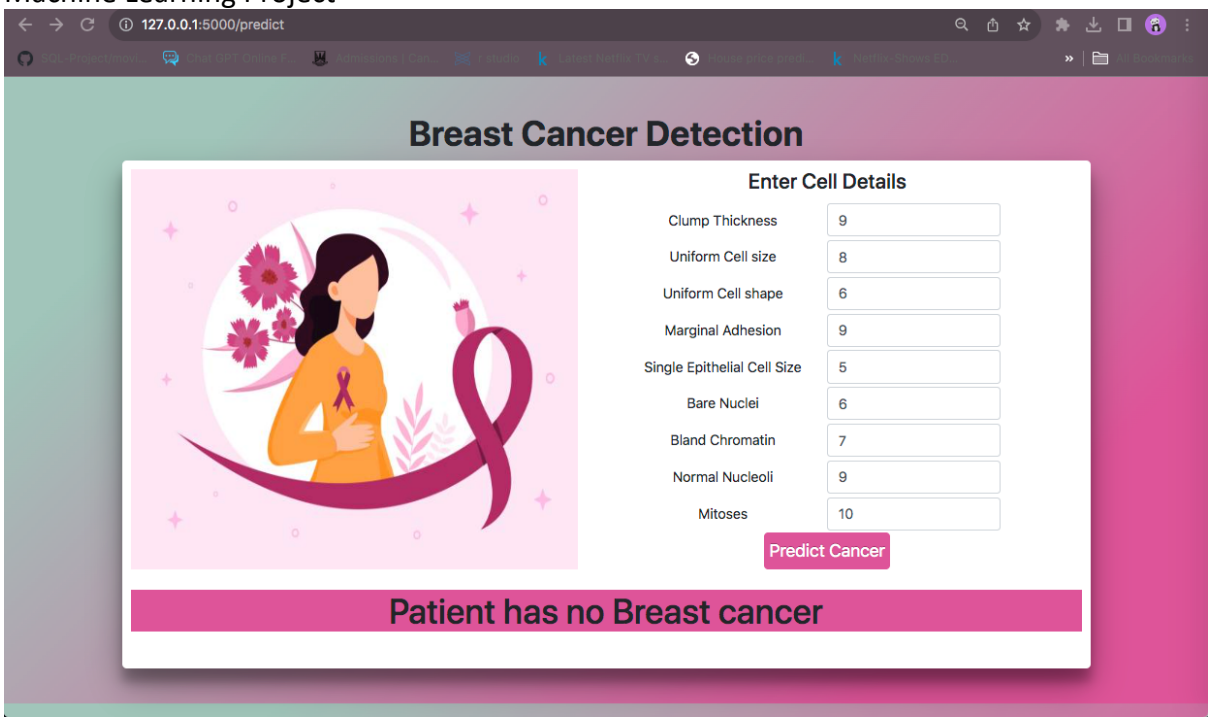


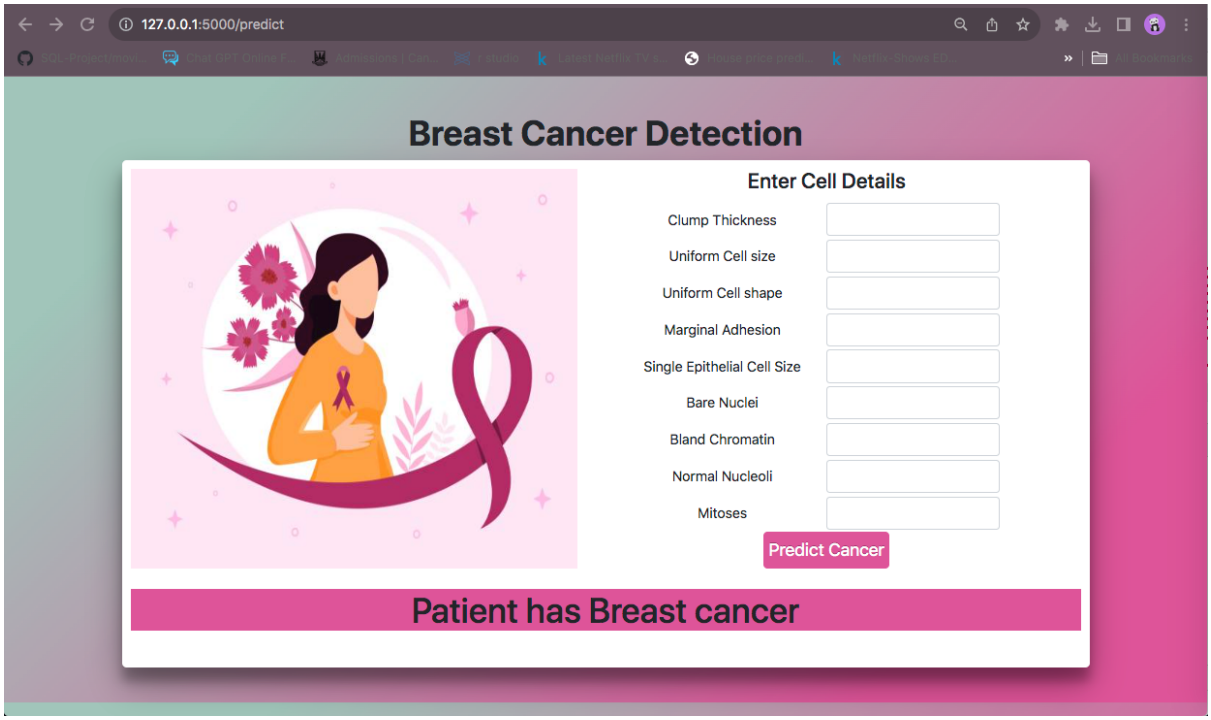
Shreya Bhattacharjee
6861
Machine Learning Project





Shreya Bhattacharjee
6861
Machine Learning Project





Shreya Bhattacharjee
6861
Machine Learning Project

Conclusion

In this project, we analyzed a dataset containing various features related to cancer detection, revealing significant insights into the distribution and relationships of these features. Visualizations such as histograms, box plots, and scatter plots showed that features like 'radius_mean', 'texture_mean', and 'area_mean' exhibit distinct distributions and correlations between cancer-positive and cancer-negative cases. Notably, 'area_mean' and 'radius_mean' were found to be strongly correlated and useful in distinguishing cancer presence. These findings suggest that these features are valuable for predictive modeling and can aid in developing more accurate diagnostic tools. Future work should focus on building and validating predictive models, incorporating additional data, and using cross-validation to ensure robustness and accuracy in cancer detection.