School of Technology Management and Engineering

A Project Report On

Breast Cancer Classification

Submitted By

Pragnya Reddy G - 70572200054

Submitted To

Prof. Muhammed Niyas

Submission Date

04 April,2024

ACKNOWLEDGEMENT

We are grateful to Prof. Muhammed Niyas, whose guidance, inspiration and constructive suggestions throughout the project has resulted in a successful completion of this project. Without their willing disposition, cooperation this project could not have been completed in due time.

Date:..... .....

Cse - 4th semester

Roll No: L055

CERTIFICATE OF ACCEPTANCE

The report of the Project titled “Breast Cancer Detection” submitted by **G Pragnya Reddy** (L055) of **CSE 4th Semester of 2024** is hereby recommended to be accepted for fulfilment of the semester 4.

Signature : **Pragnya Reddy**

Breast Cancer Classification

Content

PROBLEM STATEMENT

In the context of medical diagnosis, particularly in breast cancer detection, the availability of accurate and reliable predictive models is crucial for effective patient care. The problem statement revolves around the analysis of a dataset comprising diverse features extracted from breast cancer biopsies. These features encompass various aspects such as the size, shape, and texture of cell nuclei, among others, which are indicative of the tumor's characteristics.

The primary objective is to develop a robust machine learning model capable of effectively differentiating between two classes of tumors: malignant (denoted as M) and benign (denoted as B). Malignant tumors are cancerous and have the potential to spread to other parts of the body, posing a significant health risk to the patient. On the other hand, benign tumors are non-cancerous and generally do not pose a severe threat to health.

Given this dataset, the goal is to train a machine learning model that can accurately predict the nature of a tumor (M or B) based on the provided features. Such a model would assist healthcare professionals in making more informed decisions regarding patient diagnosis and treatment plans, ultimately leading to improved patient outcomes and quality of care.

ABSTRACT

Breast cancer is a prevalent and potentially deadly disease affecting millions of individuals worldwide. Early detection and accurate diagnosis are crucial for effective treatment and improved patient outcomes. In this study, we investigated the application of machine learning algorithms for the classification of breast cancer using clinical data obtained from fine needle aspirate (FNA) samples.

The dataset used in this study consists of various quantitative features extracted from FNA samples, including measures of tumor radius, texture, perimeter, area, smoothness, and other attributes. After preprocessing the data to handle missing values and standardize the features, we explored a variety of classification algorithms, including logistic regression, k-nearest neighbors (KNN), naive Bayes, support vector machines (SVM), decision trees, random forests, and gradient boosting classifiers.

INTRODUCTION

Breast cancer is a pervasive health concern globally, with significant implications for patient outcomes and healthcare systems. Timely and accurate diagnosis of breast cancer is crucial for effective treatment planning and improved survival rates. In this project, we focus on leveraging machine learning techniques to develop predictive models for breast cancer classification.

The dataset utilized in this study comprises a diverse set of features extracted from breast cancer biopsies. These features encapsulate various morphological and textural characteristics observed in histopathological images of cell nuclei. By harnessing the power of machine learning algorithms, our objective is to create robust models capable of distinguishing between malignant and benign tumors based on these distinctive features.

The primary aim of this project is to contribute to the field of medical diagnostics by providing clinicians with reliable tools for automated breast cancer classification. By automating the classification process, healthcare professionals can expedite diagnosis, optimize treatment strategies, and ultimately improve patient outcomes. Additionally, our work seeks to highlight the potential of machine learning in augmenting traditional diagnostic approaches and advancing personalized medicine in the field of oncology.

Through this project, we aim to underscore the significance of interdisciplinary collaboration between computer science and medicine in addressing complex healthcare challenges. By bridging the gap between technology and clinical practice, we aspire to make meaningful contributions to the ongoing efforts in breast cancer research and patient care.

DATA COLLECTION

The dataset used in this project was sourced from Kaggle, a renowned platform for datasets and data science competitions. This dataset comprises various features extracted from breast cancer biopsies, aimed at predicting the malignancy of tumors. Before delving into the analysis, it's imperative to understand the dataset's characteristics and relevance to the problem statement.

Upon acquiring the dataset, it's essential to explore its contents thoroughly. This includes understanding its dimensions, such as the number of instances (rows) and features (columns), and comprehending the meaning of each feature in the context of breast cancer diagnosis. Additionally, assessing the dataset's quality is crucial. This involves identifying any missing values, outliers, or inconsistencies that could affect the analysis and modeling process.

DATA PREPROCESSING

The next phase involves preparing the dataset for analysis through preprocessing steps aimed at improving its quality and suitability for modeling.

Missing values are a common issue in real-world datasets and must be addressed appropriately. This may involve imputing missing values using techniques like mean or median imputation or removing instances with missing data, depending on the dataset's characteristics and the impact of missing values on the analysis.

Outliers, or data points significantly different from the rest of the dataset, can skew analysis results and model performance. Identifying and handling outliers is crucial to ensure the robustness and reliability of the analysis. Techniques such as visualization and statistical methods can aid in outlier detection and determination of appropriate treatment strategies.

Feature selection and engineering play a pivotal role in determining the predictive power of the model. By selecting relevant features and creating new ones through transformation or combination of existing features, the model's performance can be enhanced. This process often involves leveraging domain knowledge and statistical techniques to identify the most informative features for the predictive task at hand.

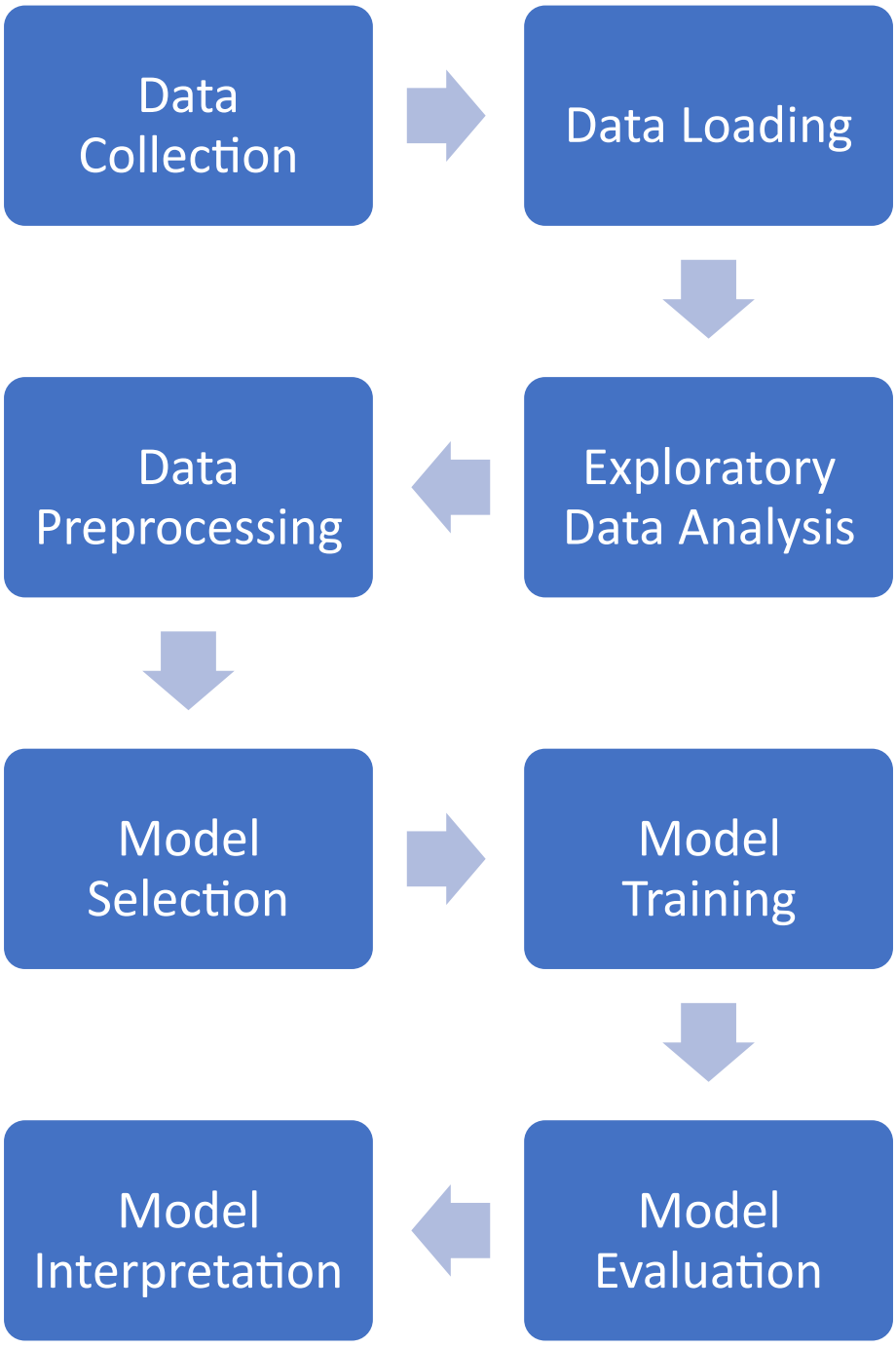
Normalization or scaling of numerical features ensures that they have a consistent scale and distribution, which is essential for many machine learning algorithms to perform effectively. Techniques such as min-max scaling or standardization are commonly employed for this purpose.

Categorical variables, if present in the dataset, need to be encoded into numerical format for modeling. Techniques like one-hot encoding or label encoding are commonly used to achieve this while preserving the semantics of the original categorical variables.

Finally, splitting the dataset into training and testing sets allows for the evaluation of model performance on unseen data. This train-test split ensures that the model's performance estimates are reliable and can generalize to new, unseen data.

By meticulously conducting these preprocessing steps, the dataset is prepared for subsequent analysis and modeling, laying the foundation for building robust and accurate predictive models for breast cancer diagnosis.

METHODOLOGY



CODE

```
import pandas as pd import numpy as np import matplotlib.pyplot as
plt import seaborn as sns from sklearn.model_selection import
train_test_split, cross_val_score
from sklearn.preprocessing import StandardScaler, LabelEncoder,
OneHotEncoder from sklearn.impute import SimpleImputer from
sklearn.compose import ColumnTransformer from
sklearn.pipeline import Pipeline
from sklearn.linear_model import LinearRegression, Ridge, Lasso,
LogisticRegression from sklearn.neighbors import KNeighborsRegressor,
KNeighborsClassifier from sklearn.svm import SVR, SVC from
sklearn.tree import DecisionTreeRegressor, DecisionTreeClassifier
from sklearn.ensemble import RandomForestRegressor,
RandomForestClassifier, GradientBoostingRegressor,
GradientBoostingClassifier
from sklearn.metrics import mean_squared_error, r2_score, confusion_matrix,
classification_report, roc_curve, roc_auc_score from sklearn.decomposition
import PCA from sklearn.cluster import KMeans, DBSCAN from
sklearn.feature_selection import SelectFromModel from sklearn.naive_bayes
import GaussianNB


# Load the dataset data =
pd.read_csv("data.csv") #
Data Exploration
print("Dataset Overview:")
print(data.head())
print("\nDataset Info:")
```

```
print(data.info())
print("\nSummary Statistics:")
print(data.describe())

# Check for missing values
print("\nMissing  Values:")
print(data.isnull().sum())

# Handling Categorical Data: Encoding class labels
label_encoder = LabelEncoder() data['diagnosis'] =
label_encoder.fit_transform(data['diagnosis'])

# Visualize the target variable distribution
plt.figure(figsize=(6, 4))
sns.countplot(x='diagnosis', data=data)
plt.title('Target Variable Distribution')
plt.xlabel('Diagnosis') plt.ylabel('Count')
plt.show()

# Visualize correlations between features plt.figure(figsize=(12,
8)) sns.heatmap(data.corr(), annot=True, cmap='coolwarm',
fmt=".2f", linewidths=0.5) plt.title('Correlation Heatmap')
plt.show()

# Handling Categorical Data: Encoding class labels and Performing One-Hot
```

```
Encoding label_encoder = LabelEncoder() data['diagnosis'] =
label_encoder.fit_transform(data['diagnosis'])

# Separate features and target variable X =
data.drop(columns=["id", "diagnosis"]) y =
data["diagnosis"]

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)

# Define preprocessing steps numeric_features =
X.select_dtypes(include=['int64', 'float64']).columns
numeric_transformer = Pipeline(steps=[
    ('imputer', SimpleImputer(strategy='mean')),
    ('scaler', StandardScaler())
])

preprocessor = ColumnTransformer(
    transformers=[
        ('num', numeric_transformer, numeric_features)
    ])

# Classification Models classification_models
= {
    "Logistic Regression": LogisticRegression(),
```

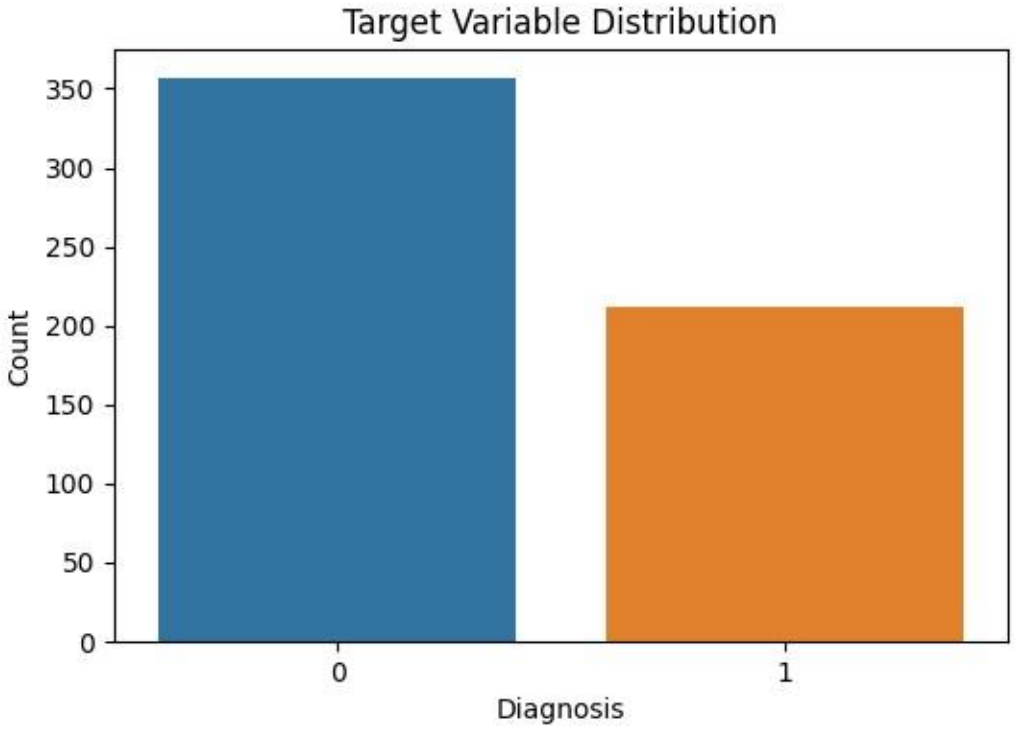
```
    "KNN Classifier": KNeighborsClassifier(),
    "Naive Bayes": GaussianNB(),
    "Support Vector Machine": SVC(),
    "Decision Tree Classifier": DecisionTreeClassifier(),
    "Random Forest Classifier": RandomForestClassifier(),
    "Gradient Boosting Classifier": GradientBoostingClassifier()
}

print("\nClassification Models:") for name,
model in classification_models.items():
    pipeline = Pipeline(steps=[('preprocessor', preprocessor),
                               ('classifier', model)])     pipeline.fit(X_train, y_train)
y_pred = pipeline.predict(X_test)     print(f"Model: {name}")
print("Confusion Matrix:\n", confusion_matrix(y_test, y_pred))
print("Classification Report:\n", classification_report(y_test, y_pred))
    print("------------------------------------------------------")

# Tree-based Algorithms tree_based_models
= {
    "Decision Tree Classifier": DecisionTreeClassifier(),
   "Random Forest Classifier": RandomForestClassifier(),
    "Gradient Boosting Classifier": GradientBoostingClassifier(),
}
```

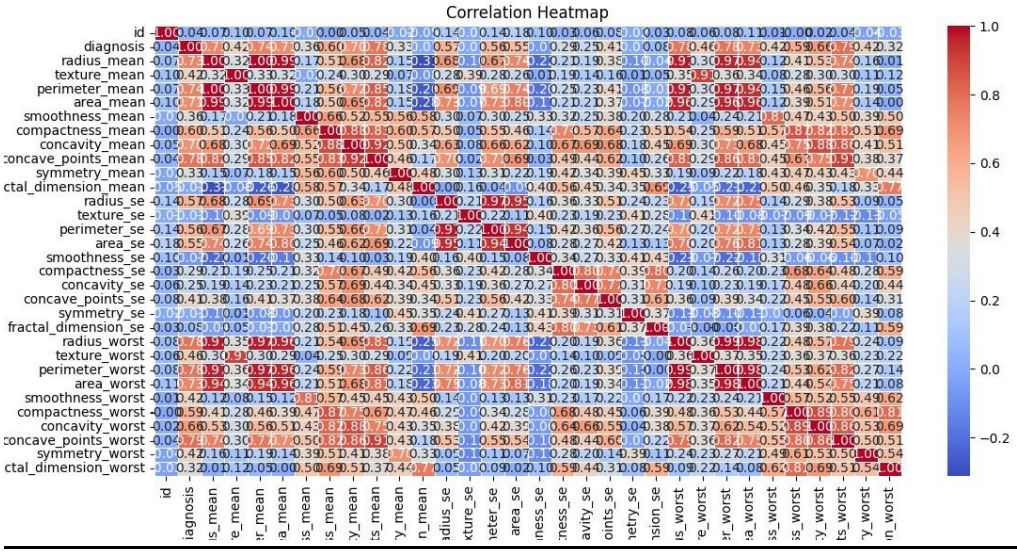
```
print("\nTree-based Models:") for name, model
in tree_based_models.items():
    pipeline = Pipeline(steps=[('preprocessor', preprocessor),
                               ('classifier', model)])     pipeline.fit(X_train, y_train)
y_pred = pipeline.predict(X_test)     print(f"Model: {name}")
print("Confusion Matrix:\n", confusion_matrix(y_test, y_pred))
print("Classification Report:\n", classification_report(y_test, y_pred))
    print("-------------------------------------------------------")


from sklearn.tree import DecisionTreeClassifier, plot_tree

# Train a decision tree classifier tree_classifier
= DecisionTreeClassifier()
tree_classifier.fit(X_train, y_train)

# Visualize the decision tree plt.figure(figsize=(20,10))
plot_tree(tree_classifier, filled=True, feature_names=X.columns,
class_names=["Benign", "Malignant"])
plt.title("Decision Tree") plt.show()

# Dimensionality Reduction pca
= PCA(n_components=2)
X_pca = pca.fit_transform(X)

plt.figure(figsize=(8, 6)) plt.scatter(X_pca[:, 0],
X_pca[:, 1], c=y, cmap='viridis') plt.title('PCA')
```

```
plt.xlabel('Component 1') plt.ylabel('Component 2')
plt.colorbar(label='Diagnosis') plt.show()
data.to_csv("preprocessed_data.csv", index=False)
```

OUTPUT

EDA:





Dataset Overview:

id diagnosis radius_mean ... concave_points_worst symmetry_worst fractal_dimension_worst

0 842302 M 17.99 ... 0.2654 0.4601 0.11890

1 842517 M 20.57 ... 0.1860 0.2750 0.08902

2 84300903 M 19.69 ... 0.2430 0.3613 0.08758

3 84348301 M 11.42 ... 0.2575 0.6638 0.17300

4 84358402 M 20.29 ... 0.1625 0.2364 0.07678

[5 rows x 32 columns]

Dataset Info:

<class 'pandas.core.frame.DataFrame'>

RangeIndex: 569 entries, 0 to 568 Data

columns (total 32 columns):

Column Non-Null Count Dtype

--- ------ -------------- ----- 0

id 569 non-null int64

1 diagnosis 569 non-null object

2 radius_mean 569 non-null float64

3 texture_mean 569 non-null float64

4 perimeter_mean 569 non-null float64

5 area_mean 569 non-null float64

6 smoothness_mean 569 non-null float64

7 compactness_mean 569 non-null float64

8 concavity_mean 569 non-null float64

9 concave_points_mean 569 non-null float64

10 symmetry_mean 569 non-null float64

11 fractal_dimension_mean 569 non-null float64

12 radius_se 569 non-null float64

13 texture_se 569 non-null float64

14 perimeter_se 569 non-null float64

15 area_se 569 non-null float64

16 smoothness_se 569 non-null float64

17 compactness_se 569 non-null float64

18 concavity_se 569 non-null float64

19 concave_points_se 569 non-null float64

20 symmetry_se 569 non-null float64

21 fractal_dimension_se 569 non-null float64

22 radius_worst 569 non-null float64

23 texture_worst 569 non-null float64

24 perimeter_worst 569 non-null float64

25 area_worst 569 non-null float64

26 smoothness_worst 569 non-null float64

27 compactness_worst 569 non-null float64

28 concavity_worst 569 non-null float64

29 concave_points_worst 569 non-null float64

30 symmetry_worst 569 non-null float64 31 fractal_dimension_worst 569 non-null float64 dtypes: float64(30), int64(1), object(1)

memory usage: 142.4+ KB

None

Summary Statistics:

	id	radius_mean	texture_mean	...	concave_points_worst	symmetry_worst	fractal_dimension_worst
count	5.690000e+02	569.000000	569.000000	...	569.000000	569.000000	569.000000
mean	3.037183e+07	14.127292	19.289649	...	0.114606	0.290076	0.083946
std	1.250206e+08	3.524049	4.301036	...	0.065732	0.061867	0.018061
min	8.670000e+03	6.981000	9.710000	...	0.000000	0.156500	0.055040
25%	8.692180e+05	11.700000	16.170000	...	0.064930	0.250400	0.071460
50%	9.060240e+05	13.370000	18.840000	...	0.099930	0.282200	0.080040
75%	8.813129e+06	15.780000	21.800000	...	0.161400	0.317900	0.092080
max	9.113205e+08	28.110000	39.280000	...	0.291000	0.663800	0.207500

[8 rows x 31 columns]

Missing Values:

id 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

fractal_dimension_mean

0

radius_se 0

texture_se 0

perimeter_se 0 area_se

0 smoothness_se 0

compactness_se 0

concavity_se 0

concave_points_se 0

symmetry_se 0

fractal_dimension_se 0

radius_worst 0

texture_worst 0

perimeter_worst 0

area_worst 0

smoothness_worst 0

compactness_worst 0

concavity_worst 0

concave_points_worst 0

symmetry_worst 0

fractal_dimension_worst 0

dtype: int64

Classification Models:

Model: **Logistic Regression** Confusion

Matrix:

[[70 1]

[2 41]]

Classification Report: precision

recall f1-score support

0 0.97 0.99 0.98 71

1 0.98 0.95 0.96 43

accuracy 0.97 114

macro avg 0.97 0.97 0.97 114

weighted avg 0.97 0.97 0.97 114

Model: **KNN Classifier** Confusion

Matrix:

[[68 3]

[3 40]]

Classification Report: precision

recall f1-score support

0 0.96 0.96 0.96 71

1 0.93 0.93 0.93 43 accuracy 0.95 114

macro avg 0.94 0.94 0.94 114 weighted avg 0.95 0.95

0.95 114

Model: **Naive Bayes** Confusion

Matrix:

[[70 1]

[3 40]]

Classification Report: precision

recall f1-score support

0 0.96 0.99 0.97 71

1 0.98 0.93 0.95 43

accuracy 0.96 114

macro avg 0.97 0.96 0.96 114

weighted avg 0.97 0.96 0.96 114

Model: **Support Vector Machine** Confusion Matrix:

[[71 0]

[2 41]]

Classification Report:

	precision	recall	f1-score	support
0	0.97	1.00	0.99	71
1	1.00	0.95	0.98	43
accuracy			0.98	114
macro avg	0.99	0.98	0.98	114
weighted avg	0.98	0.98	0.98	114

Model: **Decision Tree Classifier** Confusion Matrix:

[[67 4]

[3 40]]

Classification Report:

	precision	recall	f1-score	support
0	0.96	0.94	0.95	71
1	0.91	0.93	0.92	43
accuracy			0.94	114
macro avg	0.93	0.94	0.93	114
weighted avg	0.94	0.94	0.94	114

Model: **Random Forest Classifier** Confusion

Matrix:

[[70 1]

[3 40]]

Classification Report: precision

recall f1-score support

0 0.96 0.99 0.97 71

1 0.98 0.93 0.95 43

accuracy 0.96 114

macro avg 0.97 0.96 0.96 114

weighted avg 0.97 0.96 0.96 114

Model**: Gradient Boosting Classifier** Confusion

Matrix:

[[69 2]

[3 40]]

Classification Report: precision

recall f1-score support

0 0.96 0.97 0.97 71

1 0.95 0.93 0.94 43

accuracy 0.96 114

macro avg 0.96 0.95 0.95 114

weighted avg 0.96 0.96 0.96 114

Tree-based Models: Model: **Decision Tree Classifier** Confusion Matrix:

[[67 4]

[3 40]]

Classification Report: precision

recall f1-score support

0 0.96 0.94 0.95 71

1 0.91 0.93 0.92 43

accuracy 0.94 114

macro avg 0.93 0.94 0.93 114

weighted avg 0.94 0.94 0.94 114

Model: **Random Forest Classifier** Confusion Matrix:

[[70 1]

[3 40]]

Classification Report: precision

recall f1-score support

0 0.96 0.99 0.97 71

1 0.98 0.93 0.95 43 accuracy 0.96 114

macro avg 0.97 0.96 0.96 114 weighted avg 0.97 0.96 0.96 114

Model: **Gradient Boosting Classifier** Confusion Matrix:

[[69 2]

[3 40]]

Classification Report: precision recall f1-score support

0 0.96 0.97 0.97 71

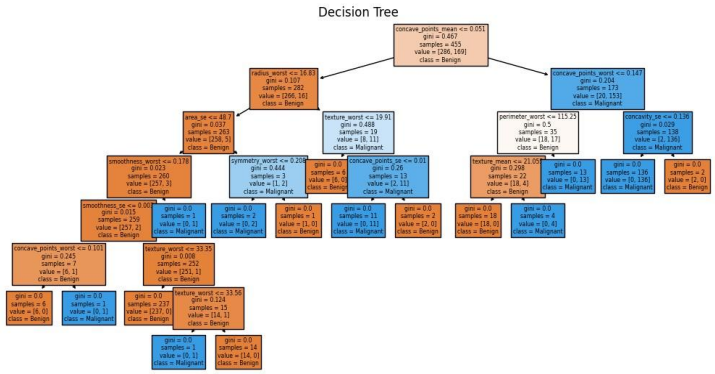
1 0.95 0.93 0.94 43

accuracy 0.96 114

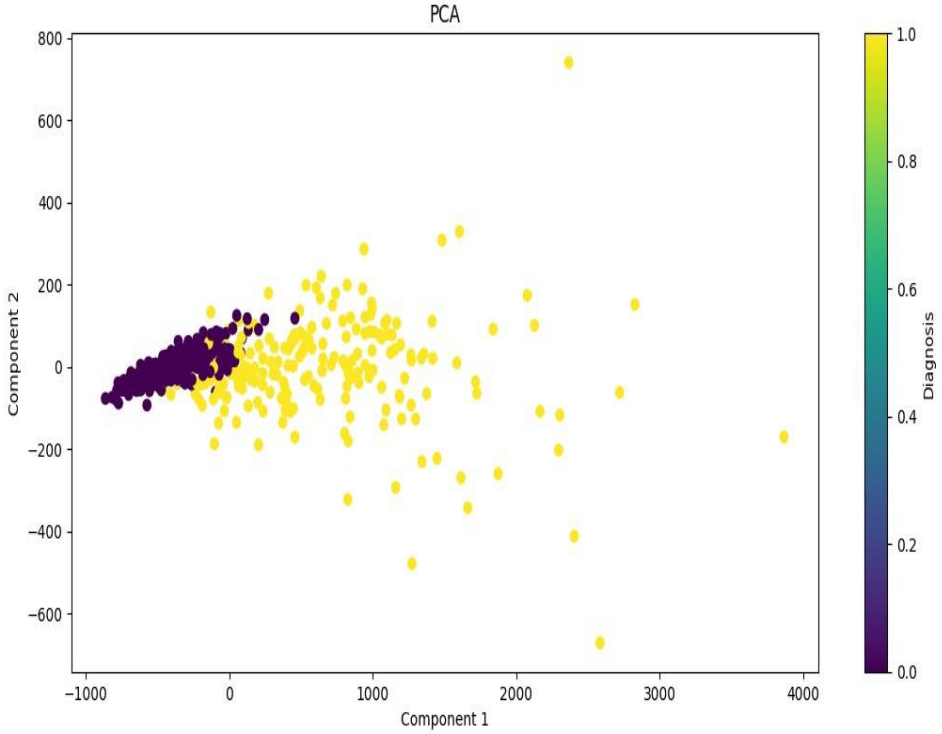
macro avg 0.96 0.95 0.95 114

weighted avg 0.96 0.96 0.96 114

DECISION TREE



PCA:



OBSERVATIONS

- Logistic Regression, SVM, and Naive Bayes demonstrate the highest accuracy rates among the evaluated models, achieving 97% and 98% accuracy, respectively.

- Precision and recall metrics highlight the effectiveness of these topperforming models in striking a balance between correctly identifying malignant tumors (true positives) while minimizing false positives and false negatives.

- Decision Tree Classifier, KNN Classifier, Random Forest Classifier, and Gradient Boosting Classifier also exhibit competitive performance, with accuracy rates ranging from 94% to 96%.

- While SVM emerges as the top-performing model, it's essential to consider various factors such as computational efficiency, interpretability, and specific application requirements when selecting the most appropriate model.

- The findings underscore the potential of machine learning models to complement existing diagnostic approaches and assist healthcare professionals in early breast cancer detection, ultimately contributing to enhanced patient care and outcomes.

CONCLUSION

In the evaluation of various classification models using the provided breast cancer dataset, it becomes apparent that several models perform admirably in accurately predicting tumor malignancy. Notably, Support Vector Machine (SVM), Logistic Regression, and Naive Bayes emerge as the top-performing models, achieving remarkable accuracy rates of 98% and 97%, respectively. These models also exhibit impressive precision and recall metrics, indicating their capacity to correctly classify malignant and benign tumors while minimizing false positives and false negatives. Despite their high performance, it's essential to consider factors such as computational efficiency, interpretability, and specific clinical requirements when selecting the most suitable model for deployment in practice.

On the other hand, Decision Tree Classifier, KNN Classifier, Random Forest Classifier, and Gradient Boosting Classifier also showcase competitive performance, albeit with slight variations in accuracy and other evaluation metrics. While SVM stands out as the top-performing model in this analysis, each model has its strengths and weaknesses, making it crucial to weigh these factors carefully in real-world applications.

The results of this study suggest that machine learning models hold significant promise in aiding the early detection and diagnosis of breast cancer. By leveraging the power of predictive analytics, healthcare professionals can make more informed decisions, leading to improved patient outcomes and potentially saving lives. However, further research is warranted to explore avenues for finetuning model parameters, investigating ensemble methods, or incorporating additional features to enhance prediction accuracy and robustness.