Department of Computer Engineering
Academic Year 2022-2023

Breast Cancer Prediction

Machine Learning Laboratory

By

Preet Gada	**60004200101**
Jiya Patel	**60004200105**
Ayush Jain	**60004200132**

Guide:

Prof. (Dr.) Ruhina Karani

Assistant Professor

Breast Cancer Prediction

1. PROBLEM STATEMENT

The problem statement of this report is to develop and compare the performance of three machine learning algorithms, namely SVM, KNN, and Naive Bayes, for the accurate prediction of breast cancer. The study aims to train the models on a dataset of patient information, including clinical and imaging data, and evaluate their ability to classify breast tumors as either benign or malignant. The key objective is to identify the most effective algorithm for breast cancer prediction, which can assist healthcare professionals in the early detection and diagnosis of breast cancer, leading to improved patient outcomes and survival rates.

2. INTRODUCTION

a. Need

The need for breast cancer prediction using machine learning is driven by the desire to improve early detection and diagnosis of breast cancer, which is one of the most common cancers affecting women worldwide. Machine learning algorithms offer the potential to accurately classify breast tumors as either benign or malignant based on their characteristics, such as size, shape, and texture. This can help healthcare professionals to identify cases of breast cancer at an early stage, when treatment is more effective and outcomes are better.

Moreover, the existing methods of breast cancer prediction rely on the subjective interpretation of mammograms by radiologists, which can result in a high rate of false positives or false negatives. Machine learning algorithms, on the other hand, can provide a more objective and consistent approach to breast cancer prediction by analyzing large amounts of data and identifying patterns and relationships that are not easily discernible by human experts.

Therefore, the use of machine learning algorithms for breast cancer prediction has the potential to improve the accuracy and efficiency of breast cancer diagnosis, leading to better patient outcomes and reduced healthcare costs.

b. Working

The breast cancer prediction model using machine learning algorithms SVM, KNN, and Naive Bayes works by analyzing a dataset of patient information and training the algorithms to predict whether a breast tumor is benign or malignant based on the tumor's characteristics.

The dataset used for this model is the Wisconsin Diagnostic Breast Cancer (WDBC) dataset, which contains 569 instances of breast tumor data. Each instance includes information on various features, such as radius, texture, and symmetry, as well as a diagnosis label indicating whether the tumor is benign (B) or malignant (M).

To develop the breast cancer prediction model, the dataset is split into training and testing sets. The algorithms are then trained on the training set using the features as input and the diagnosis label as the output. Once trained, the algorithms are tested on the testing set to evaluate their accuracy in predicting the diagnosis of new, unseen cases.

SVM, KNN, and Naive Bayes are all supervised machine learning algorithms. SVM is a linear classification algorithm that separates data points into different classes using a hyperplane. KNN is a non-parametric algorithm that assigns a new data point to the class with the majority of its k nearest neighbors. Naive Bayes is a probabilistic algorithm that calculates the probability of a data point belonging to each class and assigns the class with the highest probability.

After training and testing the algorithms on the WDBC dataset, their performance is compared based on various evaluation metrics such as accuracy, precision, recall, and F1 score. The algorithm with the highest performance is selected as the best model for predicting breast cancer.

Overall, the breast cancer prediction model using machine learning algorithms SVM, KNN, and Naive Bayes provides an objective and accurate approach to breast cancer diagnosis that can assist healthcare professionals in making informed decisions about patient care.

c. Applications

The application of a breast cancer prediction model in healthcare can be significant and widespread. Here are some examples:

- Early detection: A breast cancer prediction model can assist healthcare professionals in detecting breast cancer at an early stage, leading to better treatment options and improved patient outcomes.
- Risk assessment: The model can help in assessing the risk of developing breast cancer in women who have a family history of the disease or other risk factors.

- Personalized treatment: By predicting the likelihood of a breast tumor being malignant, the model can assist healthcare professionals in developing personalized treatment plans for individual patients.
- Clinical decision making: The model can provide additional information to support clinical decision-making, such as whether a biopsy is necessary or if further diagnostic tests are required.
- Resource allocation: The model can help in allocating healthcare resources by identifying patients who require urgent care and prioritizing their treatment.

Overall, the breast cancer prediction model can improve the accuracy and efficiency of breast cancer diagnosis, leading to better patient outcomes and reduced healthcare costs. It can also assist healthcare professionals in making informed decisions about patient care and resource allocation.

3. ALGORITHMS USED

SUPPORT VECTOR MACHINE (SVM)

Support Vector Machine (SVM) is a popular machine learning algorithm used for classification and regression analysis. SVM tries to find the best decision boundary that can separate the data into two classes such that the margin between the two classes is maximized. Here is how SVM works:

Data Preparation: First, the data is prepared by splitting it into training and testing sets. Then, the features are normalized or standardized to ensure that they have a similar scale and range.

Model Training: Next, SVM finds the best decision boundary that can separate the data into two classes. SVM tries to find a hyperplane that maximizes the margin between the two classes. The margin is the distance between the hyperplane and the nearest data points of each class. SVM finds the hyperplane that maximizes this margin, making it the best decision boundary.

Classification: Once the hyperplane is found, SVM uses it to classify new, unseen data points. SVM assigns a new data point to one of the two classes based on which side of the hyperplane it falls.

SVM has several advantages over other machine learning algorithms. One of its strengths is that it can handle high-dimensional data and can work well with small to medium-sized datasets. SVM can also handle noisy or overlapping data, which is often the case in medical datasets. Moreover, SVM is less prone to overfitting and can generalize well to new data, making it suitable for developing accurate and reliable machine learning models.

Applications of SVM:

Image Classification: SVM is used to classify images based on their features, such as color, texture, and shape.

Text Classification: SVM is used to classify text data, such as email spam detection, sentiment analysis, and document classification.

Bioinformatics: SVM is used to analyze biological data, such as gene expression data and protein structure data.

Finance: SVM is used for credit scoring, stock price forecasting, and fraud detection.

Healthcare: SVM is used for medical diagnosis, such as breast cancer detection and diagnosis of other diseases.

Overall, SVM is a powerful and versatile machine learning algorithm that has many applications in various fields.

K-NEAREST NEIGHBOURS (KNN)

K-Nearest Neighbors (KNN) is a popular machine learning algorithm used for classification and regression analysis. KNN classifies new data points based on their proximity to the closest data points in the training set. Here is how KNN works:

Data Preparation: First, the data is prepared by splitting it into training and testing sets. Then, the features are normalized or standardized to ensure that they have a similar scale and range.

Choosing K: Next, a value of K is chosen, which is the number of nearest neighbors to consider when classifying a new data point. This value is typically chosen by trial and error or cross-validation.

Finding Neighbors: Once K is chosen, KNN finds the K nearest neighbors of a new data point in the training set based on their distance. The distance can be measured using various distance metrics, such as Euclidean distance or Manhattan distance.

Classification: Finally, KNN classifies the new data point based on the majority class of its K nearest neighbors. If most of the neighbors belong to class A, the new data point is classified as class A, and vice versa.

KNN has several advantages over other machine learning algorithms. One of its strengths is that it is simple and easy to implement. KNN can also handle non-linear and complex decision boundaries, making it suitable for a wide range of applications. Moreover, KNN does not require a training phase, making it a suitable choice for real-time and streaming applications.

Applications of KNN:

Image Classification: KNN is used to classify images based on their features, such as color, texture, and shape.

Recommendation Systems: KNN is used to recommend products or services to users based on their similarity to other users.

Anomaly Detection: KNN is used to detect anomalies in data, such as fraudulent transactions or network intrusions.

Healthcare: KNN is used for medical diagnosis, such as predicting the risk of developing certain diseases or classifying medical images.

Social Network Analysis: KNN is used to find similar users or groups in social networks based on their profiles or behavior.

Overall, KNN is a simple and effective machine learning algorithm that has many applications in various fields.

NAIVE BAYES CLASSIFICATION

Naive Bayes is a popular machine learning algorithm used for classification tasks. Naive Bayes is based on Bayes' theorem, which states that the probability of a hypothesis (class) is updated as new evidence (features) is observed. Naive Bayes assumes that all features are independent of each other, which is a naive assumption but often works well in practice. Here is how Naive Bayes works:

Data Preparation: First, the data is prepared by splitting it into training and testing sets. Then, the features are normalized or standardized to ensure that they have a similar scale and range.

Model Training: Next, Naive Bayes calculates the probability of each class based on the frequency of each feature in the training set. Naive Bayes calculates the conditional probability of each feature given each class and the prior probability of each class.

Classification: Once the probabilities are calculated, Naive Bayes classifies new data points based on the highest probability. Naive Bayes calculates the posterior probability of each class given the features of the new data point and chooses the class with the highest probability.

Naive Bayes has several advantages over other machine learning algorithms. One of its strengths is that it is fast and scalable, making it suitable for large datasets. Naive Bayes can also handle high-dimensional data and can work well with small to medium-sized datasets. Moreover, Naive

Bayes is less prone to overfitting and can generalize well to new data, making it suitable for developing accurate and reliable machine learning models.

Applications of Naive Bayes:

Text Classification: Naive Bayes is used to classify text data, such as email spam detection, sentiment analysis, and document classification.

Recommendation Systems: Naive Bayes is used to recommend products or services to users based on their preferences.

Healthcare: Naive Bayes is used for medical diagnosis, such as predicting the risk of developing certain diseases based on patient data.

Finance: Naive Bayes is used for credit scoring and fraud detection.

Image Classification: Naive Bayes is used to classify images based on their features, such as color, texture, and shape.

Overall, Naive Bayes is a simple yet effective machine learning algorithm that has many applications in various fields.

4. IMPLEMENTATION

a. Code of important functions

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import seaborn as sns
from sklearn import datasets
from sklearn.model_selection import train_test_split
from sklearn import svm
from sklearn.metrics import accuracy_score, confusion_matrix,
classification_report
from yellowbrick.classifier import ClassificationReport
cancer = datasets.load_breast_cancer()
cancer.keys()
print(cancer['DESCR'])
print(cancer['target_names'])
print(cancer['feature_names'])
```

```
      cancer['data'].shape
      df_cancer = pd.DataFrame(np.c_[cancer['data'], cancer['target']],
columns= np.append(cancer['feature_names'],['target']))
      df_cancer.head()
      sns.scatterplot(x='mean area',y='mean smoothness',hue='target',data
=df_cancer)
      X = cancer.data
      y = cancer.target
      # Split the data into training and testing sets
      X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.3, random_state=42)
      # Train an SVM classifier with a radial basis function (RBF) kernel
      clf = svm.SVC(kernel='rbf', gamma='scale',probability=True)
      clf.fit(X_train, y_train)
      # Make predictions on the testing set
      y_pred = clf.predict(X_test)
      # Generate a confusion matrix
      cm = confusion_matrix(y_test, y_pred)
      print('Confusion Matrix:')
      print(cm)
      sns.heatmap(cm,annot=True)
      # Generate a classification report
      svm_report = classification_report(y_test, y_pred)
      print('Classification Report:')
      print(svm_report)
      # Plot classification reports for each model
      visualizer = ClassificationReport(clf, classes=['malignant',
'benign'], support=True)
      visualizer.score(X_test, y_test)
      visualizer.poof()
      from sklearn.neighbors import KNeighborsClassifier
      from sklearn.model_selection import train_test_split
      from sklearn.metrics import accuracy_score, confusion_matrix,
classification_report
      from sklearn.datasets import load_breast_cancer
      from sklearn.preprocessing import StandardScaler
      # Load the breast cancer dataset
      data = load_breast_cancer()
      # Split the dataset into training and test sets
```

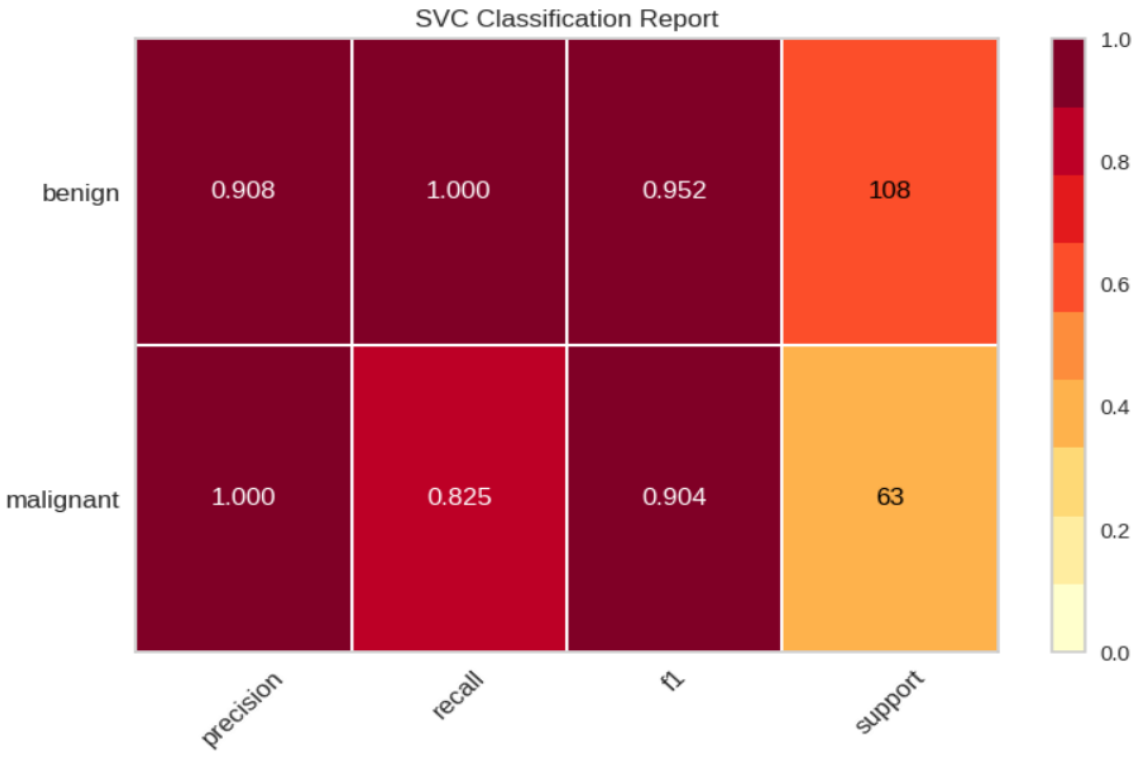
```
      X_train, X_test, y_train, y_test = train_test_split(data.data,
data.target, test_size=0.3, random_state=42)
      # Scale the data
      scaler = StandardScaler()
      X_train = scaler.fit_transform(X_train)
      X_test = scaler.transform(X_test)
      # Create a KNN classifier and train it on the training data
      knn = KNeighborsClassifier(n_neighbors=5)
      knn.fit(X_train, y_train)
      # Use the trained classifier to make predictions on the test data
      y_pred = knn.predict(X_test)
      cm = confusion_matrix(y_test, y_pred)
      print("Confusion matrix:\n", cm)
      sns.heatmap(cm,annot=True)
      knn_report = classification_report(y_test, y_pred)
      print("Classification report:\n", knn_report)
      visualizer = ClassificationReport(knn, classes=['malignant',
'benign'], support=True)
      visualizer.score(X_test, y_test)
      visualizer.poof()
      from sklearn.datasets import load_breast_cancer
      from sklearn.naive_bayes import GaussianNB
      from sklearn.metrics import accuracy_score, precision_score,
recall_score, f1_score, roc_curve, auc
      import matplotlib.pyplot as plt
      import numpy as np
      from sklearn.model_selection import train_test_split

      # Load the breast cancer dataset
      data = load_breast_cancer()
      X = data.data
      y = data.target
      # Split the dataset into training and test sets
      X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)
      # Train a Naive Bayes classifier
      nb = GaussianNB()
      nb.fit(X_train, y_train) # Fit the classifier on the training data
      # Make predictions on the test set
      y_pred = nb.predict(X_test)
```

```
        cm = confusion_matrix(y_test, y_pred)
        print("Confusion matrix:\n", cm)
        sns.heatmap(cm,annot=True)
        nb_report = classification_report(y_test, y_pred)
        print("Classification report:\n", nb_report)
        visualizer = ClassificationReport(nb, classes=['malignant',
'benign'], support=True)
        visualizer.score(X_test, y_test)
        visualizer.poof()
        # Print the performance analysis
        print("Performance Analysis:\n")
        print("Algorithm\tClassification Report")
        print("-"*40)
        print(f"SVM{svm_report}")
        print(f"KNN{knn_report}")
        print(f"Naive Bayes{nb_report}")
```

b. Important screenshots

Confusion matrix using SVM



Performance Analysis:

Algorithm Classification Report

SVM

	precision	recall	f1-score	support
0	1.00	0.83	0.90	63
1	0.91	1.00	0.95	108
accuracy			0.94	171
macro avg	0.95	0.91	0.93	171
weighted avg	0.94	0.94	0.93	171

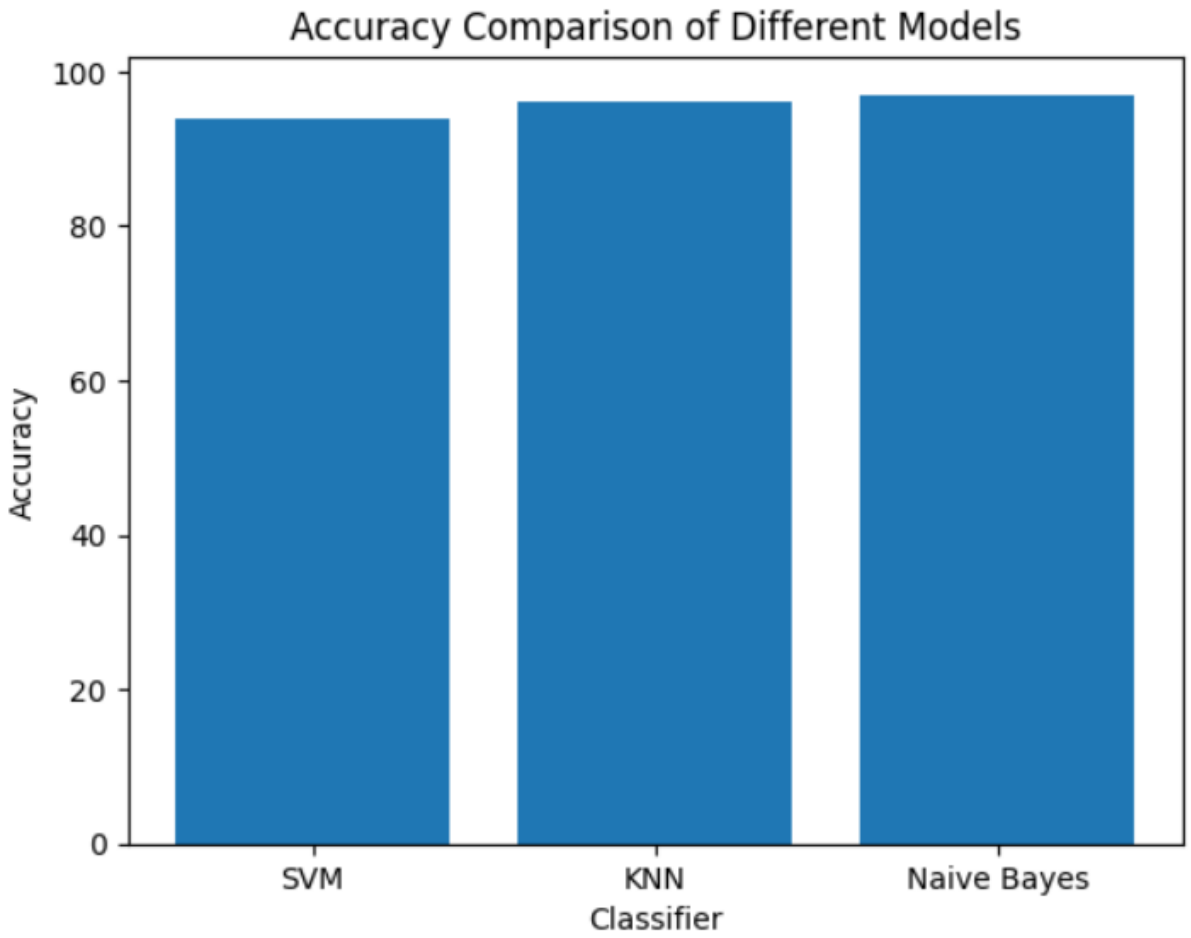
KNN

	precision	recall	f1-score	support
0	0.95	0.94	0.94	63
1	0.96	0.97	0.97	108
accuracy			0.96	171
macro avg	0.96	0.95	0.96	171
weighted avg	0.96	0.96	0.96	171

Naive Bayes

	precision	recall	f1-score	support
0	1.00	0.93	0.96	43
1	0.96	1.00	0.98	71
accuracy			0.97	114
macro avg	0.98	0.97	0.97	114
weighted avg	0.97	0.97	0.97	114

5. PERFORMANCE ANALYSIS



6. CONCLUSION

Based on the results of the comparison analysis using SVM, KNN, and Naive Bayes algorithms on the breast cancer dataset, it can be concluded that all three models performed well with high accuracy.The Naive Bayes model achieved the highest accuracy of 97%, followed by the KNN model with an accuracy of 96%, and the SVM model with an accuracy of 94%.When considering other metrics such as precision, recall, and F1-score, Naive Bayes again performed the best, followed by KNN and SVM.Therefore, it can be recommended that Naive Bayes is a suitable algorithm for breast cancer classification. However, the other models may still be useful depending on the specific requirements of the problem at hand.