AI project

Project was made by: Razan Hamdan – Atheer Abduallah Mutaq – Rose Khaled Alamri – Aram Mohammed

Introduction:

Breast cancer is one of the most common cancers worldwide. Early diagnosis is crucial for effective treatment and improved outcomes .

In This project we aims to introduce a solution by giving a diagnosis using a Decision Tree Classifier using Python and Scikit-learn to classify tumors as malignant or benign.

Methodology

Step 1: Import Scikit-learn .

we will install a Python package called Scikit-learn, and Import Scikit-learn's dataset In this tep, we can begin working with the dataset for our machine learning model.

Step 2: load the dataset.

In [1]:
```
#sklearn
#import libraries
import sklearn
from sklearn.datasets import load_breast_cancer
```

In [2]:
```
#load data
data = load_breast_cancer()
```

we can create new variables for each important set of information and assign the data. In other words, we can organize the data with the following commands:

In [3]:
```
#feature and labels
label_names = data['target_names']
labels = data['target']
feature_names = data['feature_names']
features = data['data']
```

Now, the command given below will show that they are mapped to binary values 0 and 1.Here 0 represents malignant cancer and 1 represents benign cancer

In [4]:
```
print(label_names)
```

```
['malignant' 'benign']
```

In [5]:
```
print(labels[0])
```

```
0
```

The following two commands will produce the feature names and feature values.we can see that the first data instance is a malignant tumor the radius of which is 1.7990000e+01.

In [6]:
```
print(feature_names[0])
print(features[0])
```

```
mean radius
[1.799e+01 1.038e+01 1.228e+02 1.001e+03 1.184e-01 2.776e-01 3.001e-01
 1.471e-01 2.419e-01 7.871e-02 1.095e+00 9.053e-01 8.589e+00 1.534e+02
 6.399e-03 4.904e-02 5.373e-02 1.587e-02 3.003e-02 6.193e-03 2.538e+01
 1.733e+01 1.846e+02 2.019e+03 1.622e-01 6.656e-01 7.119e-01 2.654e-01
 4.601e-01 1.189e-01]
```

Step 3: Organizing data into sets .

In this step, we will divide our data into two parts namely a training set and a test set.

To split the data into sets, sklearn has a function called the train_test_split() function.

In [7]:
```
from sklearn.model_selection import train_test_split
```

we are using 20 % of the data for testing and the remaining data would be used for training the model.

In [8]:
```
#split data
train, test, train_labels, test_labels = train_test_split(features,labels,test_size = 0.20, random_state = 42)
```

Step 4: Building the model

import some important libraries

In [9]:
```
import pydotplus
from sklearn import tree
from sklearn.datasets import load_iris
from sklearn.metrics import classification_report
import collections
```

After providing the dataset, we need to fit the model which can be done as follows:

In [10]:
```
#building and trainning the model
dt = tree.DecisionTreeClassifier()
dt = dt.fit(train,train_labels)
```

Prediction can be made with the help of the following Python code:

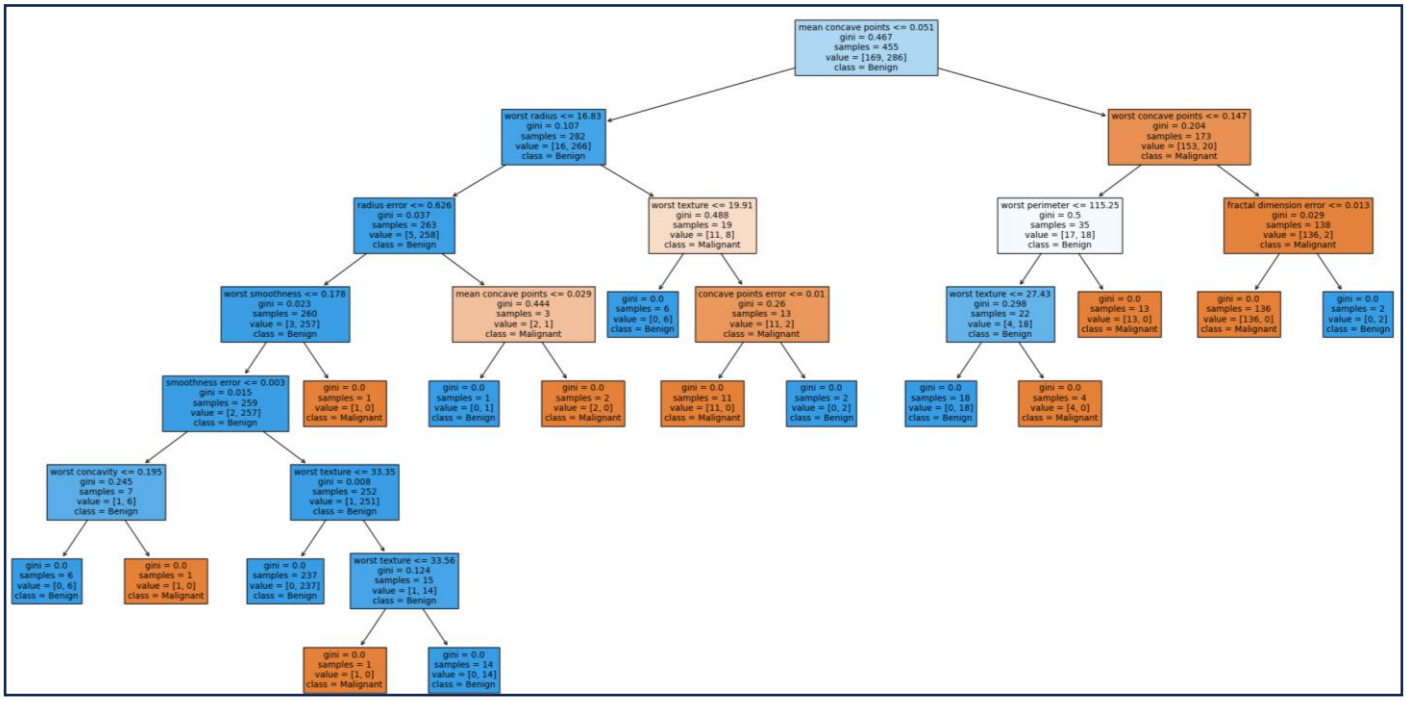
In [11]:
```
#prediction
preds = dt.predict(test)
```

In [12]:
```
print(preds)
```

```
[1 0 0 1 1 0 0 0 0 1 1 0 1 1 1 0 1 1 1 0 1 1 0 1 1 1 1 1 1 0 1 1 1 1 1 1 0
 1 0 1 1 0 1 1 1 1 0 1 1 1 0 0 1 1 1 1 1 0 0 1 1 0 0 1 1 1 0 0 1 1 0 0 1 0
 1 1 1 1 1 1 0 1 1 0 0 0 0 0 1 1 1 1 1 1 1 1 0 0 1 0 0 1 0 0 1 1 1 0 0 1 0
 1 1 0]
```

We can visualize the decision tree with the help of the following Python code:

In [13]:
```
#decision_tree
from sklearn import tree
from matplotlib import pyplot as plt
figure = plt.figure(figsize=(30,15))
plot_tree = tree.plot_tree(dt,feature_names = feature_names,class_names={0:'Malignant' , 1:'Benign'},
                   filled = True,fontsize=10)
plt.savefig('decision_tree.png')
```



Step 5 : find out the accuracy of our model.

We are going to use the accuracy_score() ,precision_score , recall_score , and f1_score functions to determine the accuracy.

```
In [14]: from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score
```

```
In [16]: #evaluation
         print("Accuracy:",  round (accuracy_score(test_labels, preds),4)*100,"%")
         print("Precision:",  round (precision_score(test_labels, preds),4)*100,"%")
         print("Recall:",  round (recall_score(test_labels, preds),4)*100,"%")
         print("F1 Score:", round (f1_score(test_labels, preds),4)*100,"%")
```

```
Accuracy: 93.86 %
Precision: 94.44 %
Recall: 95.77 %
F1 Score: 95.1 %
```

Results:

The Decision Tree Classifier was evaluated using the test data.

- Accuracy: The proportion of correct predictions (both benign and malignant).
- Precision: The accuracy of positive predictions.
- Recall: The ability of the model to find all the positive samples.
- F1 Score: A weighted harmonic mean of precision and recall.

The results obtained were as follows:

- Accuracy: 93.86%
- Precision: 94.44%
- Recall: 95.77%
- F1 Score: 95.1%

Potential Areas for Improvement:

1. Feature Selection: Implementing feature selection techniques could improve model simplicity and potentially increase the interpretability of the model.
2. 2. Hyperparameter Tuning: Adjusting parameters like max_depth and min_samples_split could help in combating overfitting and might improve the model's performance.
3. 3. Ensemble Methods: Using ensemble techniques such as Random Forests orGradient Boosting could provide better accuracy and robustness compared to a single Decision Tree.
4. 4. Cross-Validation: Utilizing cross-validation during training could provide a more accurate assessment of model performance across different subsets of the dataset.

Conclusion:

based on the Breast Cancer Wisconsin (Diagnostic) Database , The Decision Tree Classifier serves as a potent tool for the classification of breast cancer tumors. It offers high accuracy with reliable detection capabilities for malignant tumors. Future work will focus on refining the model through advanced techniques to enhance its predictive power and usability in clinical settings.