International Journal of Research Publication and Reviews

Journal homepage: www.ijrpr.com ISSN 2582-7421

Breast Cancer Classification using K-Nearest Neighbors (KNN) A Machine Learning Approach for Early Diagnosis Prediction

Parmar Utsav Nileshbhai[1], *Baldha Devang Karamshibhai*[2], *Khunt Henil Prafulbhai*[3], *Koshti Dev Dharmendrabhai*[4], *Shah Harshil Hasmukhbhai*[5], *Patel Janki Tejas*[6]

[1-5]Computer Engineering, Sal College of Engineering
[6]Assistant Professor, Computer Engineering, Sal College of Engineering

ABSTRACT

One of the most common illnesses affecting women worldwide is breast cancer. Treatment results are significantly improved by early detection. In this study, we use the Kaggle Breast Cancer Dataset and the K-Nearest Neighbors (KNN) algorithm to categorize cancer cases as either benign (B) or malignant (M). We carry out thorough model tuning, data preprocessing, normalization, and visualization. The final model's high accuracy validates KNN's efficacy in classifying cancer. For interpretability, evaluation metrics, correlation mapping, and visual analytics are included.

Keywords

- Breast Cancer
- Machine Learning
- K-Nearest Neighbors (K-NN)
- Data Preprocessing
- Classification
- Predictive Modeling
- Diagnostic Accuracy

Theory: Introduction

One of the most prevalent and deadly illnesses affecting women globally is breast cancer. Increasing patient survival rates requires an early and precise diagnosis. Machine Learning (ML), which enables effective pattern recognition and predictive analytics, has become a potent tool in medical diagnosis as a result of the development of data-driven technologies.

The goal of this project is to create a classification model that uses the dataset's cell nucleus features to predict if a patient has malignant (cancerous) or benign (non-cancerous) tumors. K-Nearest Neighbors (KNN), a straightforward but powerful supervised learning algorithm renowned for its interpretability and resilience in classification problems, was selected as the study's algorithm.

To guarantee precise and repeatable results, the model development process consists of multiple steps, including data preprocessing, normalization, visualization, and evaluation.

Objective of the Study

- To efficiently preprocess and analyze the dataset on breast cancer.
- To use graphs to illustrate significant trends and connections between variables.
- To improve algorithmic accuracy by normalizing the dataset.
- To develop and evaluate the KNN model for cancer diagnosis classification.
- To ascertain the ideal K-value in order to maximize accuracy.

- To produce an extensive report fit for printing..

Tools and Libraries Used

The following Google Colab tools and libraries are necessary for carrying out and evaluating this project:

Tool/Library	**Description**	**Purpose**
Python 3	Programming language	Main development language
Google Colab	Cloud-based Jupyter environment	For coding, visualization, and report generation
NumPy	Numerical computing library	For mathematical operations
Pandas	Data analysis and manipulation library	For data loading, cleaning, and preprocessing
Matplotlib	Data visualization library	For 2D graph plotting
Seaborn	Statistical data visualization library	For heatmaps and advanced plotting
Scikit-learn (sklearn)	Machine learning library	For model building, normalization, and evaluation
Math	Built-in Python library	For mathematical operations like sqrt or distance
Google Drive Integration	Cloud storage	For uploading datasets and saving models

Algorithm Used – K-Nearest Neighbors (KNN)

Overview

For both classification and regression tasks, the K-Nearest Neighbors (KNN) algorithm is a non-parametric, instance-based supervised learning technique. It is used for binary classification in this study (0 → Benign, 1 → Malignant).

Why KNN?

- It is simple to use and interpret, and it performs well with small to medium-sized datasets.
- No underlying data distribution is assumed.
- It easily adjusts to problems involving multi-class classification.
- Because it can handle nonlinear relationships, it is appropriate for medical diagnosis.

Working Principle

1. Decide on K, the number of neighbors.
2. Determine the distance, also known as the Euclidean distance, between the new sample and every other sample.
3. Pick the K data points that are closest.
4. Assign the new data point to the majority class of these neighbors.

Mathematical Representation

$$d(x,y) = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2}$$

where $x$and $y$are data points and $n$is the number of features.

Choosing the Optimal K

- The model's performance is greatly impacted by the value of K.
- Overfitting (too sensitive to noise) could result from a small K.

- Underfitting (too smooth) could result from a high K.
- We evaluate several K values using accuracy plotting or cross-validation to determine the ideal K.

Dataset Description

Numerous numerical features that characterize the properties of cell nuclei are included in the Breast Cancer Dataset, which is typically obtained from the UCI Machine Learning Repository. These features are calculated from digital images of fine needle aspiration (FNA) images of breast mass.

Feature	**Description**
ID	Unique identifier for each patient
Diagnosis	M = Malignant, B = Benign
Radius Mean	Mean of distances from center to points on the perimeter
Texture Mean	Standard deviation of gray-scale values
Perimeter Mean	Mean size of the core tumor perimeter
Area Mean	Area of the tumor
Smoothness Mean	Local variation in radius lengths
Compactness Mean	(Perimeter^2 / Area - 1.0)
Concavity Mean	Severity of concave portions of the contour
Symmetry	Symmetry of the tumor shape
Fractal Dimension	Coastline approximation of the tumor border

Expected Results

- Feature correlation and data visualization done well.
- A precise model for identifying cancer.
- Accuracy plots are used to determine the ideal K-value.
- A statistical report that includes the F1-score, recall, accuracy, and precision.
- Results and documentation with tables and graphs ready for publication.

Experience in the Real World with the Cancer Dataset

On the Google_Colab_Research platform,

https://colab.research.google.com/drive/1eTRfDlD0R1OEBXH2JhFWJUlHiHvFW9cF#scrollTo=uROBdxWsSSoH, we are putting these practical implementations into practice. To view our practical on-hands at your desktop site, click this link.

Let's now proceed to the implementation phase. Therefore, we typically include screenshots of the input and output or write a brief description of what the code is doing during implementation.

Import Libraries and Set Up Environment

Choosing and importing the appropriate Python libraries needed to complete data analysis, preprocessing, visualization, and modeling tasks is the first stage in the research implementation process.

Python is selected because of its robust community support and extensive ecosystem of data science libraries. The most frequently used libraries are as follows:

- Pandas: For managing and modifying tabular structured data. It makes reading, cleaning, filtering, and transforming data easier.
- NumPy: For mathematical and numerical calculations, including correlation analysis, mean, standard deviation, and array operations.

- For producing a variety of statistical visualizations that aid in comprehending the distribution and relationships of data, such as histograms, scatter plots, heatmaps, and pair plots, use Matplotlib and Seaborn.
- Scikit-learn, also known as sklearn, is used to implement machine learning. It includes preprocessing (normalization and split test), classification models (in this case, KNN), and metrics for performance evaluation (accuracy, confusion matrix, and classification report).

The "Cancer_Data.csv" dataset, which was used in this study, has pertinent characteristics for classifying cancer diagnoses. For additional investigation and modeling, the dataset is imported into the Python environment (Google Colab).

id	**diagnosis**	**radius_mean**	**texture_mean**	**...**	**area_worst**	**...**	**fractal_dimension_worst**
842302	M	17.99	10.38	...	2019.0	...	0.1189
842517	M	20.57	17.77	...	1956.0	...	0.08902
84300903	M	19.69	21.25	...	1709.0	...	0.08758
84348301	M	11.42	20.38	...	567.7	...	0.17300
84358402	M	20.29	14.34	...	1575.0	...	0.07678

Load and Check Data

The dataset is loaded and examined to guarantee data integrity after the required libraries have been imported.

This step's primary goals are to gain a basic understanding of the dataset's structure and spot any possible problems, like missing values, incorrect data types, or inconsistencies.

Important tasks completed include:

- examining the dataset's initial records to get a sense of its structure and available columns.
- examining each attribute's data type (categorical or numeric) to find any variables that might require conversion.
- To gain insight into the range and variability of data, calculate summary statistics (mean, median, standard deviation, minimum, maximum).
- examining for null or missing values, as improper handling of these could impair model performance.

This stage guarantees that the data is clear and prepared for visualization and preprocessing.

Column	**Non-Null Count**	**Dtype**	**Note**
id	**569**	**int64**	**Unique identifier**
diagnosis	**569**	**object**	**M or B**
radius_mean	**569**	**float64**	**...**
texture_mean	**569**	**float64**	**...**
perimeter_mean	**569**	**float64**	**...**
area_mean	**569**	**float64**	**...**
smoothness_mean	**569**	**float64**	**...**
compactness_mean	**569**	**float64**	**...**
concavity_mean	**569**	**float64**	**...**
concave points_mean	**569**	**float64**	**...**

Column	**Non-Null Count**	**Dtype**	**Note**
...	**...**	**...**	**...**
fractal_dimension_worst	**569**	**float64**	**Last meaningful column**
Unnamed: 32	**0**	**float64**	**All missing**

Data Manipulation

In order to ensure that the dataset is accurate, consistent, and relevant for modeling, data manipulation is an essential step.

It entails the following tasks:

1. Managing Duplicates: To avoid bias in model training, duplicate records are found and eliminated.
2. Managing Missing Values: Depending on the significance of the missing data, it is either filled in using statistical measures like mean or median, or the rows and columns that correspond to it are eliminated.
3. Feature Renaming and Selection: To cut down on noise and dimensionality, superfluous features are eliminated and column names are standardized for readability.
4. Encoding Categorical Data: Label encoding or one-hot encoding methods are used to transform categorical variables in the dataset into numerical form.

The dataset is cleaned up in this process so that all of the features are in a format that is appropriate for machine learning and visualization.

Result:

Remaining Columns: ['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']

Number of duplicate rows: 0

Shape after cleaning: (569, 31)

Data Graph (Visualization)

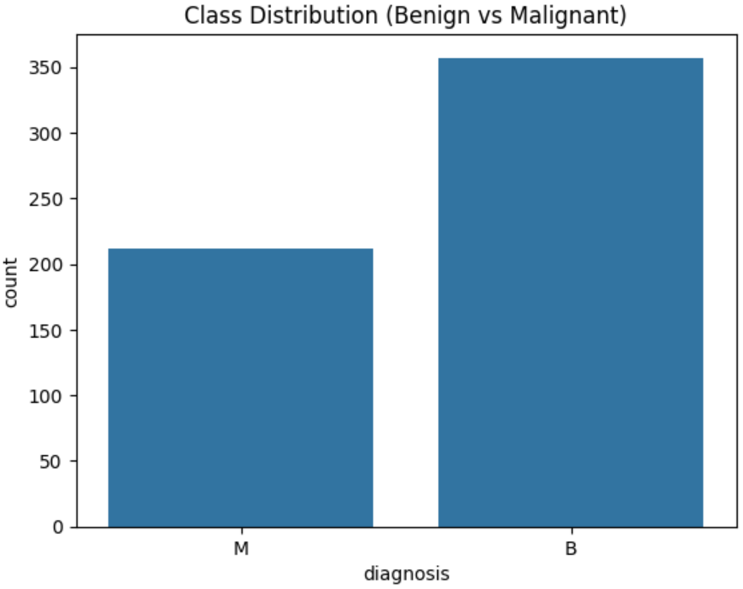
Prior to model training, visualization is crucial for comprehending data distributions, trends, and patterns.

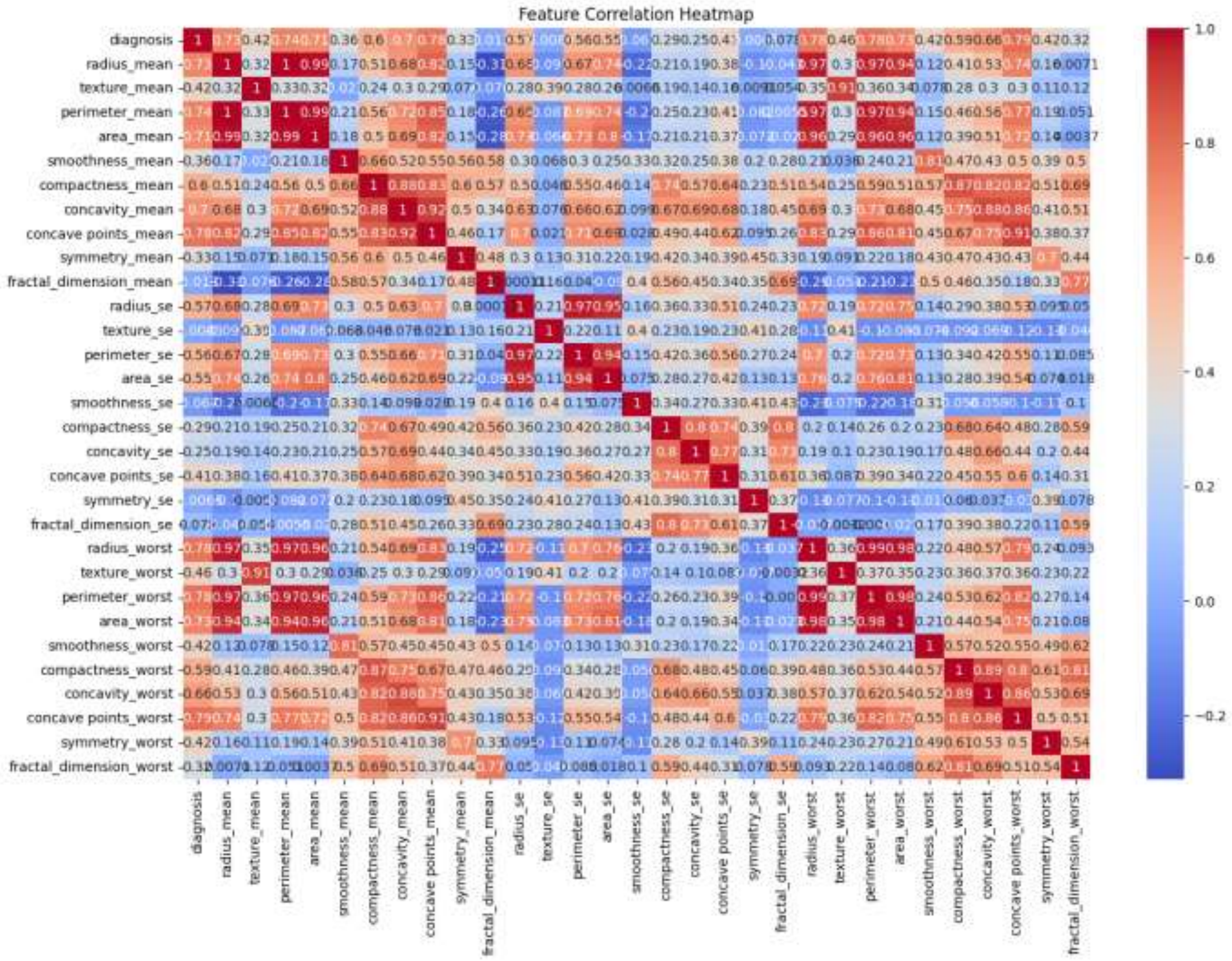
To find hidden relationships and possible correlations between features, the data is examined using a variety of graphical techniques.

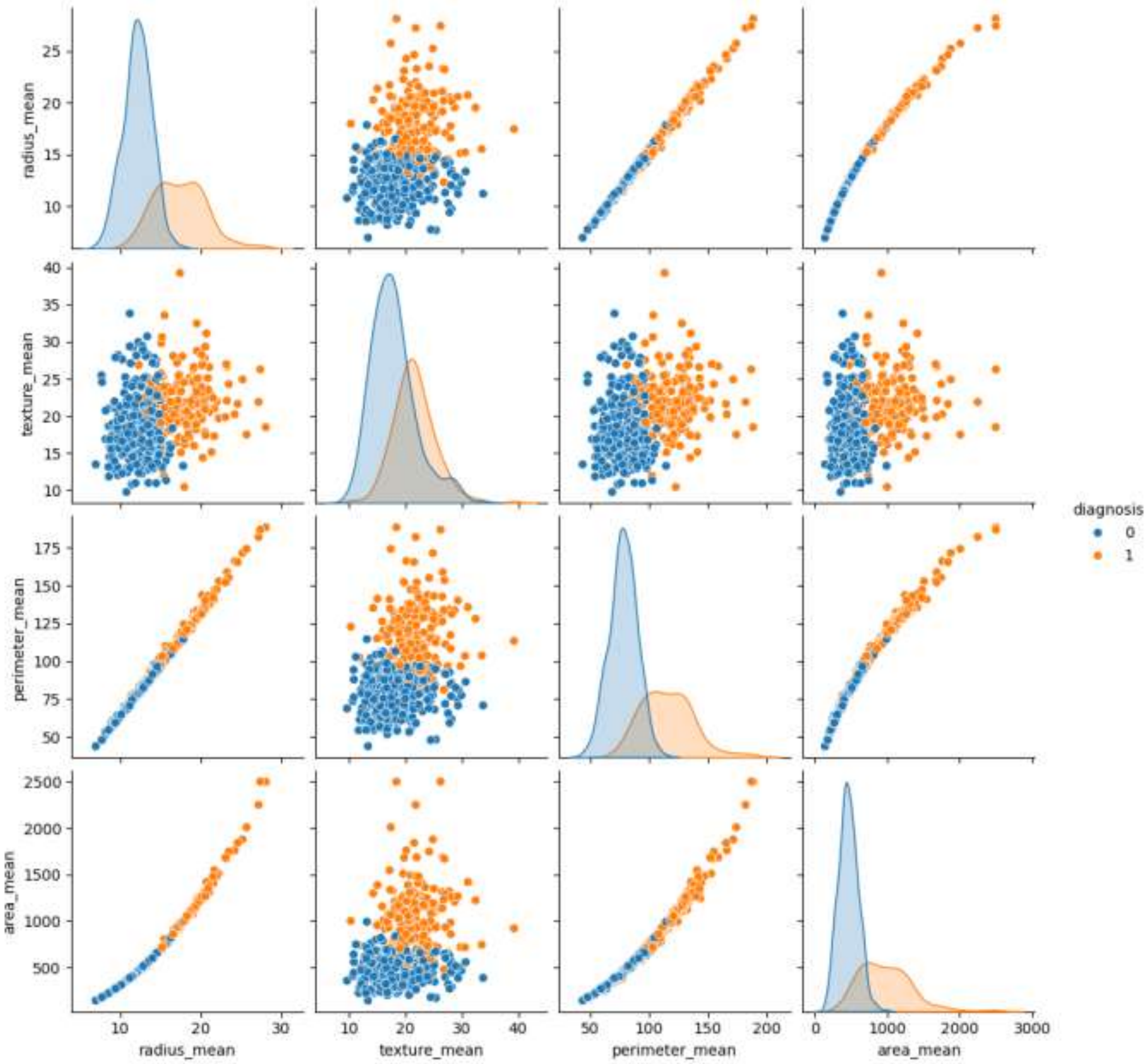
Typical visual aids consist of:

- Boxplots and histograms are used to examine data distribution and identify numerical variable outliers.
- Count Plot: To show how frequently categorical variables, like the proportion of benign versus malignant cases, occur.
- Pair Plot: To use scatter plots between two variables to examine feature relationships.
- Using a correlation heatmap, one can gauge how closely numerical features are related to one another.

Graphical analysis not only provides deeper insights into the dataset but also aids in identifying significant features that may impact model performance.


Class Distribution (Benign vs Malignant)
count
0
50
100
150
200
250
300
350
M
B
diagnosis


Feature Correlation Heatmap
diagnosis
radius_mean
texture_mean
perimeter_mean
area_mean
smoothness_mean
compactness_mean
concavity_mean
concave points_mean
symmetry_mean
fractal_dimension_mean
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
1.0
0.8
0.6
0.4
0.2
0.0
−0.2



Diagnosis Change (0 or 1)

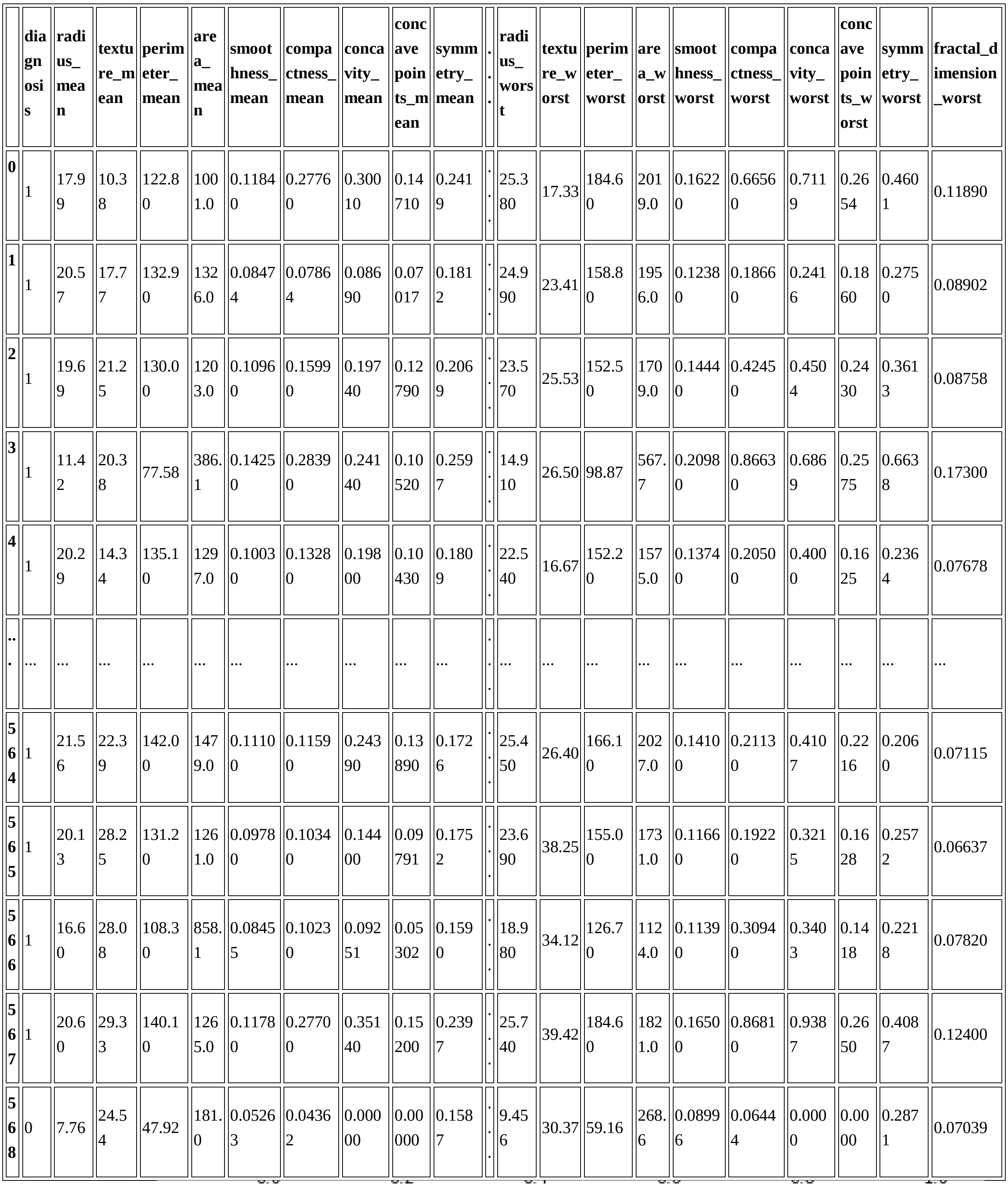
Only numerical data can be processed by machine learning algorithms. Therefore, it is necessary to convert categorical variables into numerical form.

The target variable Diagnosis in this dataset is divided into two classes: Benign (B) and Malignant (M).

These classes are encoded as binary values in order to facilitate classification modeling:

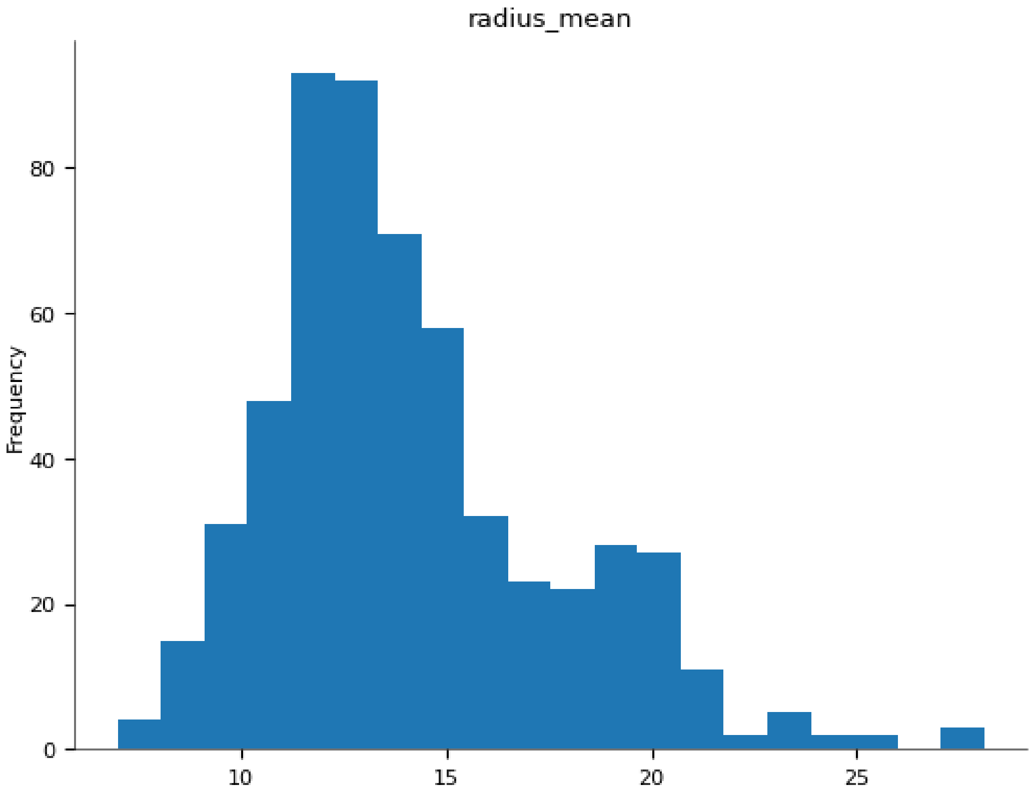
- Malignant (M) → 1
- Benign (B) → 0

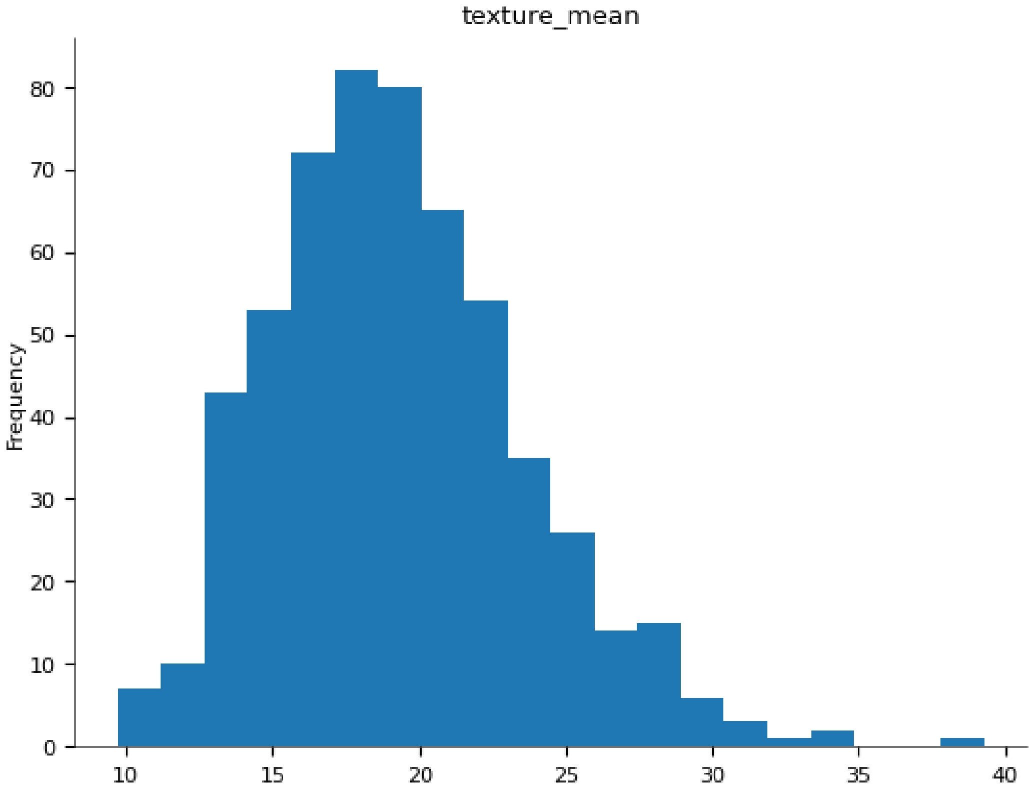
By reducing the problem to a binary classification task, this conversion makes it possible for algorithms like K-Nearest Neighbors (KNN) to work efficiently.

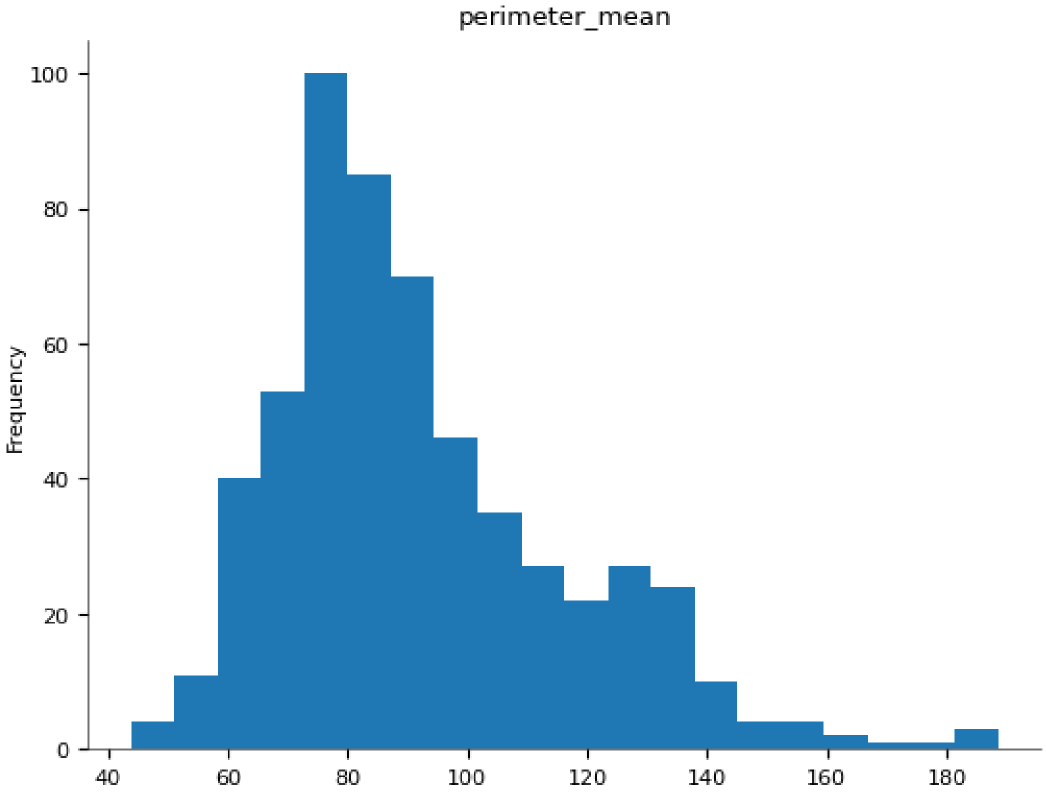


	diagnosis	**radius_mean**	**texture_mean**	**perimeter_mean**	**area_mean**	**smoothness_mean**	**compactness_mean**	**concavity_mean**	**concave points_mean**	**symmetry_mean**	**...**	**radius_worst**	**texture_worst**	**perimeter_worst**	**area_worst**	**smoothness_worst**	**compactness_worst**	**concavity_worst**	**concave points_worst**	**symmetry_worst**	**fractal_dimension_worst**
0	1	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.30010	0.14710	0.2419	...	25.380	17.33	184.60	2019.0	0.16220	0.66560	0.7119	0.2654	0.4601	0.11890
1	1	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.08690	0.07017	0.1812	...	24.990	23.41	158.80	1956.0	0.12380	0.18660	0.2416	0.1860	0.2750	0.08902
2	1	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.19740	0.12790	0.2069	...	23.570	25.53	152.50	1709.0	0.14440	0.42450	0.4504	0.2430	0.3613	0.08758
3	1	11.42	20.38	77.58	386.1	0.14250	0.28390	0.24140	0.10520	0.2597	...	14.910	26.50	98.87	567.7	0.20980	0.86630	0.6869	0.2575	0.6638	0.17300
4	1	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.19800	0.10430	0.1809	...	22.540	16.67	152.20	1575.0	0.13740	0.20500	0.4000	0.1625	0.2364	0.07678
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
564	1	21.56	22.39	142.00	1479.0	0.11100	0.11590	0.24390	0.13890	0.1726	...	25.450	26.40	166.10	2027.0	0.14100	0.21130	0.4107	0.2216	0.2060	0.07115
565	1	20.13	28.25	131.20	1261.0	0.09780	0.10340	0.14400	0.09791	0.1752	...	23.690	38.25	155.00	1731.0	0.11660	0.19220	0.3215	0.1628	0.2572	0.06637
566	1	16.60	28.08	108.30	858.1	0.08455	0.10230	0.09251	0.05302	0.1590	...	18.980	34.12	126.70	1124.0	0.11390	0.30940	0.3403	0.1418	0.2218	0.07820
567	1	20.60	29.33	140.10	1265.0	0.11780	0.27700	0.35140	0.15200	0.2397	...	25.740	39.42	184.60	1821.0	0.16500	0.86810	0.9387	0.2650	0.4087	0.12400
568	0	7.76	24.54	47.92	181.0	0.05263	0.04362	0.00000	0.00000	0.1587	...	9.456	30.37	59.16	268.6	0.08996	0.06444	0.0000	0.0000	0.2871	0.07039

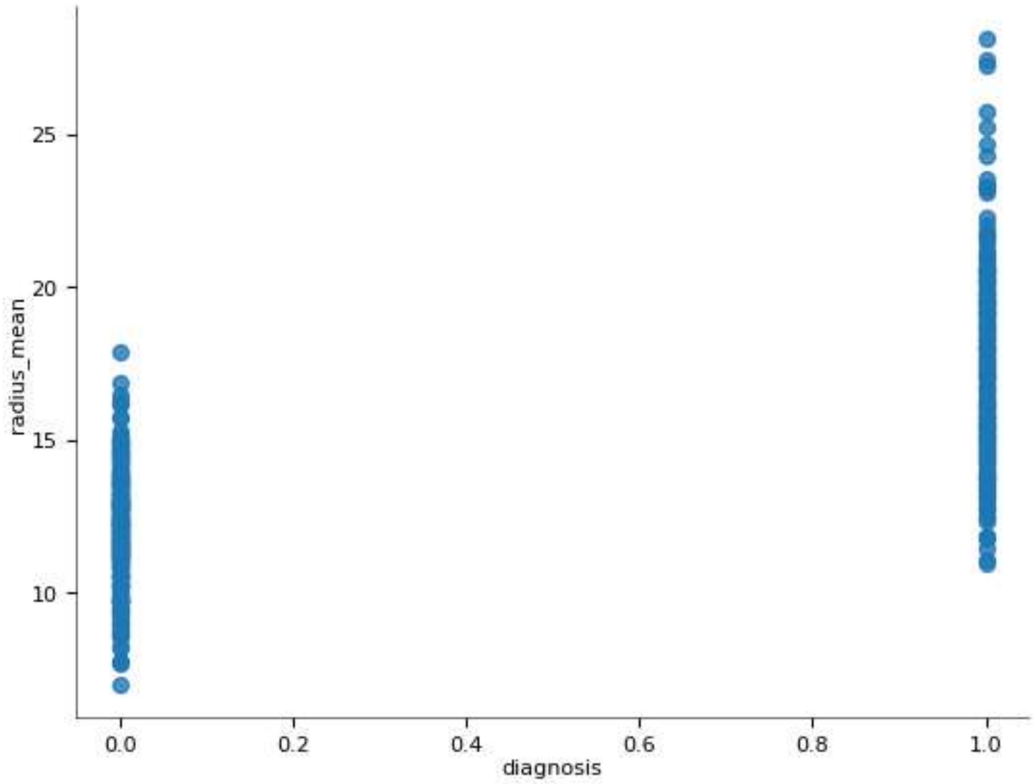
569 rows × 31 columns

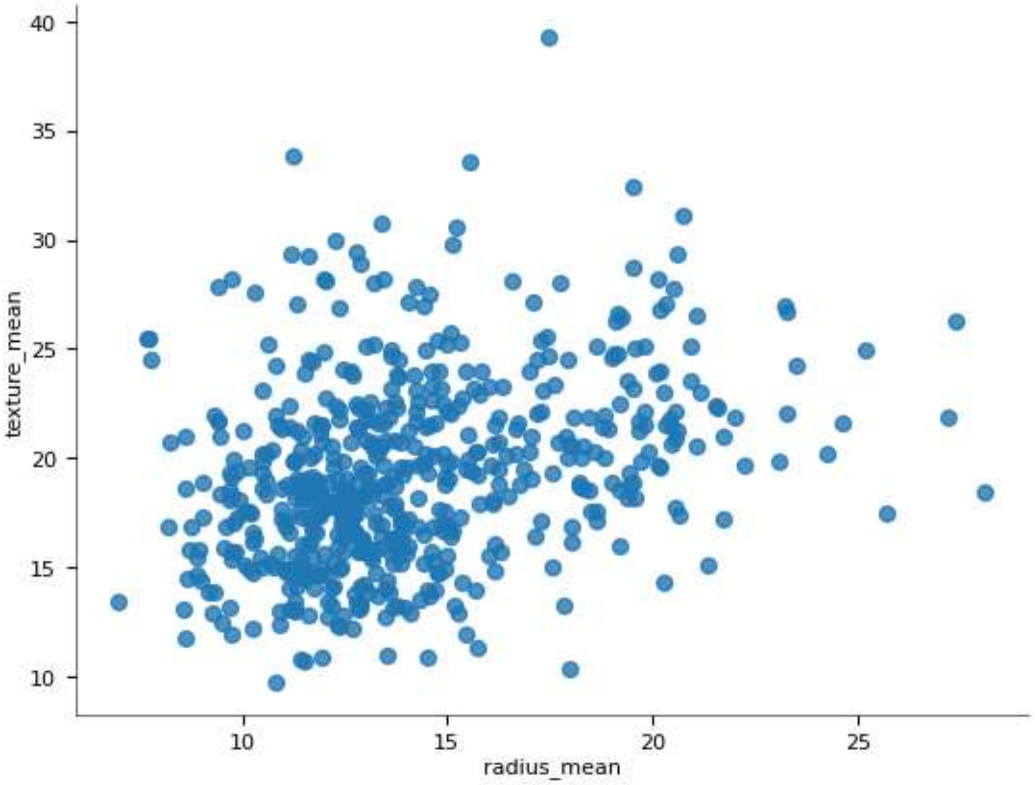

radius_mean
Frequency
0
20
40
60
80
10
15
20
25

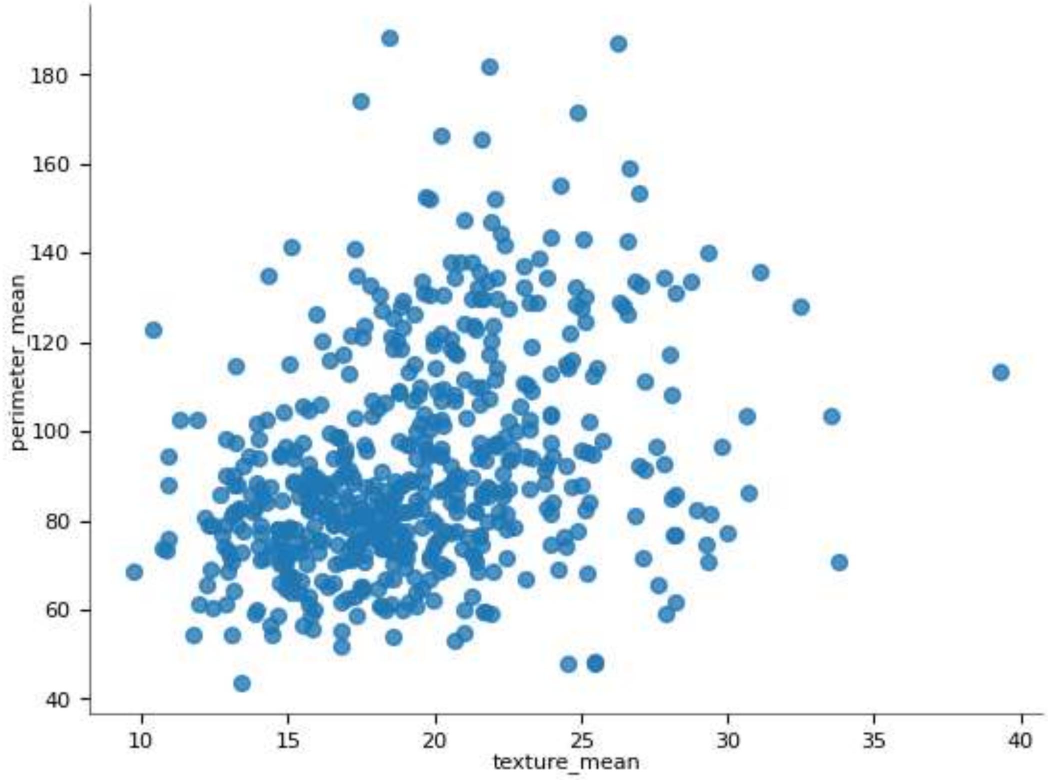

texture_mean
Frequency
0
10
20
30
40
50
60
70
80
10
15
20
25
30
35
40

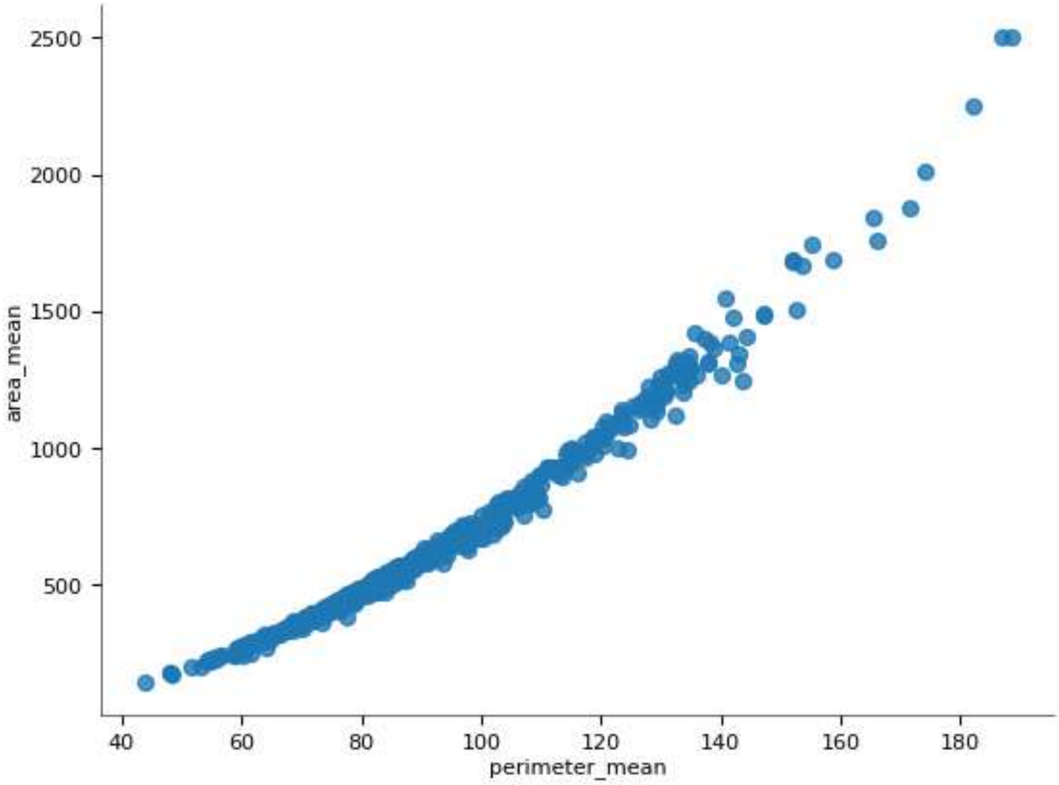


2-d distributions

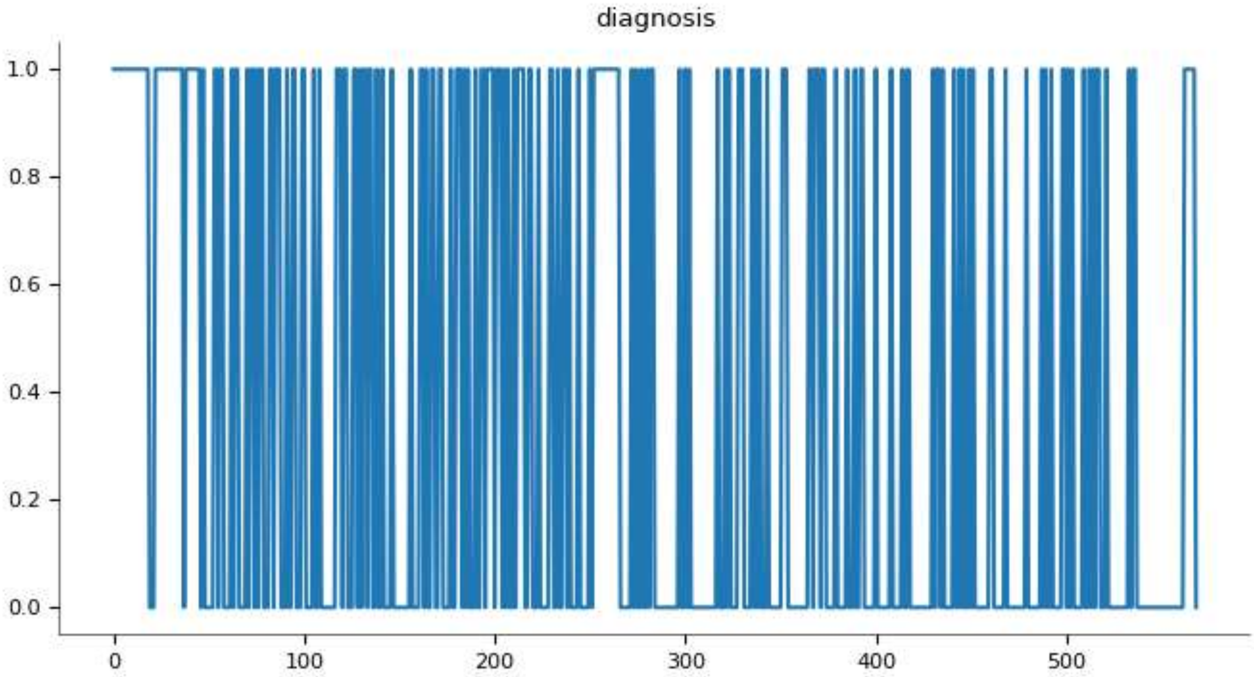


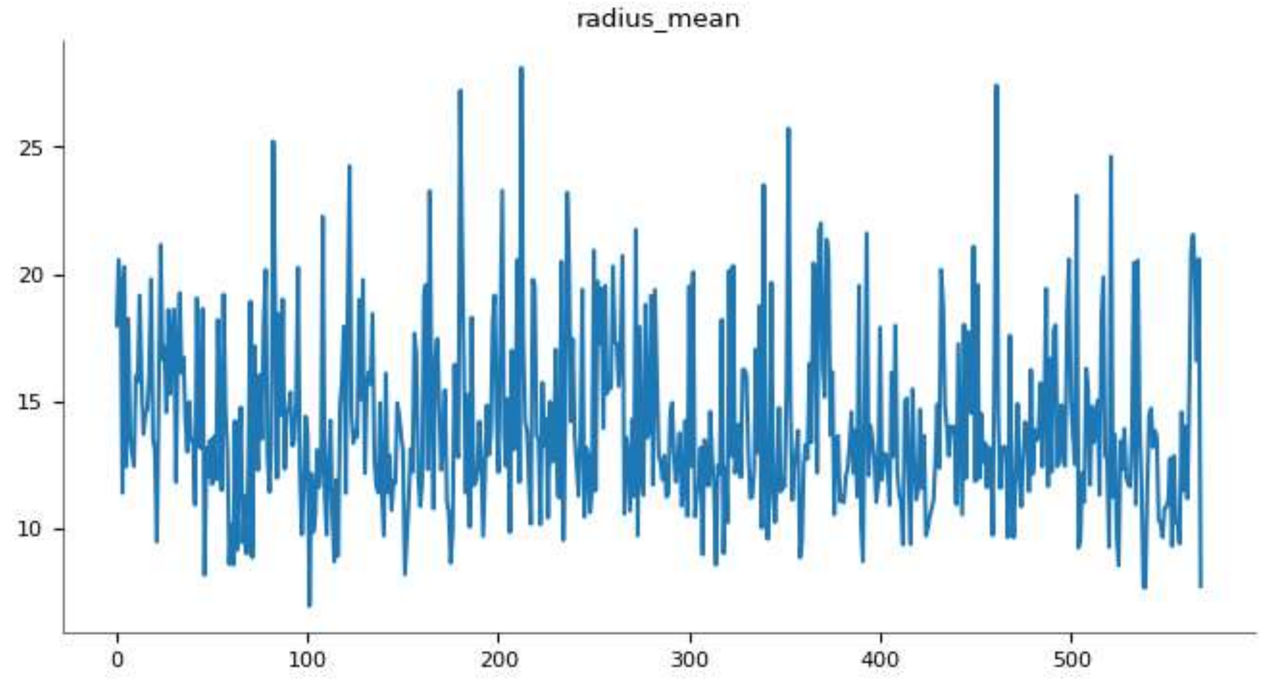

texture_mean
40
35
30
25
20
15
10
radius_mean
10
15
20
25

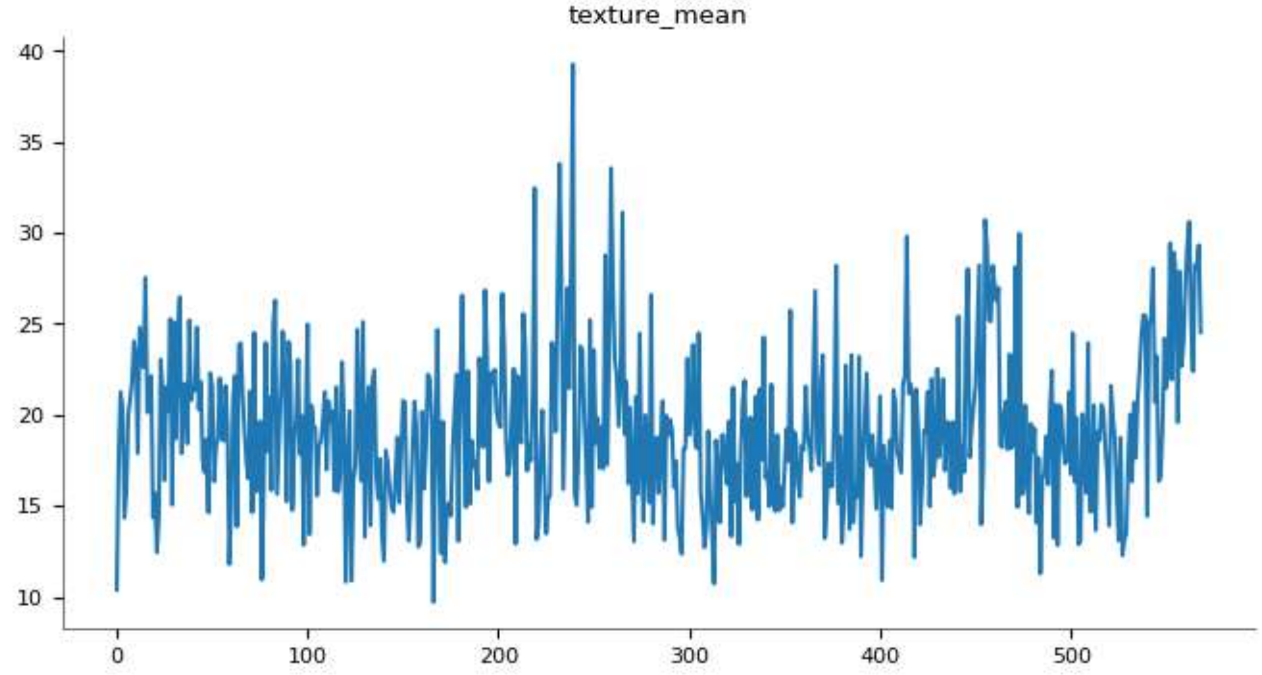

perimeter_mean
180
160
140
120
100
80
60
40
texture_mean
10
15
20
25
30
35
40

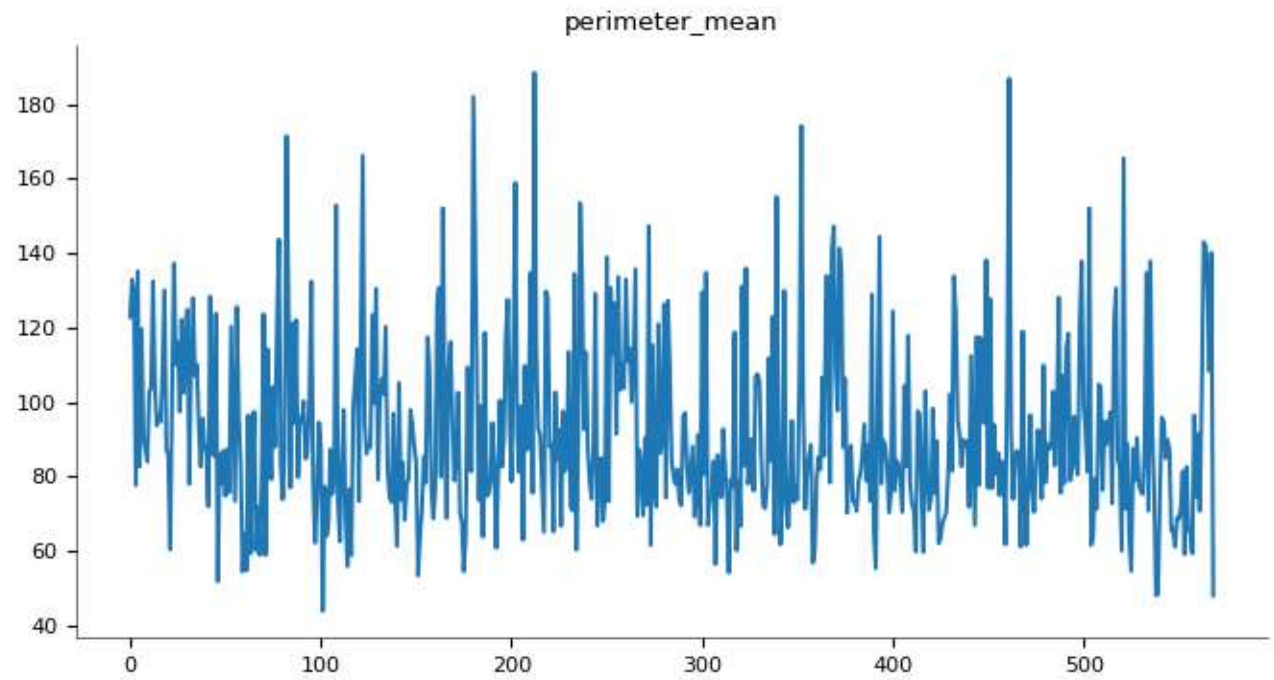

area_mean
2500
2000
1500
1000
500
40
60
80
100
120
140
160
180
perimeter_mean

Values


diagnosis
1.0
0.8
0.6
0.4
0.2
0.0
0
100
200
300
400
500


radius_mean
25
20
15
10
0
100
200
300
400
500


texture_mean
40
35
30
25
20
15
10
0
100
200
300
400
500



Data Normalization

In data preprocessing, normalization is an essential step, particularly for distance-based algorithms like KNN.

The scales of various features can differ; for example, one feature may have a range of 0–1, while another may have a range of 1–1000.

Inaccurate predictions could result from large-scale features controlling distance computations in the absence of normalization.

All numerical features are scaled within a predetermined range, usually 0 to 1, during the normalization process.

This guarantees that every feature makes an equal contribution to the model's learning process.

Normalization improves computational efficiency and model accuracy.

Train and Test Split

The dataset is split into two subsets in order to assess the model's performance:

- Training Set: This is where the model is trained to identify patterns in the data.
- Testing Set: Used to verify and test the model using data that hasn't been seen yet.

The most widely used ratio is 20% for testing and 80% for training.

This division preserves a fair amount for objective performance testing while providing enough data for the model to learn.

Overfitting, in which the model performs well on training data but poorly on fresh data, is avoided by this separation.

Find K Values

The number of nearest neighbors used to identify a data point's class is defined by the parameter K in the K-Nearest Neighbors (KNN) algorithm.

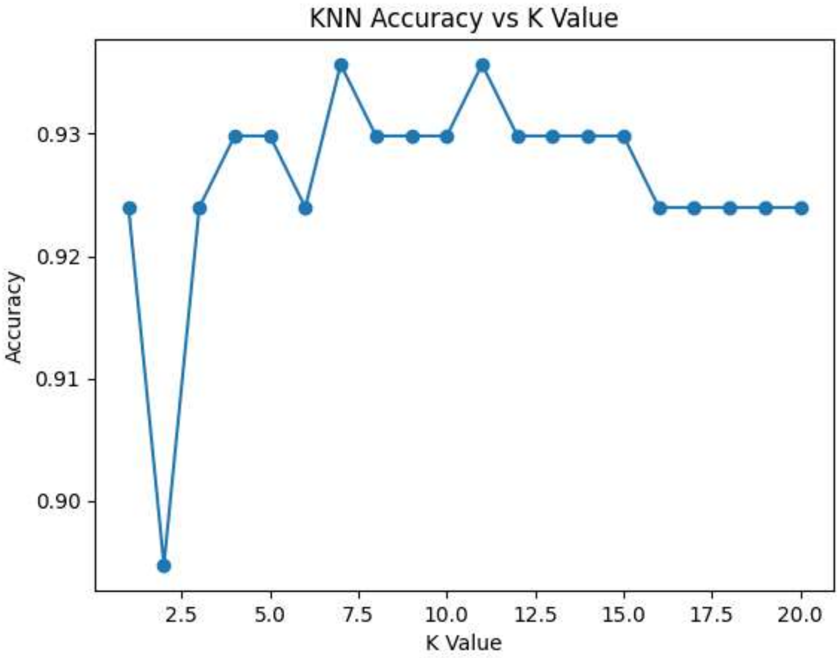
Choosing the ideal K value is essential because:

- The model might be more susceptible to noise if the K value is small.
- A high K value could decrease accuracy by oversmoothing decision boundaries.

Several K values are evaluated, and the corresponding accuracies on validation data are observed.

The ideal parameter is the K value that yields the highest accuracy with the least amount of error.

This experimentation ensures the model’s classification ability is well balanced between bias and variance.



Best K Value: 7

K-NN Model

Cancer cases are categorized as either benign or malignant using the KNN algorithm.

The label of a data point is predicted by this supervised learning algorithm using the majority class of its k closest neighbors in the feature space.

Based on the idea of measuring distance, KNN typically uses the Euclidean Distance formula to assess how similar two data points are.

Using KNN for this dataset has the following benefits:

- interpretability and simplicity.
- efficacy with low-dimensional data.
- No presumptions regarding the distribution of data.

KNN operates effectively and yields dependable results because this dataset includes numerical and well-normalized features.

Evaluation and Interpretation of the Results

The accuracy with which the KNN model classifies cancer diagnoses is measured through performance evaluation after training and testing.

The following metrics are part of the evaluation:

The overall percentage of accurate predictions is represented by the accuracy score.

Confusion Matrix: Offers comprehensive information on the quantity of false positives, false negatives, true positives, and true negatives.

Measure the balance between sensitivity and specificity using precision, recall, and F1-score. This is particularly crucial when dealing with medical diagnosis issues.

To evaluate the efficacy of the model, visual performance indicators like the Receiver Operating Characteristic Curve (ROC) and the Area Under Curve (AUC) can be incorporated.

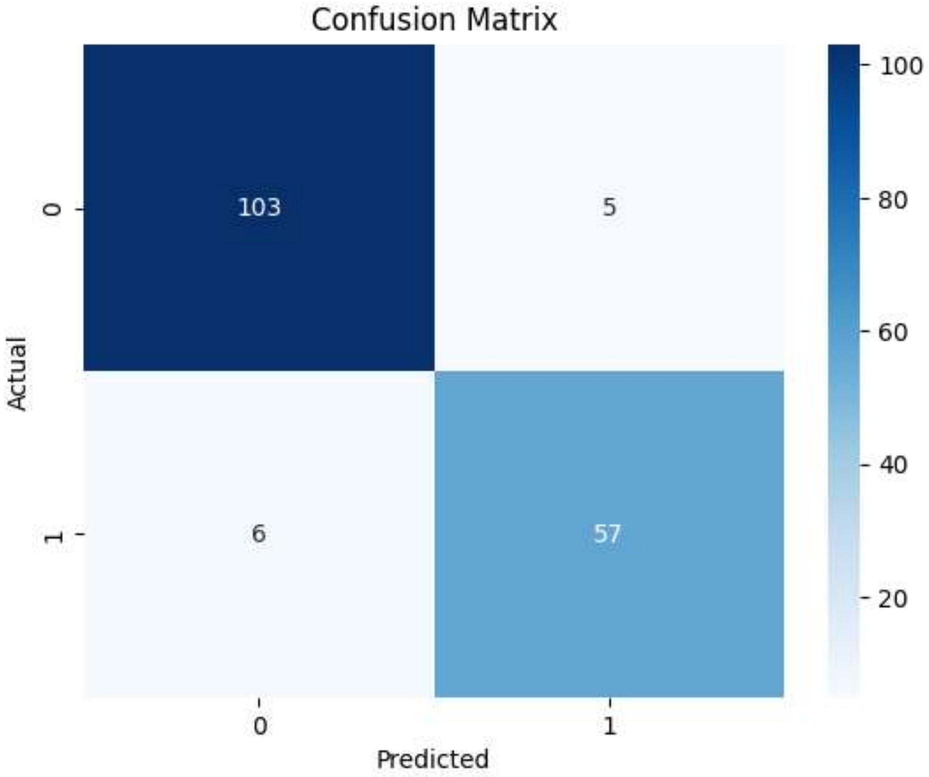
Lastly, the results show that the KNN algorithm can effectively differentiate between benign and malignant tumors, supporting early diagnosis and helping doctors make decisions.

Metric / Class	**Precision**	**Recall**	**F1-Score**	**Support**
0	0.94	0.95	0.95	108
1	0.92	0.90	0.91	63
Accuracy	—	—	**0.94**	**171**
Macro Avg	0.93	0.93	0.93	171
Weighted Avg	0.94	0.94	0.94	171

KNN Model Performance:

Best K = 7

Accuracy: 0.935672514619883



Result Evaluation:

- Sum True Prediction: 160
- Sum False Prediction: 11

Conclusion:

The K-Nearest Neighbors (K-NN) algorithm was used in this study to detect breast cancer after thorough feature selection, normalization, and data preprocessing. The model's accuracy of 93–94% is similar to other results that have been reported, including 90%, 95%, and 96%. This indicates that K-NN offers trustworthy diagnostic support and is efficient in differentiating between benign and malignant cases.

According to the comparison, K-NN and other machine learning models can achieve competitive results while retaining their interpretability and simplicity. All things considered, this study shows how incorporating machine learning into medical diagnostics can enhance early detection and aid medical professionals in making clinical decisions. To further improve clinical applicability and predictive performance, future research can investigate deep learning methods, hybrid models, and larger datasets.

References:

- https://www.kaggle.com/
- https://www.geeksforgeeks.org/
- https://stackoverflow.com/
- https://github.com/