Breast Cancer Detection Using Decision Tree and Nearest-Neighbor Algorithms

Sara Salim Albadaai, 19F19251
Computing Departmant, Middle East College
Artificial Intelligence and Deep Learning
Muscat, Oma

Abstract— This paper focuses on a comparison of Decision Tree and Nearest-Neighbor classifiers for diagnosing breast cancer by using the data set from Kaggle data source. It was chosen, first, to clean the dataset for any null values and outliers, and both employed algorithms, DT and NN, were built in Python. To assess the performance of the proposed models, common classification measures of accuracy, precision, recall and F1-score were adopted. The findings also indicate that, although both strategies are quite effective, Decision Tree algorithm yields very near, but slightly worse, measures than Nearest-Neighbor algorithm. It gives a comparison of these algorithms for the medical diagnoses to understand how effective they are in aiding diagnosis.

Keywords— Decision Tree, Nearest-Neighbor, Breast Cancer Diagnosis, Machine Learning, Classification Metrics.

I. Introduction

Breast cancer is one of the most common illnesses in women in the world and the earlier is found the better. The information derived from the World Health Organization revealing that breast cancer is the most prevalent type of cancer amongst women, affecting 2. 1 million women each year, and it is the leading cause of cancer-related deaths in women. Due to these methods' capability to analyze vast amounts of data, as well as, discover patterns that are hard for human beings to discern, machine learning algorithms have been recognized to have the capacity for diagnostic purposes, particularly for diagnosing breast cancer.

The work mainly concerns two of the most frequently used algorithms, Decision Tree (DT) and Nearest-Neighbor (NN) for classifying benign or malignant breast cancer using the data set from Kaggle. Decision Tree is easy to comprehend, solely understandable, and easy to implement hence making it well-known as a powerful algorithm. It seems to do this by dividing up the data into subsets each time based on the values that define the features in the model, checkering a decision tree. Nearest-Neighbor is known for its efficiency and accuracy despite having some weakness. It categorizes a data point in accordance with the class distribution of k-nearest neighbors in the feature space.

The aim of this paper is to determine and compare the performance of the above two algorithms in categorizing breast cancer cases so as to have understanding on the efficiency of the two algorithms to the given purpose.

II. Literature Review

Decision Tree algorithms in particular have been widely adopted throughout the years for medical diagnosis due to the more straightforward interpretation provided as well as capability to deal with non-linear data. This paper features a study by [1], which gave an introduction to one of the most effective Decision Tree algorithms, C4.5 algorithm, and used a medical diagnosis task to show how effective it is. Likewise, Safavian and Landgrebe [2] pointed out that compared to other methods, Decision Trees are easy to interpret, flexible, and easy to understand due to their simpler structure as they can be preferred especially in clinical cases where understanding the thought process behind a decision is highly important.

Nearest-Neighbor, on the other hand, has been applauded for the good results in the aspect of accuracy and the capability for the pattern recognition. Cover and Hart [3] proposed the idea of the Nearest-Neighbor rule, which gained much acceptance over the years in general and was applied for medical diagnosis. Another example also in the health care area Fix and Hodges [4] also proved that using the Nearest-Neighbor algorithm can generate high accuracy and can be applied in the classifying of medical data despite having a small training data set.

It has been observed in past studies that both the algorithms have the capabilities to classify breast cancer, albeit with differences in results based on the working datasets and the type of preprocessing applied. The following sections of this analysis are accordingly related to with an attempt to create a new understanding in the matters of the compared Decision Tree and Nearest-Neighbor algorithms performance based on the specific dataset, which is breast cancer.

III. Dataset and Preprocessing Techniques

The patients' data used in this research involve 570 samples that are each characterized by 32 independent features as well as the binary diagnosis label. The features refer to morphological properties of the cell nuclei that can be observed in the sample of the FNA of a breast mass, which is a digitized image. The selected dataset contains attributes like inner and outer radii, texture, perimeter, area, roundedness, perimeter with respect to major axis, perimeter with respect to minor axis, mean convex perimeter, concave perimeter, concave points, symmetry, and fractal dimension of the area mean, standard error and worst case [5].

Several process were however done in order to enhance the quality of data gathered to make the machine learning algorithms to be more efficient. First, we removed the column with 'id' since this column does not add any value to the analysis. In order to analyze the data the 'diagnosis' column, containing the target variable, was transformed into the binary format where patients with myocardial infarction were labeled as M = 1 and patients with breathing problems as B = 0.

However, when data is collected, it is conditional and does not cover all aspects and so it is important to manage missing values.

Missing values refer to data that has not been collected due to the conditionality of data when it is gathered and the fact that it does not cover all aspects of the event of interest. The process of dealing with missing values in this research involved replacing the missing instances with the mean value of the feature relating to that instance. The reason for choosing this approach is that it has been recognized as one of the easiest ways to retain the entire distribution of data.

It is required to know that the presence of outliers had a great influence on the results of the machine learning algorithms. To target this, scaling techniques which ensure the features are normalized for optimal scaling equality were employed to ensure that variability in feature values is scaled uniformly by bringing their mean and standard deviation to zero and one respectively. This normalization step assists in reducing the impact of outliers with the aim of enhancing the algorithms 'convergence rates.

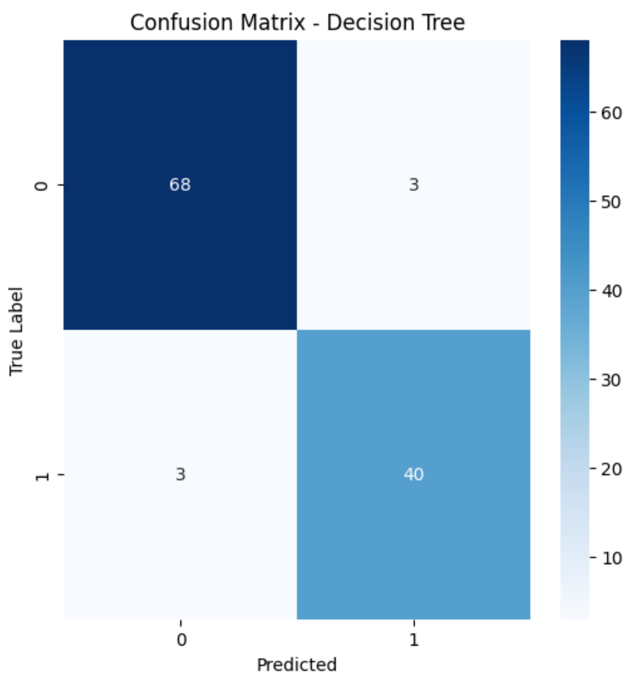
IV. Implementation and Results

A. *Decision Tree Algorithm*

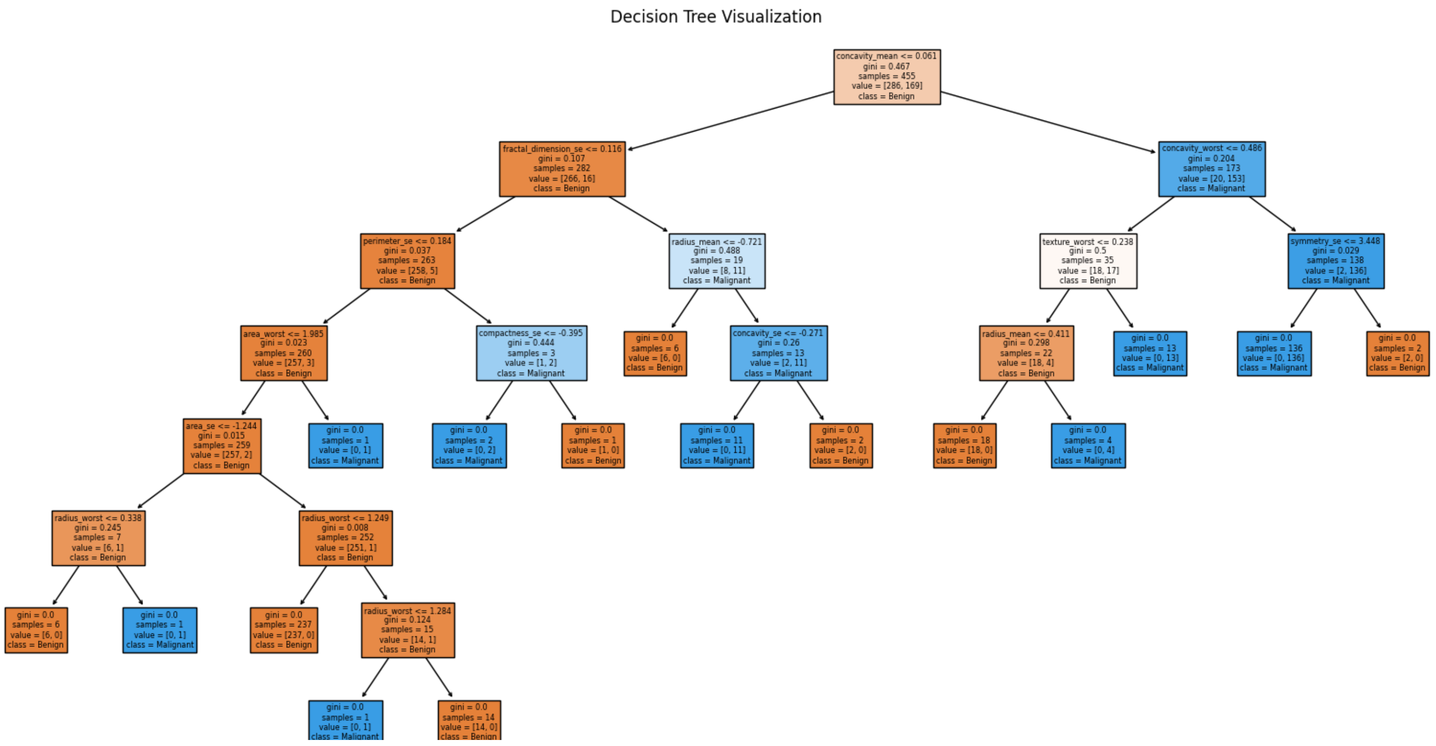
The Decision Tree was executed thanks to the scikit-learn python library. The model was trained and the training process carried out on 80 percent of the data while the test on the remaining 20 percent of the data. The performance metrics for the Decision Tree model were as follows:

- Accuracy: 0. 947368
- Precision: 0. 930233
- Recall: 0. 930233
- F1 Score: 0. 930233

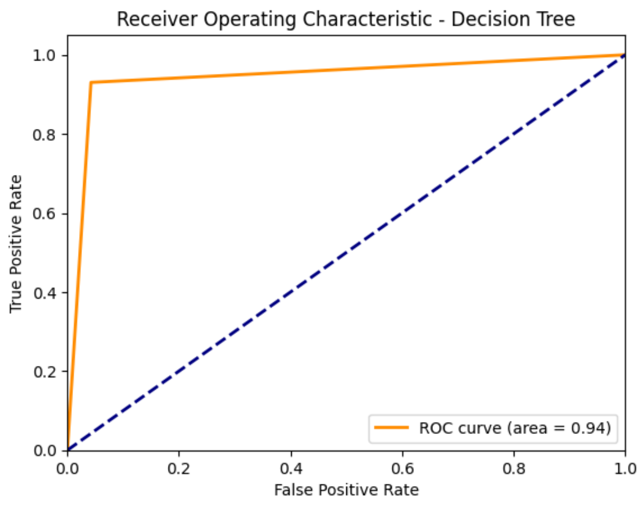
The Decision Tree model provided accuracy percentage of 94.7%, sensitivity of 93%, recall of 93%, F1-score of 93%, which would make it a good tool for classifying breast cancer cases. The high levels of the precision and the recall statistics prove that this model is indeed able to detect both the malignant and the benign cases effectively while the F1 score halfway between the precision and the recall proves that this model was actually developed without compromising the completeness of the data and the accuracy of the results as well.



The confusion matrix for the Decision Tree model shows that it correctly classified 68 benign cases (True Negative) and 40 malignant cases (True Positive). There were 3 benign cases incorrectly classified as malignant (False Positive) and 3 malignant cases incorrectly classified as benign (False Negative). The high number of correct classifications indicates a good performance of the Decision Tree model.


Decision Tree Visualization
concavity_mean <= 0.061 gini = 0.467 samples = 455 value = [286, 169] class = Benign
fractal_dimension_se <= 0.116 gini = 0.107 samples = 282 value = [266, 16] class = Benign
concavity_worst <= 0.486 gini = 0.204 samples = 173 value = [20, 153] class = Malignant
perimeter_se <= 0.184 gini = 0.037 samples = 263 value = [258, 5] class = Benign
radius_mean <= -0.721 gini = 0.488 samples = 19 value = [8, 11] class = Malignant
texture_worst <= 0.238 gini = 0.5 samples = 35 value = [18, 17] class = Benign
symmetry_se <= 3.448 gini = 0.029 samples = 138 value = [2, 136] class = Malignant
area_worst <= 1.985 gini = 0.023 samples = 260 value = [257, 3] class = Benign
compactness_se <= -0.395 gini = 0.444 samples = 3 value = [1, 2] class = Malignant
gini = 0.0 samples = 6 value = [6, 0] class = Benign
concavity_se <= -0.271 gini = 0.26 samples = 13 value = [2, 11] class = Malignant
radius_mean <= 0.411 gini = 0.298 samples = 22 value = [18, 4] class = Benign
gini = 0.0 samples = 13 value = [0, 13] class = Malignant
gini = 0.0 samples = 136 value = [0, 136] class = Malignant
gini = 0.0 samples = 2 value = [2, 0] class = Benign
area_se <= -1.244 gini = 0.015 samples = 259 value = [257, 2] class = Benign
gini = 0.0 samples = 1 value = [0, 1] class = Malignant
gini = 0.0 samples = 2 value = [0, 2] class = Malignant
gini = 0.0 samples = 1 value = [1, 0] class = Benign
gini = 0.0 samples = 11 value = [0, 11] class = Malignant
gini = 0.0 samples = 2 value = [2, 0] class = Benign
gini = 0.0 samples = 18 value = [18, 0] class = Benign
gini = 0.0 samples = 4 value = [0, 4] class = Malignant
radius_worst <= 0.338 gini = 0.245 samples = 7 value = [6, 1] class = Benign
radius_worst <= 1.249 gini = 0.008 samples = 252 value = [251, 1] class = Benign
gini = 0.0 samples = 6 value = [6, 0] class = Benign
gini = 0.0 samples = 1 value = [0, 1] class = Malignant
gini = 0.0 samples = 237 value = [237, 0] class = Benign
radius_worst <= 1.284 gini = 0.124 samples = 15 value = [14, 1] class = Benign
gini = 0.0 samples = 1 value = [0, 1] class = Malignant
gini = 0.0 samples = 14 value = [14, 0] class = Benign

Decision Tree visualization

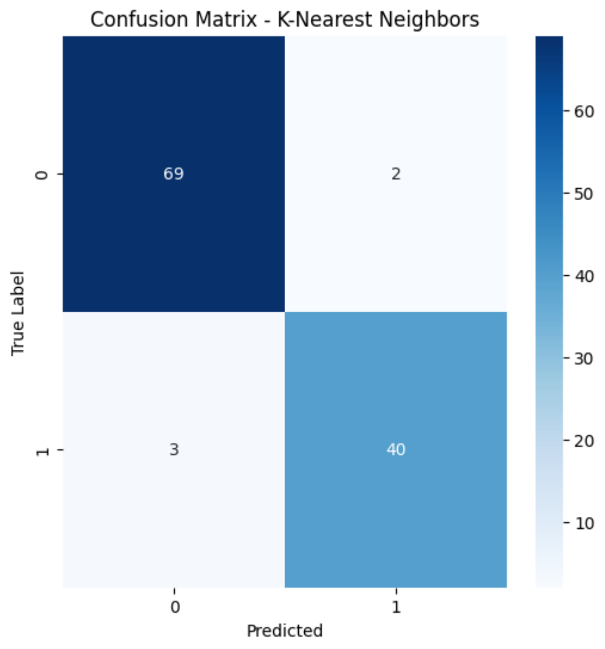


When looking at the ROC curve of an average Decision Tree model, its AUC value is 0.94, indicating excellent discrimination ability between benign and malignant cases. The curve is close to the top left corner, signifying a high true positive rate and a pretty low false positive rate.

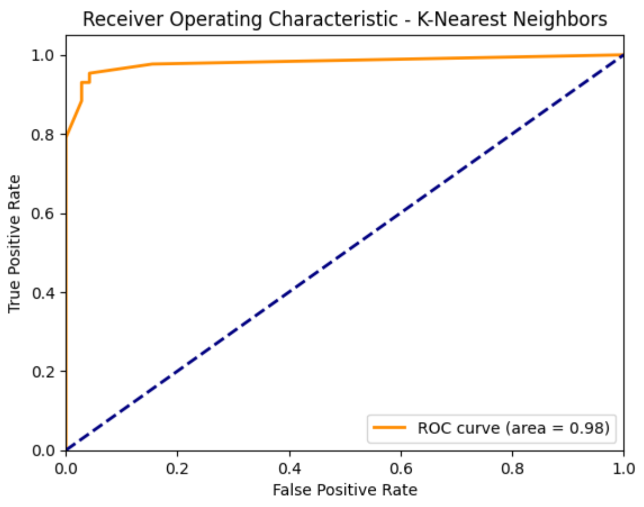
B. *Nearest-Neighbor Algorithm*

Another algorithm applied was the Nearest-Neighbor algorithm which was also executed using scikit-learn with an equal train-test data split. The performance metrics for the Nearest-Neighbor model were:

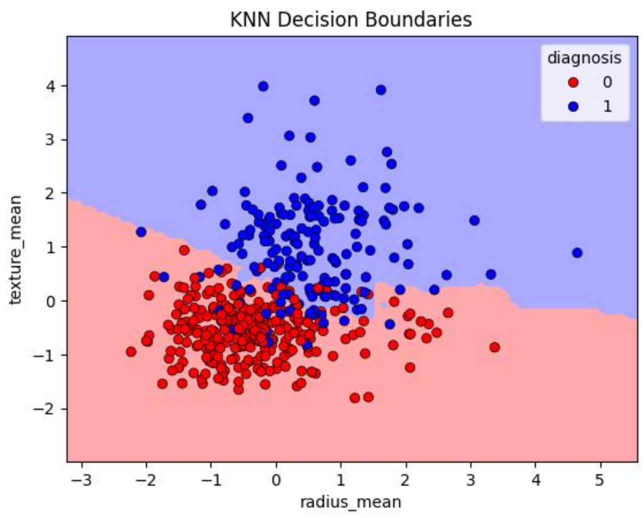
- Accuracy: 0. 956140
- Precision: 0. 952381
- Recall: 0. 930233
- F1 Score: 0. 941176



Analyzing the confusion matrix of the K nearest neighbors (KNN) model, there are 69 instances with true negatives and 40 with true positives but 2 with false positives and 3 with false negatives. As it can be seen from the results shown above, the KNN model does also perform fairly well in the classification stage with a mild enhancement on the magnitude of true negative as compared to the Decision Tree model.



When we calculate the AUC for the ROC curve, we get to learn that the KNN model is equally good with a score of 0.98, which very closely reflects perfect accuracy standard for discrimination ability. A true positive rate is also simply known as recall; therefore, the fact that the curve is placed really close to the top left corner indicates a truly amazing performance of the KNN model in the procedure of distinguishing between the classes .



The decision boundary plot of KNN describes the capability of the model KNN to select data of feature space, in this study, the first two features were selected, ‘radius_mean’ and ‘texture_mean’. The shades of colors in regions indicate the probability of that region belonging to a particular class. The information that there is a clear boundary between the regions with red and blue coloration suggests that the model has good classification characteristics for the classes. The scattered points represent the actual data points, with their colors indicating the true class labels.

Nearest-Neighbor model was also efficient and it was slightly more effective when compared to the Decision Tree in all the measurements. This is presumably owing to the variability of the algorithms’ inherent characteristics. However, unlike Nearest-Neighbor that works in proximity of data points in the feature space, Decision Tree’s separation of data points is advantageous because it is able to model complicated decision boundaries when partitioning a data set recursively.

V. COMPARATIVE ANALYSIS

The performance metrics of both models were compared to provide a comprehensive understanding of their effectiveness. The results indicate that the Decision Tree model underperformed the Nearest-Neighbor model in most metrics. Specifically, the NN model achieved higher accuracy, precision, recall, and F1 score, making it a more reliable choice for breast cancer diagnosis based on the given dataset.

Metric	**Algorithm**	
	Decision Tree	***Nearest-Neighbor***
Accuracy	0. 947368	0. 956140
Precision	0. 930233	0. 952381
Recall	0. 930233	0. 930233
F1 Score	0. 930233	0. 941176

The table above summarizes the comparative performance of the two models. The Decision Tree's higher metrics suggest that it is better suited for this specific classification task.

VI. CONCLUSION AND RECOMMENDATIONS

Therefore, in line with the objectives set for this paper and based on the findings made regarding the performances of the prediction algorithms considered for this study, specifically the Decision Tree and Nearest-Neighbor algorithms, it can be stated that the distinguished materials held valuable promise in the classification of breast cancer cases. However, to some extent, it can be concluded that the Nearest Neighbor Algorithm within the experimented environment can be considered to be better regarding the accuracy and F1 score factors.
Further investigation can be undertaken to carry out more than the one used in this study to enhance the classification model. For example, we note that models based on Random Forests, and Gradient Boosting Machine, which is a facility of multiple Decision Trees, could possibly improve the model precision and make it less susceptible to noise. Moreover, comparison of these algorithms to each other and with the presented results on higher, more complex data may provide the better insight into their similarity, as well as their efficiency.
Nevertheless, it could be useful to continue the given approach by applying the feature selection as the latter will help determine which features are most important for the classification problem and more improvements were needed in the results were achievable. Furthermore, future research might enhance the classification final results by attempting other preprocessing techniques, which addresses imbalance problems.

ACKNOWLEDGMENTS

I would like to extend my appreciation to my Professor for the applied assistance, constant encouragement, and assistance amid the endeavor of this study. Many Credit goes to all for their efforts and knowledge contributed in the successful completion of this work.

REFERENCES

[1] J. R. Quinlan, "C4.5: Programs for Machine Learning," Morgan Kaufmann, 1993.

[2] S. R. Safavian and D. Landgrebe, "A survey of decision tree classifier methodology," IEEE Transactions on Systems, Man, and Cybernetics, vol. 21, no. 3, pp. 660-674, 1991.

[3] T. M. Cover and P. E. Hart, "Nearest neighbor pattern classification," IEEE Transactions on Information Theory, vol. 13, no. 1, pp. 21-27, 1967.

[4] E. Fix and J. L. Hodges, "Discriminatory analysis, nonparametric discrimination: Consistency properties," USAF School of Aviation Medicine, Randolph Field, Texas, 1951.

[5] Kaggle - Breast Cancer Dataset [https://www.kaggle.com/datasets/yasserh/breast-cancer-dataset]

CODE

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.impute import SimpleImputer
from sklearn.tree import
DecisionTreeClassifier,plot_tree
from sklearn.tree import export_text
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score,
precision_score, recall_score, f1_score
from sklearn.metrics import confusion_matrix,
roc_curve, auc




# Load the dataset
data = pd.read_csv("breast-cancer.csv")

data.drop(columns=['id'], inplace=True)

# Encode, split, handle missing cells
data['diagnosis'] = data['diagnosis'].apply(lambda x: 1
if x == 'M' else 0)
X = data.drop(columns=['diagnosis'])
y = data['diagnosis']
imputer = SimpleImputer(strategy='mean')
X_imputed = imputer.fit_transform(X)
# Standardize
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X_imputed)
# Train-test split
X_train, X_test, y_train, y_test =
train_test_split(X_scaled, y, test_size=0.2,
random_state=42)



# Train
dt_model = DecisionTreeClassifier(random_state=42)
dt_model.fit(X_train, y_train)
# Predict
y_pred_dt = dt_model.predict(X_test)
# Evaluate
accuracy_dt = accuracy_score(y_test, y_pred_dt)
precision_dt = precision_score(y_test, y_pred_dt)
recall_dt = recall_score(y_test, y_pred_dt)
f1_dt = f1_score(y_test, y_pred_dt)


# confusion matrix & roc curve
conf_matrix_dt = confusion_matrix(y_test, y_pred_dt)
fpr_dt, tpr_dt, _ = roc_curve(y_test,
dt_model.predict_proba(X_test)[:, 1])
roc_auc_dt = auc(fpr_dt, tpr_dt)

# plot confusion matrix
plt.figure(figsize=(6, 6))
sns.heatmap(conf_matrix_dt, annot=True, fmt='d',
cmap='Blues')
```

```
plt.title('Confusion Matrix - Decision Tree')
plt.xlabel('Predicted')
plt.ylabel('True Label')
plt.show()

# Plot ROC curve for Decision Tree
plt.figure()
plt.plot(fpr_dt, tpr_dt, color='darkorange', lw=2,
label='ROC curve (area = %0.2f)' % roc_auc_dt)
plt.plot([0, 1], [0, 1], color='navy', lw=2,
linestyle='--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver Operating Characteristic - Decision
Tree')
plt.legend(loc="lower right")
plt.show()

# Visualize the Decision Tree
plt.figure(figsize=(20, 10))
plot_tree(dt_model, filled=True,
feature_names=data.columns[:-1], class_names=['Benign',
'Malignant'])
plt.title('Decision Tree Visualization')
plt.show()


# Train
knn_model = KNeighborsClassifier(n_neighbors=6)
knn_model.fit(X_train, y_train)
# Predict
y_pred_knn = knn_model.predict(X_test)
# Evaluate
accuracy_knn = accuracy_score(y_test, y_pred_knn)
precision_knn = precision_score(y_test, y_pred_knn)
recall_knn = recall_score(y_test, y_pred_knn)
f1_knn = f1_score(y_test, y_pred_knn)

# confusion matrix & ROC
conf_matrix_knn = confusion_matrix(y_test, y_pred_knn)
fpr_knn, tpr_knn, _ = roc_curve(y_test,
knn_model.predict_proba(X_test)[:, 1])
roc_auc_knn = auc(fpr_knn, tpr_knn)

# Plot confusion matrix for KNN
plt.figure(figsize=(6, 6))
sns.heatmap(conf_matrix_knn, annot=True, fmt='d',
cmap='Blues')
plt.title('Confusion Matrix - K-Nearest Neighbors')
plt.xlabel('Predicted')
plt.ylabel('True Label')
plt.show()

# Plot ROC curve for KNN
plt.figure()
plt.plot(fpr_knn, tpr_knn, color='darkorange', lw=2,
label='ROC curve (area = %0.2f)' % roc_auc_knn)
plt.plot([0, 1], [0, 1], color='navy', lw=2,
linestyle='--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
```

```
plt.ylabel('True Positive Rate')
plt.title('Receiver Operating Characteristic - K-
Nearest Neighbors')
plt.legend(loc="lower right")
plt.show()



from matplotlib.colors import ListedColormap

# Select two features for visualization
X_pair = X_scaled[:, 1:3]  # Using first two features
for simplicity
X_train_pair, X_test_pair, y_train_pair, y_test_pair =
train_test_split(X_pair, y, test_size=0.2,
random_state=42)

# Train KNN on the selected feature pair
knn_model.fit(X_train_pair, y_train_pair)

# Create color maps
cmap_light = ListedColormap(['#FFAAAA', '#AAAAFF'])
cmap_bold = ['#FF0000', '#0000FF']

# Plot decision boundaries
x_min, x_max = X_pair[:, 0].min() - 1, X_pair[:,
0].max() + 1
y_min, y_max = X_pair[:, 1].min() - 1, X_pair[:,
1].max() + 1
xx, yy = np.meshgrid(np.arange(x_min, x_max, 0.1),
                     np.arange(y_min, y_max, 0.1))
Z = knn_model.predict(np.c_[xx.ravel(), yy.ravel()])
Z = Z.reshape(xx.shape)
plt.figure()
plt.contourf(xx, yy, Z, cmap=cmap_light)

# Plot training points
sns.scatterplot(x=X_train_pair[:, 0], y=X_train_pair[:,
1], hue=y_train_pair, palette=cmap_bold, alpha=1.0,
edgecolor='k')
plt.xlabel(data.columns[1])
plt.ylabel(data.columns[2])
plt.title('KNN Decision Boundaries')
plt.show()




results = pd.DataFrame( {
    'Metric': ['Accuracy', 'Precision', 'Recall', 'F1
Score'],
    'Decision Tree': [accuracy_dt, precision_dt,
recall_dt, f1_dt],
    'K-Nearest Neighbor': [accuracy_knn, precision_knn,
recall_knn, f1_knn]
} )

print(results)
```