Breast Cancer Classification using K-Nearest Neighbors (KNN)

Abstract

Breast cancer is one of the most common and life-threatening diseases affecting women worldwide. Early diagnosis plays a crucial role in improving survival rates. In this project, we develop a machine learning-based classification model using the K-Nearest Neighbors (KNN) algorithm to predict whether a breast cancer tumor is malignant (positive) or benign (negative). The model is trained on the Wisconsin Breast Cancer dataset and evaluated using cross-validation, confusion matrix, and ROC-AUC score. The experimental results show that the KNN classifier achieves high accuracy, demonstrating its potential as a decision-support tool in medical diagnosis.

Problem Statement

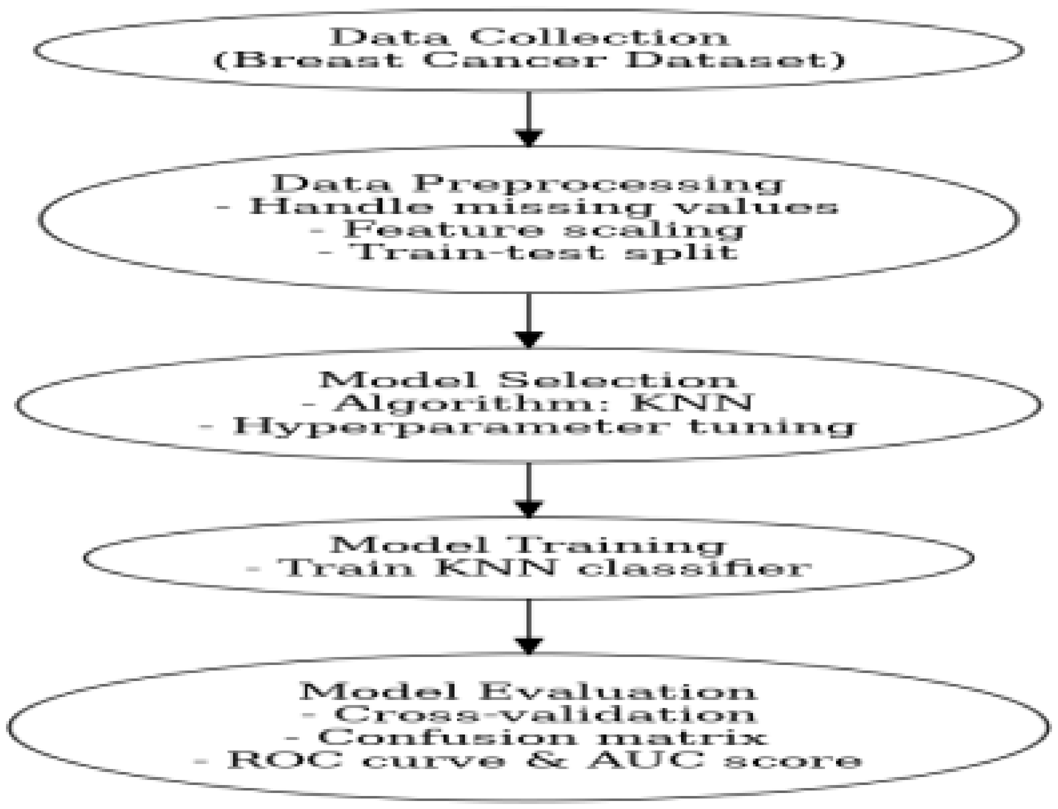
The challenge is to build an efficient and reliable classification model that can distinguish between malignant and benign breast tumors using clinical and imaging data. Traditional methods rely on manual diagnosis, which may lead to delays and inaccuracies. A machine learning-based approach can automate prediction and assist oncologists in decision-making, thereby reducing human error and improving accuracy.

Motivation

Breast cancer is a major global health concern, and early detection plays a crucial role in improving survival rates and treatment outcomes. Traditional methods of manual diagnosis, although effective, can be time-consuming and prone to human error or misinterpretation, which may delay treatment decisions. In this context, machine learning offers a promising approach to developing automated, accurate, and efficient decision-support systems that can assist medical professionals in diagnosis. Among various algorithms, K-Nearest Neighbors (KNN) stands out as a simple yet powerful supervised learning method capable of classifying tumors based on similarity in feature space, thereby providing a reliable and effective tool for breast cancer prediction.

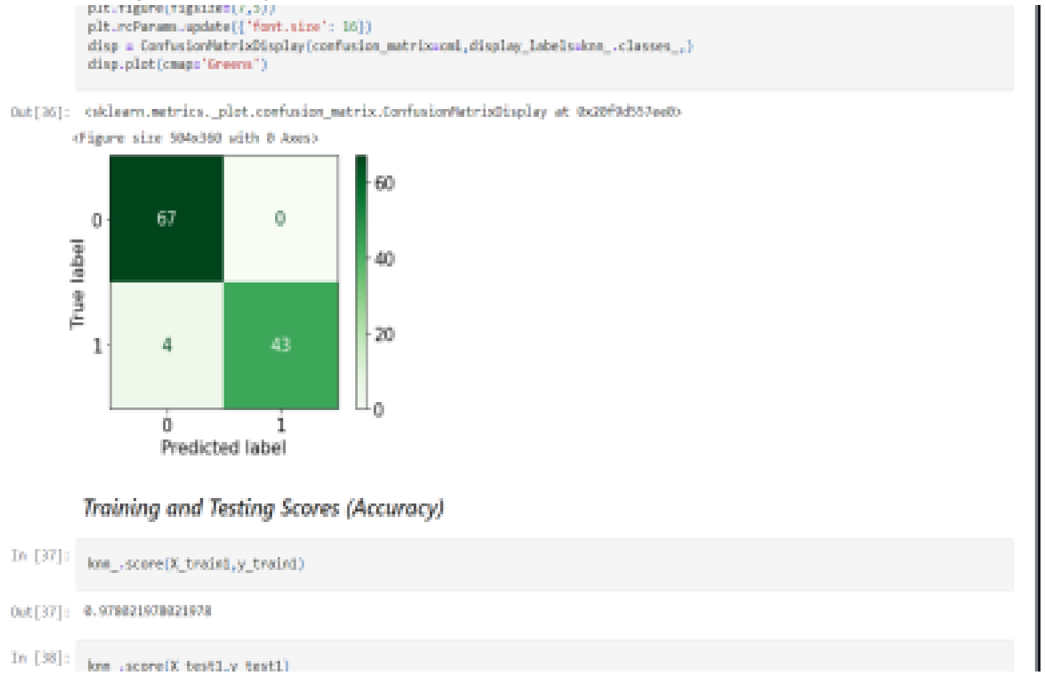
System Architecture

The system architecture for breast cancer classification using K-Nearest Neighbors (KNN) follows a structured pipeline. The process begins with data collection and preprocessing, where the Wisconsin Breast Cancer dataset is used as input. Preprocessing steps include handling missing values, applying feature scaling to normalize the data, and splitting the dataset into training and testing subsets. In the model selection stage, the KNN algorithm is chosen as the primary classifier, and hyperparameter tuning is carried out by varying the number of neighbors (K) to identify the optimal value for improved performance. Following this, the model training phase involves fitting the KNN classifier to the training dataset. Finally, the system performance is assessed in the evaluation phase using multiple metrics, including cross-validation accuracy, confusion matrix, and ROC curve with AUC score, to measure accuracy, precision, and overall classification effectiveness.



Results & Outputs

The results of the KNN classifier on the Breast Cancer dataset are summarized below: 1. Cross-validation accuracy: The best performance was achieved around K = 10–15 with accuracy ≈ 93%. 2. Confusion Matrix: The model correctly classified 67 true negatives, 43 true positives, with only 4 false negatives and 0 false positives. 3. Training Accuracy: 97.8% 4. Testing Accuracy: 96.4% 5. ROC Curve and AUC: The ROC curve achieved an AUC score of 0.96, showing excellent discrimination capability. These results demonstrate that the KNN classifier performs effectively on the dataset with high accuracy and low error rate.



```
plt.figure(figsize=(7,5))
plt.rcParams.update({'font.size': 16})
disp = ConfusionMatrixDisplay(confusion_matrix=cm1,display_labels=knn_.classes_,)
disp.plot(cmap='Greens')
```

```
Out[36]: <sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x20f9d557ee0>
<Figure size 504x360 with 0 Axes>
```

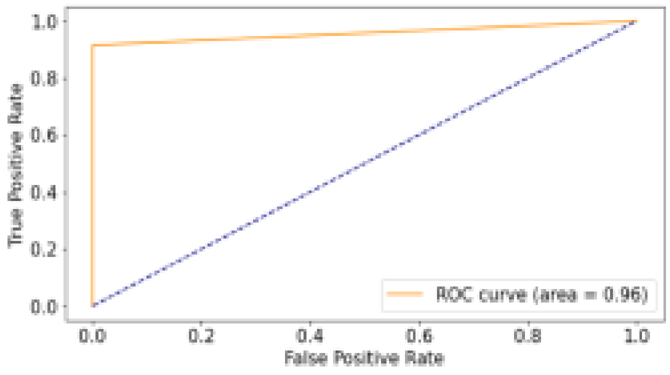
Training and Testing Scores (Accuracy)

```
In [37]: knn_.score(X_train1,y_train1)
```

```
Out[37]: 0.978021978021978
```

```
In [38]: knn_.score(X_test1,y_test1)
In [40]: fpr_, tpr_, thresholds_ = roc_curve(y_test1, y_pred_2)
         roc_auc1 = auc(fpr_, tpr_)

         # Plot ROC curve
         plt.figure(figsize=(10,5))
         plt.plot(fpr_, tpr_, color='darkorange', label='ROC curve (area = %0.2f)' % roc_auc1)
         plt.plot([0, 1], [0, 1], color='navy', linestyle='--')
         plt.xlabel('False Positive Rate')
         plt.ylabel('True Positive Rate')
         plt.legend(loc="lower right")
         plt.show()
```

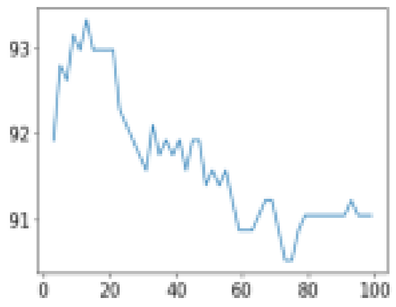


```
ROC Curve Area = 0.96, which is determined as from ROC-AUC score as well
```

```
for i in k_range:
    knn_model = KNeighborsClassifier(n_neighbors=i)
    #5 Fold Cross Validation
    scores = cross_val_score(knn_model, X1, y1,cv = 5, scoring = 'accuracy')
    scores = (scores*100).round(4)
    k_scores.append(scores.mean())
```

```
In [43]: sns.lineplot(x=k_range,y=k_scores)
         #For Cross Validation in KNN

Out[43]: <AxesSubplot:>
```



```
In [ ]:
```

Conclusion

The KNN classifier achieved high accuracy (96% testing accuracy and 0.96 ROC-AUC score) on breast cancer classification. The system demonstrates strong potential as a clinical decision support tool to assist doctors in early breast cancer diagnosis. Future work can include integrating advanced algorithms like Random Forest, SVM, or Deep Learning for improved performance and scalability.

Code :

```
import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

df = pd.read_csv(r'C:\Users\KIIT\OneDrive\Documents\Analytics and ML\Breastcancer classifier
(Wisconsin)\data.csv')

df.head(10)

#Drop the unnecessary columns.

col = ['id','Unnamed: 32']

df = df.drop(col,axis=1)

df

#Description of statistics from dataset

df.describe()

df.head(5)

from sklearn.preprocessing import LabelEncoder


diagnosis_y = LabelEncoder()
df['diagnosis'] = diagnosis_y.fit_transform(df['diagnosis'])

from sklearn.model_selection import train_test_split

from sklearn.preprocessing import StandardScaler

from sklearn.metrics import confusion_matrix, roc_auc_score, roc_curve,
auc

from sklearn.metrics import ConfusionMatrixDisplay


df

X1 = df.drop('diagnosis',axis=1)

y1 = df['diagnosis']

X1

y1

X_train1, X_test1, y_train1, y_test1 =
train_test_split(X1,y1,test_size=0.2,random_state=0)


X_train1.shape

X_test1.shape

y_train1.shape, y_test1.shape

from sklearn.neighbors import KNeighborsClassifier


knn_1 = KNeighborsClassifier(n_neighbors=5) #no. of nearest neighbours = 5


scale1 = StandardScaler()

X_train1 = scale1.fit_transform(X_train1)

X_test1 = scale1.fit_transform(X_test1)

X_train1

X_test1

knn_ = knn_1.fit(X_train1,y_train1)
```

```
y_pred1 = knn_.predict(X_test1)

from sklearn.metrics import accuracy_score

val = accuracy_score(y_pred1,y_test1)
print(f"Accuracy for KNN with K = 5, {val*100:.4f}")

Accuracy for KNN with K = 5, 96.4912
```

```
cm1 = confusion_matrix(y_test1,y_pred1)

#HeatMap for Confusion Matrix
plt.figure(figsize=(7,5))
plt.rcParams.update({'font.size': 16})
disp = ConfusionMatrixDisplay(confusion_matrix=cm1,display_labels=knn_.classes_,)
disp.plot(cmap='Greens')
knn_.score(X_train1,y_train1)
knn_.score(X_test1,y_test1)
y_pred_2 = knn_.predict(X_test1)

# AUC score for the binary classification problem
auc_score1 = roc_auc_score(y_test1, y_pred_2)

print("AUC Score:", auc_score1)

AUC Score: 0.9574468085106382
```

```
fpr_, tpr_, thresholds_ = roc_curve(y_test1, y_pred_2)
roc_auc1 = auc(fpr_, tpr_)

# Plot ROC curve
plt.figure(figsize=(10,5))
plt.plot(fpr_, tpr_, color='darkorange', label='ROC curve (area = %0.2f)' % roc_auc1)
plt.plot([0, 1], [0, 1], color='navy', linestyle='--')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.legend(loc="lower right")
plt.show()
from sklearn.model_selection import cross_val_score
```

```python
k_range = range(3,100,2)
k_scores = []
for i in k_range:
    knn_model = KNeighborsClassifier(n_neighbors=i)
    #5 Fold Cross Validation
    scores = cross_val_score(knn_model, X1, y1,cv = 5, scoring = 'accuracy')
    scores = (scores*100).round(4)
    k_scores.append(scores.mean())

sns.lineplot(x=k_range,y=k_scores)
#For Cross Validation in KNN
```

References

- Wisconsin Breast Cancer Dataset (UCI Repository) - Scikit-learn Documentation - Seaborn and Matplotlib for visualization