Breast Cancer Classification

Decision Tree

by

Muhammad Fauzan

About Me

Hi, I’m Muhammad Fauzan, an Informatics Engineering student at Universitas Brawijaya with a strong interest in Data Science and Machine Learning. I enjoy exploring data to solve real-world problems using tools like Python, SQL, and data visualization. I’m always looking for opportunities to grow by working on hands-on projects and learning new skills.

Through my academic journey, I’ve gained experience in data analysis, web development, and machine learning. I’m open to new challenges and collaborations—feel free to connect with me to discuss potential opportunities.

Tools used

About Dataset

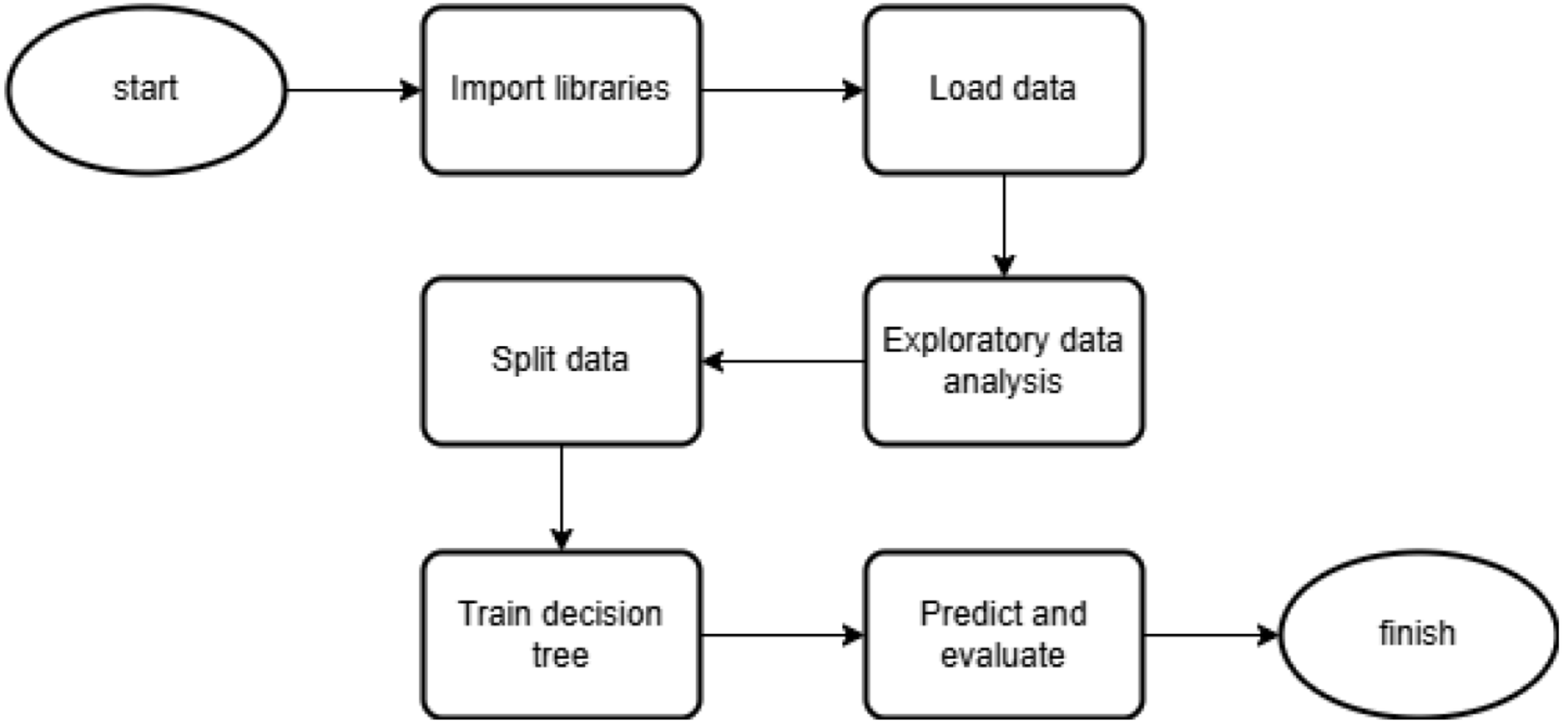
Breast Cancer

The dataset used in this project is the Breast Cancer Dataset from Scikit-learn, which consists of 569 data samples with 30 numerical features related to breast tumor characteristics. This dataset aims to classify between benign and malignant tumors. This dataset is widely used in testing machine learning models because of its relevance and good data structure.

Decision Tree

The model used is Decision Tree, an algorithm that works by recursively dividing data based on certain features until reaching a final decision. This algorithm is simple to understand, able to handle non-linear relationships, and allows visualization of the prediction process in the form of an intuitive tree structure.

Flowchart


start
Import libraries
Load data
Exploratory data analysis
Split data
Train decision tree
Predict and evaluate
finish

import libraries and load data

Breast Cancer

```
from sklearn.datasets import load_breast_cancer
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

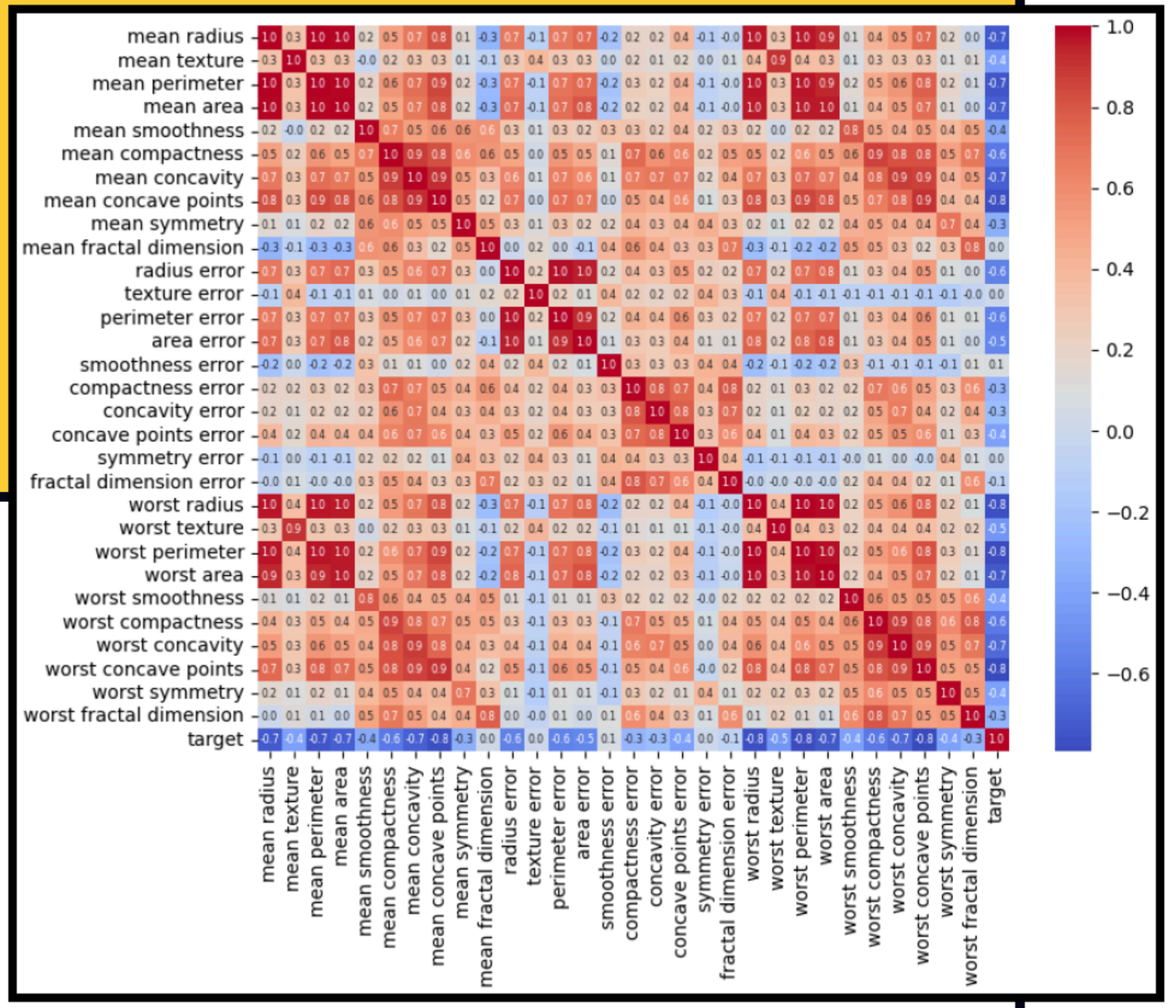
```
cancer = load_breast_cancer()

X = cancer.data
y = cancer.target

df_X = pd.DataFrame(X, columns=cancer.feature_names)
df_y = pd.Series(y, name='target')

df = pd.concat([df_X, df_y], axis=1)
```

Overview Data

The dataset contains 569 samples with 30 numerical features describing the biological characteristics of tumors (such as radius, texture, and smoothness). The targets are of two classes: malignant and benign.

```
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   mean radius             569 non-null    float64
 1   mean texture            569 non-null    float64
 2   mean perimeter          569 non-null    float64
 3   mean area               569 non-null    float64
 4   mean smoothness         569 non-null    float64
 5   mean compactness        569 non-null    float64
 6   mean concavity          569 non-null    float64
 7   mean concave points     569 non-null    float64
 8   mean symmetry           569 non-null    float64
 9   mean fractal dimension  569 non-null    float64
 10  radius error            569 non-null    float64
 11  texture error           569 non-null    float64
 12  perimeter error         569 non-null    float64
 13  area error              569 non-null    float64
 14  smoothness error        569 non-null    float64
 15  compactness error       569 non-null    float64
 16  concavity error         569 non-null    float64
 17  concave points error    569 non-null    float64
 18  symmetry error          569 non-null    float64
 19  fractal dimension error 569 non-null    float64
 20  worst radius            569 non-null    float64
 21  worst texture           569 non-null    float64
 22  worst perimeter         569 non-null    float64
 23  worst area              569 non-null    float64
 24  worst smoothness        569 non-null    float64
 25  worst compactness       569 non-null    float64
 26  worst concavity         569 non-null    float64
 27  worst concave points    569 non-null    float64
 28  worst symmetry          569 non-null    float64
 29  worst fractal dimension 569 non-null    float64
```

	mean radius	mean texture	mean perimeter	mean area	mean smoothness	mean compactness	mean concavity	mean concave points	mean symmetry	mean fractal dimension	...	worst texture	worst perimeter	worst area	worst smoothness	worst compactness	worst concavity	worst concave points	worst symmetry	worst fractal dimension
0	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.30010	0.14710	0.2419	0.07871	...	17.33	184.60	2019.0	0.1622	0.6656	0.7119	0.2654	0.4601	0.11890
1	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.08690	0.07017	0.1812	0.05667	...	23.41	158.80	1956.0	0.1238	0.1866	0.2416	0.1860	0.2750	0.08902
2	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.19740	0.12790	0.2069	0.05999	...	25.53	152.50	1709.0	0.1444	0.4245	0.4504	0.2430	0.3613	0.08758
3	11.42	20.38	77.58	386.1	0.14250	0.28390	0.24140	0.10520	0.2597	0.09744	...	26.50	98.87	567.7	0.2098	0.8663	0.6869	0.2575	0.6638	0.17300
4	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.19800	0.10430	0.1809	0.05883	...	16.67	152.20	1575.0	0.1374	0.2050	0.4000	0.1625	0.2364	0.07678
5	12.45	15.70	82.57	477.1	0.12780	0.17000	0.15780	0.08089	0.2087	0.07613	...	23.75	103.40	741.6	0.1791	0.5249	0.5355	0.1741	0.3985	0.12440
6	18.25	19.98	119.60	1040.0	0.09463	0.10900	0.11270	0.07400	0.1794	0.05742	...	27.66	153.20	1606.0	0.1442	0.2576	0.3784	0.1932	0.3063	0.08368
7	13.71	20.83	90.20	577.9	0.11890	0.16450	0.09366	0.05985	0.2196	0.07451	...	28.14	110.60	897.0	0.1654	0.3682	0.2678	0.1556	0.3196	0.11510
8	13.00	21.82	87.50	519.8	0.12730	0.19320	0.18590	0.09353	0.2350	0.07389	...	30.73	106.20	739.3	0.1703	0.5401	0.5390	0.2060	0.4378	0.10720
9	12.46	24.04	83.97	475.9	0.11860	0.23960	0.22730	0.08543	0.2030	0.08243	...	40.68	97.65	711.4	0.1853	1.0580	1.1050	0.2210	0.4366	0.20750



Correlation

The correlation matrix shows the relationship between the existing features, the higher the value, the tighter the correlation between the features. By visualizing the correlation matrix, it will be easier to determine what features will be used to build a machine learning model.

Split data and Train Model

Split data

```
X_train, X_test, y_train, y_test = train_test_split(df_X, df_y, test_size=0.2, random_state=42)
```

This code splits the dataset into two parts: training data and test data.

- X_train and y_train: Used to train the model.
- X_test and y_test: Used to test the model performance. The test_size=0.2 parameter indicates that 20% of the data is used for testing, while 80% is used for training. The random_state=42 parameter ensures that the data split results are consistent every time the code is run.

Train model

```
model = DecisionTreeClassifier(random_state=42)
print(model.fit(X_train, y_train))
```

```
DecisionTreeClassifier(random_state=42)
```

The code above creates a classification model using DecisionTreeClassifier, the model is fit on the training data X_train and target y_train, which means the model is trained to learn patterns from the data.

Predict and Evaluate

1. **Predict**: model.predict(X_test) is used to predict the outcome of the test data X_test using a previously trained model.

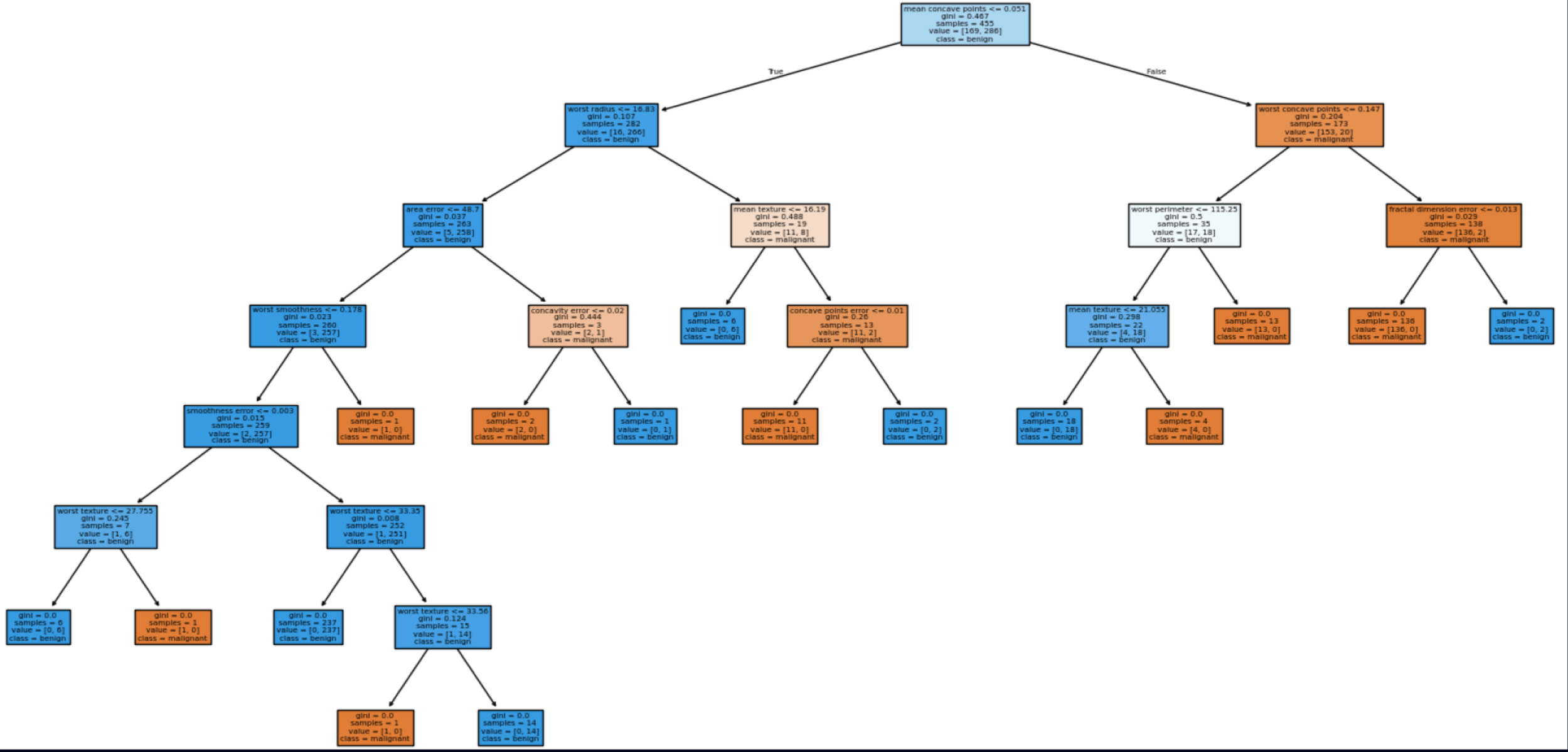
2. **Accuracy**: accuracy_score(y_test, y_pred) calculates accuracy by comparing the predicted result of y_pred with the original value of y_test. The accuracy produced by the Decision Tree model is 94.74%, which means the model can predict the results very well.

3. **Output**: print() displays a classification report that includes the model accuracy in percentage format with two decimal places.

```
y_pred = model.predict(X_test)

accuracy = accuracy_score(y_test, y_pred)

print("Laporan Klasifikasi:")
print(f"Akurasi: {accuracy * 100:.2f}%")
```

```
Laporan Klasifikasi:
Akurasi: 94.74%
```

Decision Tree Result



Thank you!

ig: mfzznn_ | github: https://github.com/Muzann11