SIMPLE CLASSIFICATION OF BREAST CANCER DIAGNOSTIK USING A DECISION TREE

PAJAR

INTRODUCTION & OBJECTIVE

Wisconsin Breast Cancer Diagnostic is one of the toy datasets from scikit-learn. It’s a classic and very easy binary classification dataset. It has 569 instances, and attributes; including: 30 numeric, predictive attributes and the class.

The objective is to build a machine learning model for predicting whether a breast tumor is benign (0) or malignant (1) using Random Forest Algorithm..

table of CONTENT

INPUT DATA

EXPLORASI DATA

DATA MODELING

DATA VISUALIZATION

TOOLS

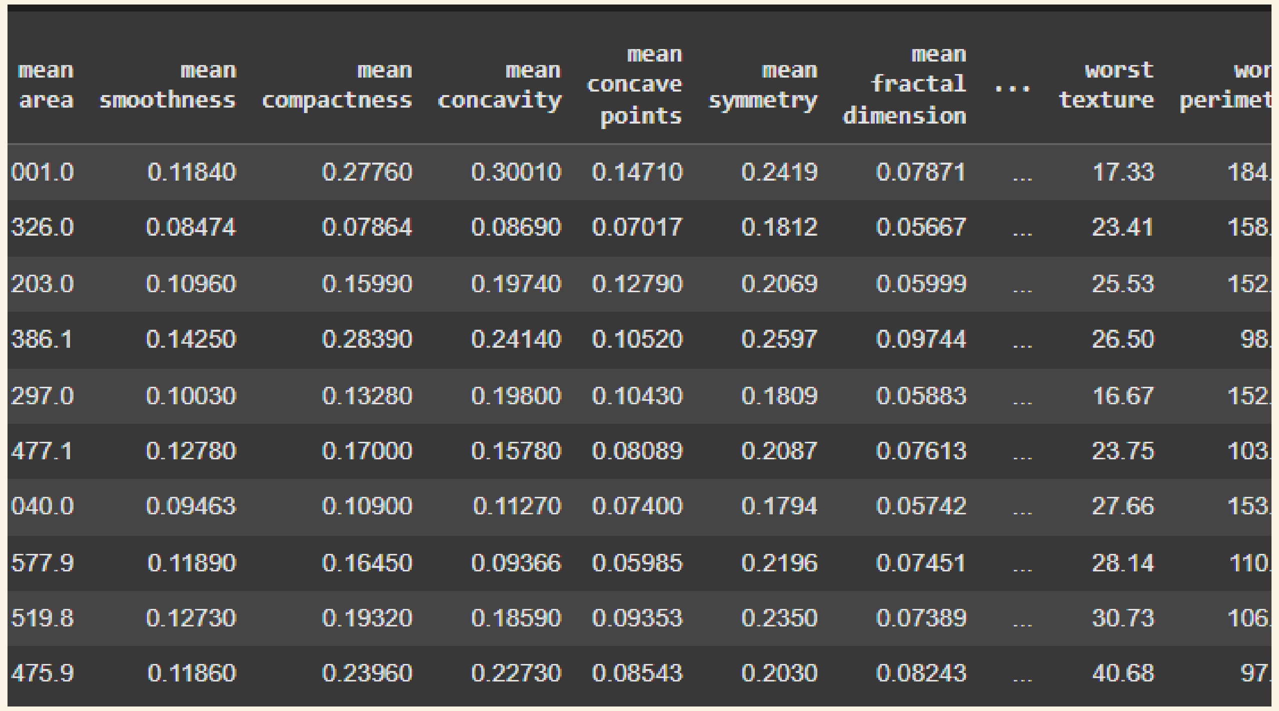
pandas

seaborn

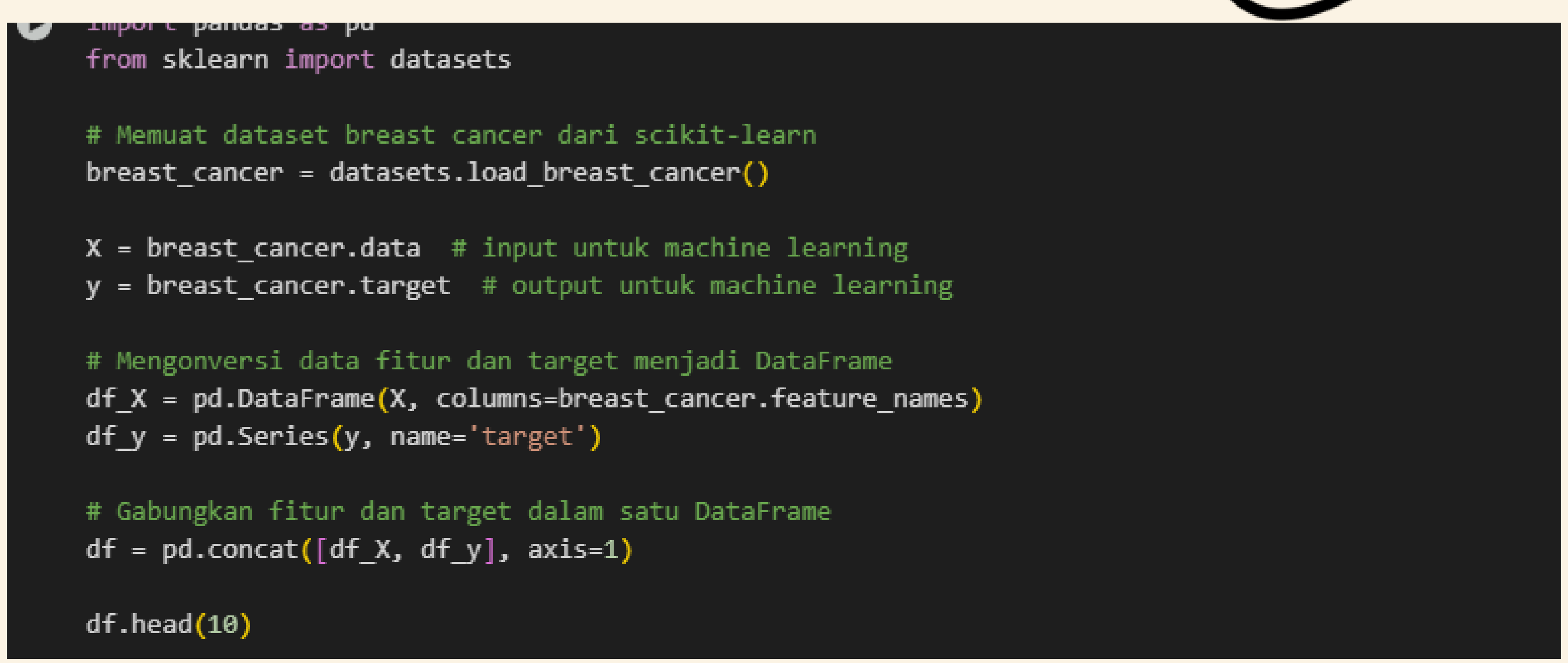
scikit learn

matplotlib

INPUT DATA

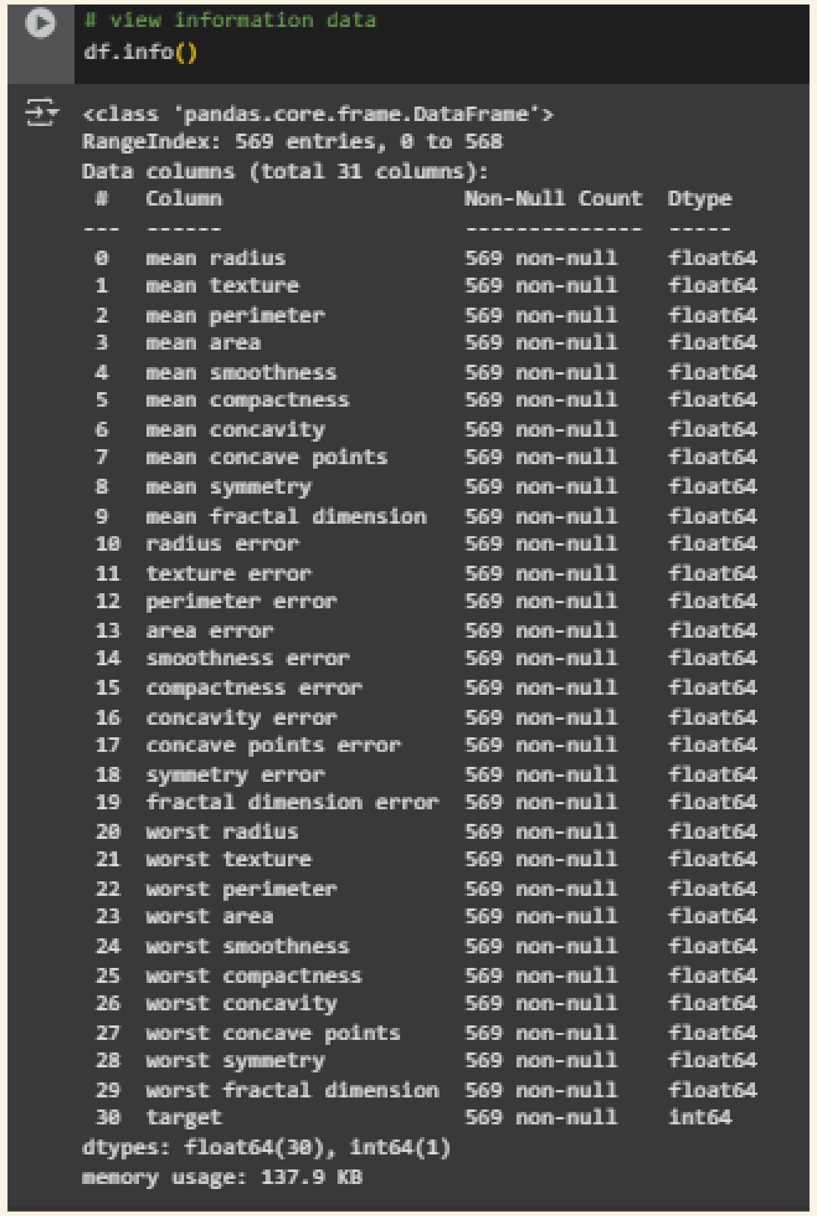


mean area	mean smoothness	mean compactness	mean concavity	mean concave points	mean symmetry	mean fractal dimension	...	worst texture	wor perimet
001.0	0.11840	0.27760	0.30010	0.14710	0.2419	0.07871	...	17.33	184.
326.0	0.08474	0.07864	0.08690	0.07017	0.1812	0.05667	...	23.41	158.
203.0	0.10960	0.15990	0.19740	0.12790	0.2069	0.05999	...	25.53	152.
386.1	0.14250	0.28390	0.24140	0.10520	0.2597	0.09744	...	26.50	98.
297.0	0.10030	0.13280	0.19800	0.10430	0.1809	0.05883	...	16.67	152.
477.1	0.12780	0.17000	0.15780	0.08089	0.2087	0.07613	...	23.75	103.
040.0	0.09463	0.10900	0.11270	0.07400	0.1794	0.05742	...	27.66	153.
577.9	0.11890	0.16450	0.09366	0.05985	0.2196	0.07451	...	28.14	110.
519.8	0.12730	0.19320	0.18590	0.09353	0.2350	0.07389	...	30.73	106.
475.9	0.11860	0.23960	0.22730	0.08543	0.2030	0.08243	...	40.68	97.



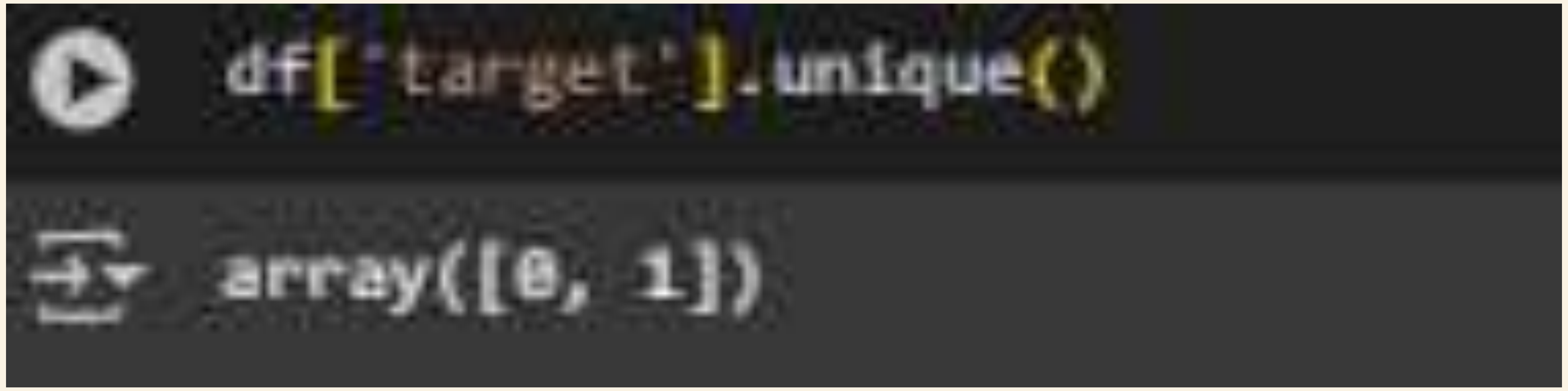
```
import pandas as pd
from sklearn import datasets

# Memuat dataset breast cancer dari scikit-learn
breast_cancer = datasets.load_breast_cancer()

X = breast_cancer.data  # input untuk machine learning
y = breast_cancer.target  # output untuk machine learning

# Mengonversi data fitur dan target menjadi DataFrame
df_X = pd.DataFrame(X, columns=breast_cancer.feature_names)
df_y = pd.Series(y, name='target')

# Gabungkan fitur dan target dalam satu DataFrame
df = pd.concat([df_X, df_y], axis=1)

df.head(10)
```

EXPLORASI DATA ANALIS



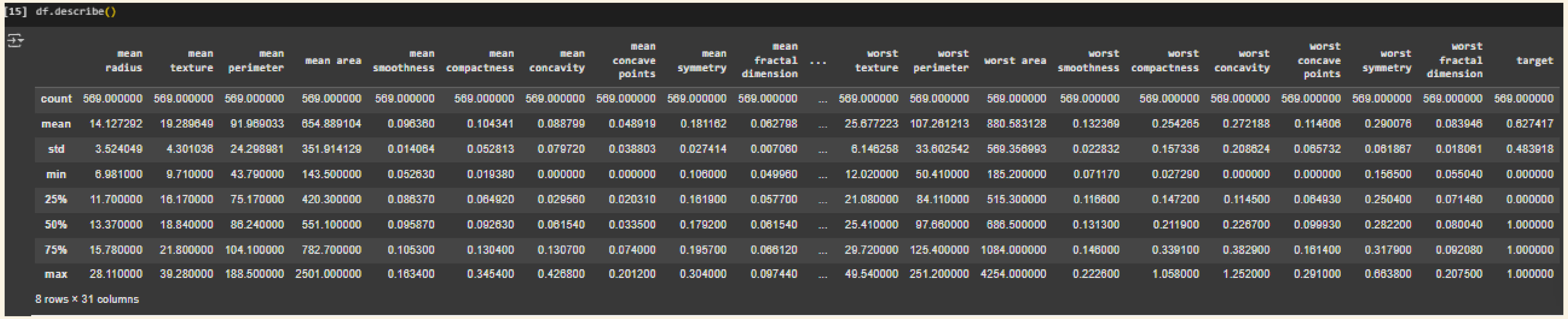
```
# view information data
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 569 entries, 0 to 568
Data columns (total 31 columns):
 #   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
 30  target                   569 non-null    int64  
dtypes: float64(30), int64(1)
memory usage: 137.9 KB
```



```
df['target'].unique()
```

```
array([0, 1])
```

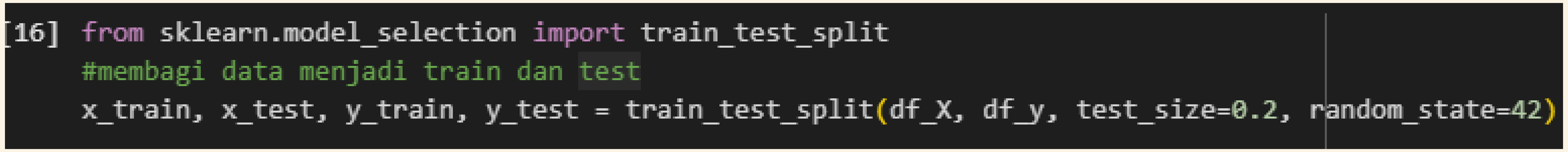


```
[15] df.describe()
```

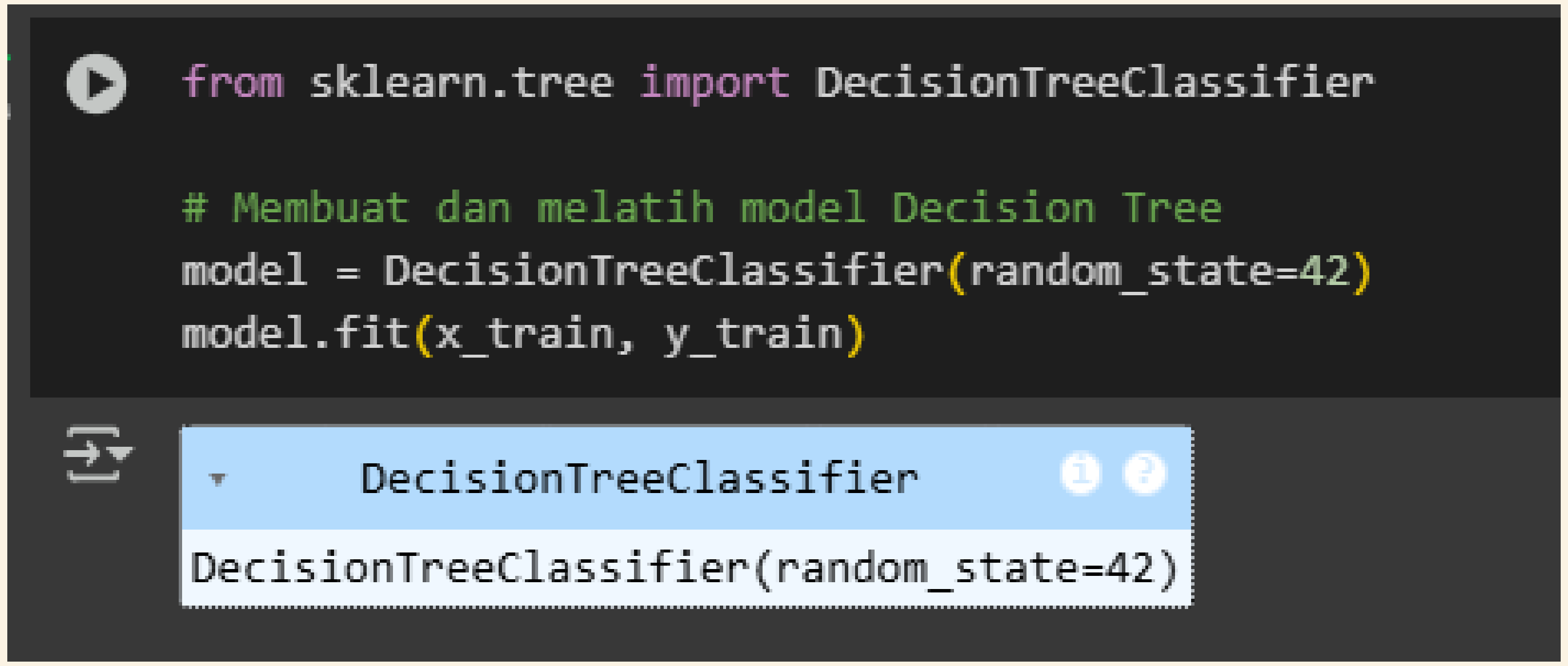
	mean radius	mean texture	mean perimeter	mean area	mean smoothness	mean compactness	mean concavity	mean concave points	mean symmetry	mean fractal dimension	...	worst texture	worst perimeter	worst area	worst smoothness	worst compactness	worst concavity	worst concave points	worst symmetry	worst fractal dimension	target
count	569.000000	569.000000	569.000000	569.000000	569.000000	569.000000	569.000000	569.000000	569.000000	569.000000	...	569.000000	569.000000	569.000000	569.000000	569.000000	569.000000	569.000000	569.000000	569.000000	569.000000
mean	14.127292	19.289649	91.969033	654.889104	0.096360	0.104341	0.088799	0.048919	0.181162	0.062798	...	25.677223	107.261213	880.583128	0.132369	0.254265	0.272188	0.114606	0.290076	0.083946	0.627417
std	3.524049	4.301036	24.298981	351.914129	0.014064	0.052813	0.079720	0.038803	0.027414	0.007060	...	6.146258	33.602542	569.356993	0.022832	0.157336	0.208624	0.065732	0.061867	0.018061	0.483918
min	6.981000	9.710000	43.790000	143.500000	0.052630	0.019380	0.000000	0.000000	0.106000	0.049960	...	12.020000	50.410000	185.200000	0.071170	0.027290	0.000000	0.000000	0.156500	0.055040	0.000000
25%	11.700000	16.170000	75.170000	420.300000	0.086370	0.064920	0.029560	0.020310	0.161900	0.057700	...	21.080000	84.110000	515.300000	0.116600	0.147200	0.114500	0.064930	0.250400	0.071460	0.000000
50%	13.370000	18.840000	86.240000	551.100000	0.095870	0.092630	0.061540	0.033500	0.179200	0.061540	...	25.410000	97.660000	686.500000	0.131300	0.211900	0.226700	0.099930	0.282200	0.080040	1.000000
75%	15.780000	21.800000	104.100000	782.700000	0.105300	0.130400	0.130700	0.074000	0.195700	0.066120	...	29.720000	125.400000	1084.000000	0.146000	0.339100	0.382900	0.161400	0.317900	0.092080	1.000000
max	28.110000	39.280000	188.500000	2501.000000	0.163400	0.345400	0.426800	0.201200	0.304000	0.097440	...	49.540000	251.200000	4254.000000	0.222600	1.058000	1.252000	0.291000	0.663800	0.207500	1.000000

8 rows × 31 columns

DATA MODELING

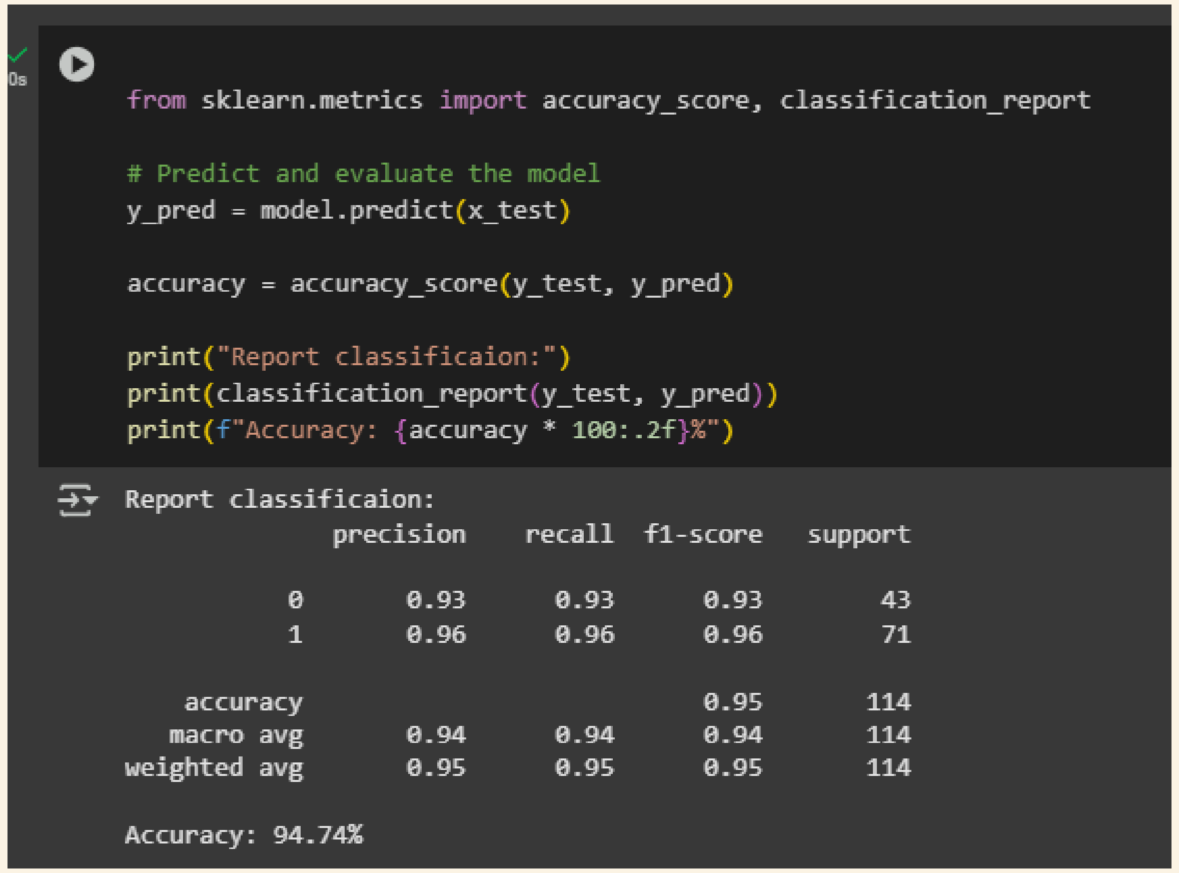


```
[16] from sklearn.model_selection import train_test_split
     #membagi data menjadi train dan test
     x_train, x_test, y_train, y_test = train_test_split(df_X, df_y, test_size=0.2, random_state=42)
```



```
from sklearn.tree import DecisionTreeClassifier

# Membuat dan melatih model Decision Tree
model = DecisionTreeClassifier(random_state=42)
model.fit(x_train, y_train)
```

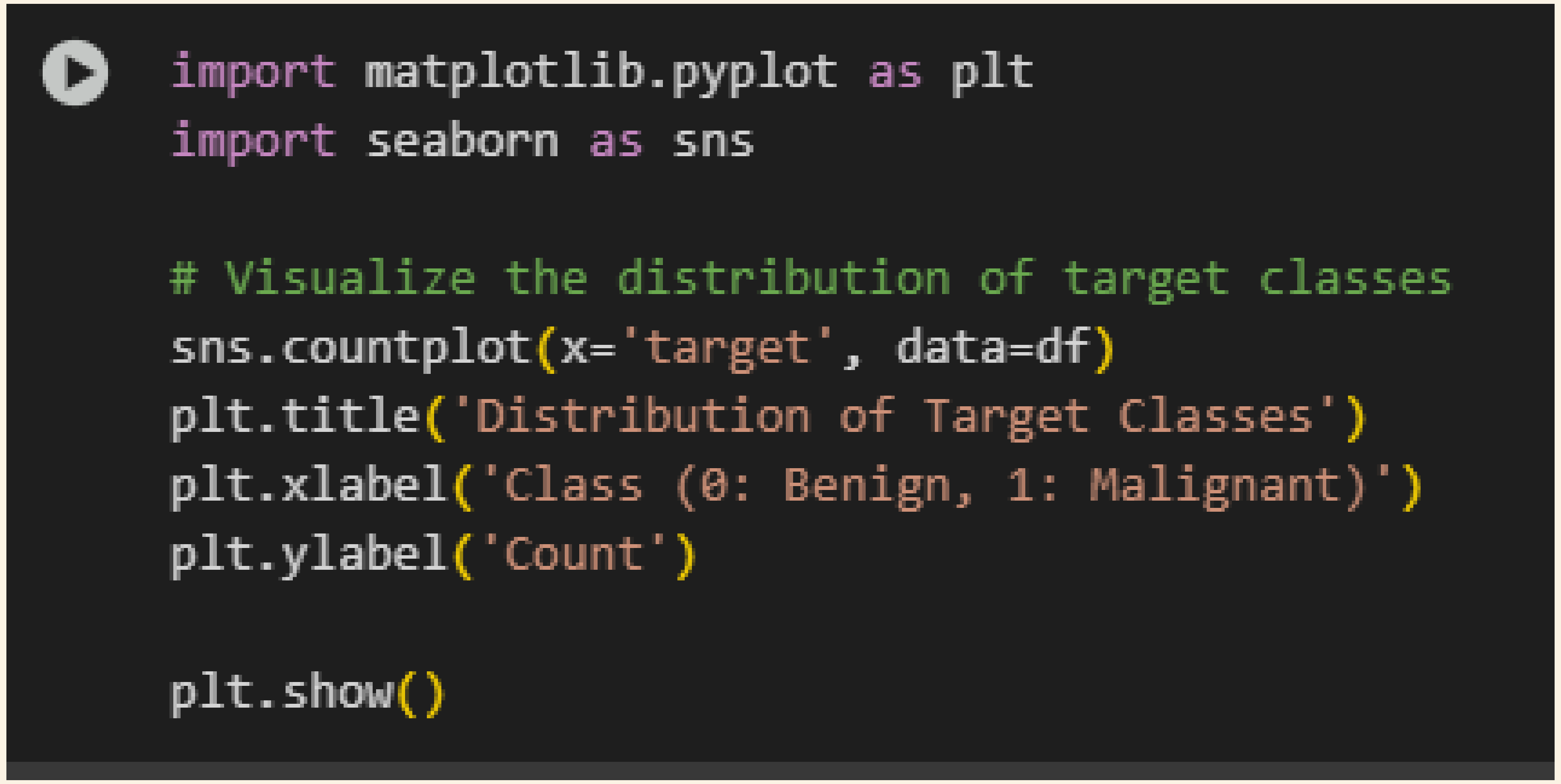
```
DecisionTreeClassifier
DecisionTreeClassifier(random_state=42)
```



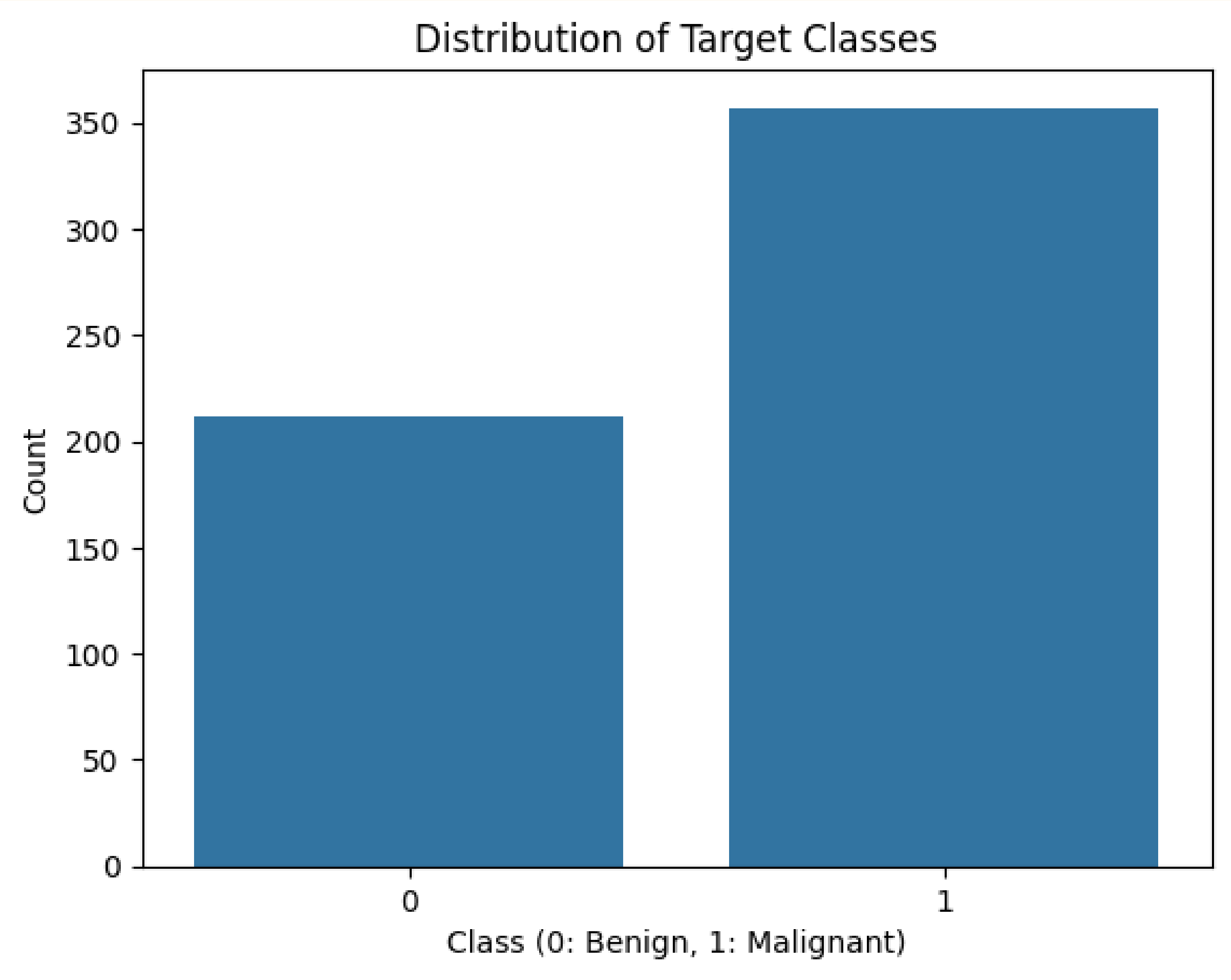
```
from sklearn.metrics import accuracy_score, classification_report

# Predict and evaluate the model
y_pred = model.predict(x_test)

accuracy = accuracy_score(y_test, y_pred)

print("Report classificaion:")
print(classification_report(y_test, y_pred))
print(f"Accuracy: {accuracy * 100:.2f}%")
```

```
Report classificaion:
              precision    recall  f1-score   support

           0       0.93      0.93      0.93        43
           1       0.96      0.96      0.96        71

    accuracy                           0.95       114
   macro avg       0.94      0.94      0.94       114
weighted avg       0.95      0.95      0.95       114

Accuracy: 94.74%
```

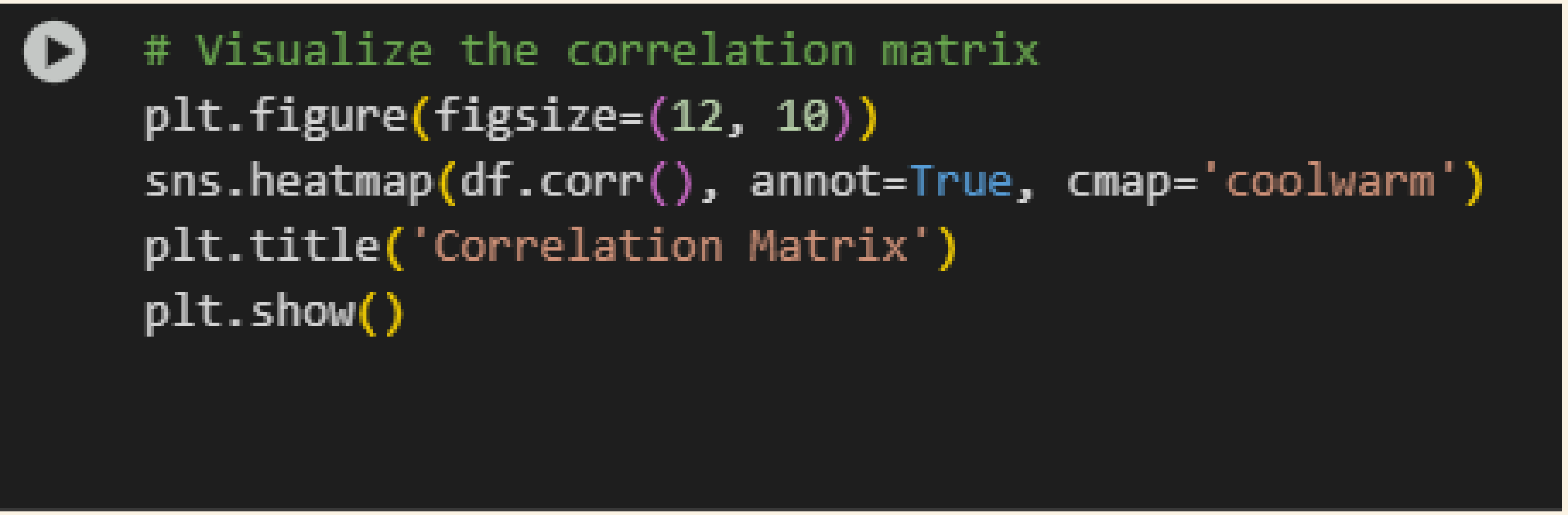
DATA VISUALIZATION



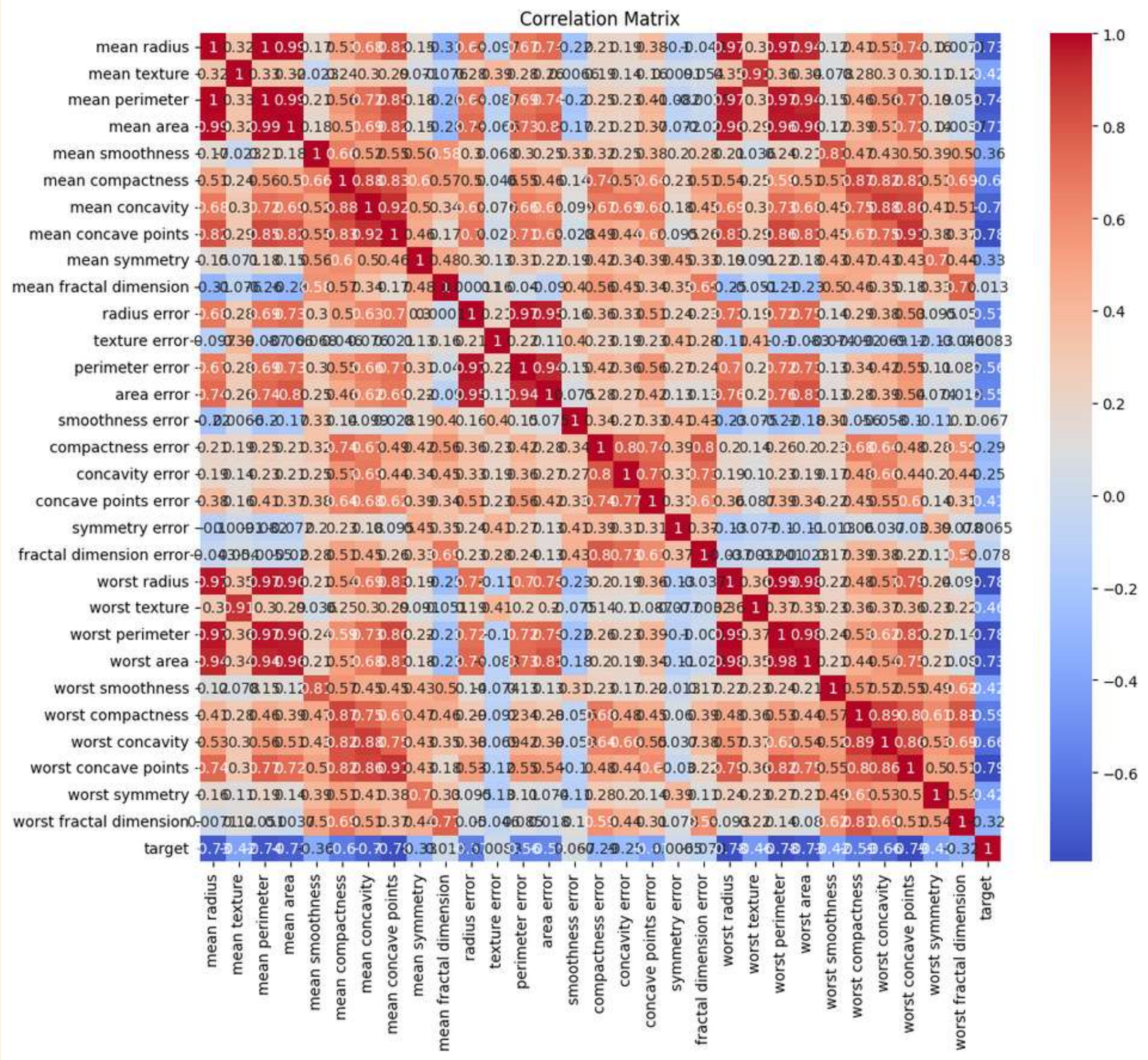
```
import matplotlib.pyplot as plt
import seaborn as sns

# Visualize the distribution of target classes
sns.countplot(x='target', data=df)
plt.title('Distribution of Target Classes')
plt.xlabel('Class (0: Benign, 1: Malignant)')
plt.ylabel('Count')

plt.show()
```



DATA VISUALIZATION

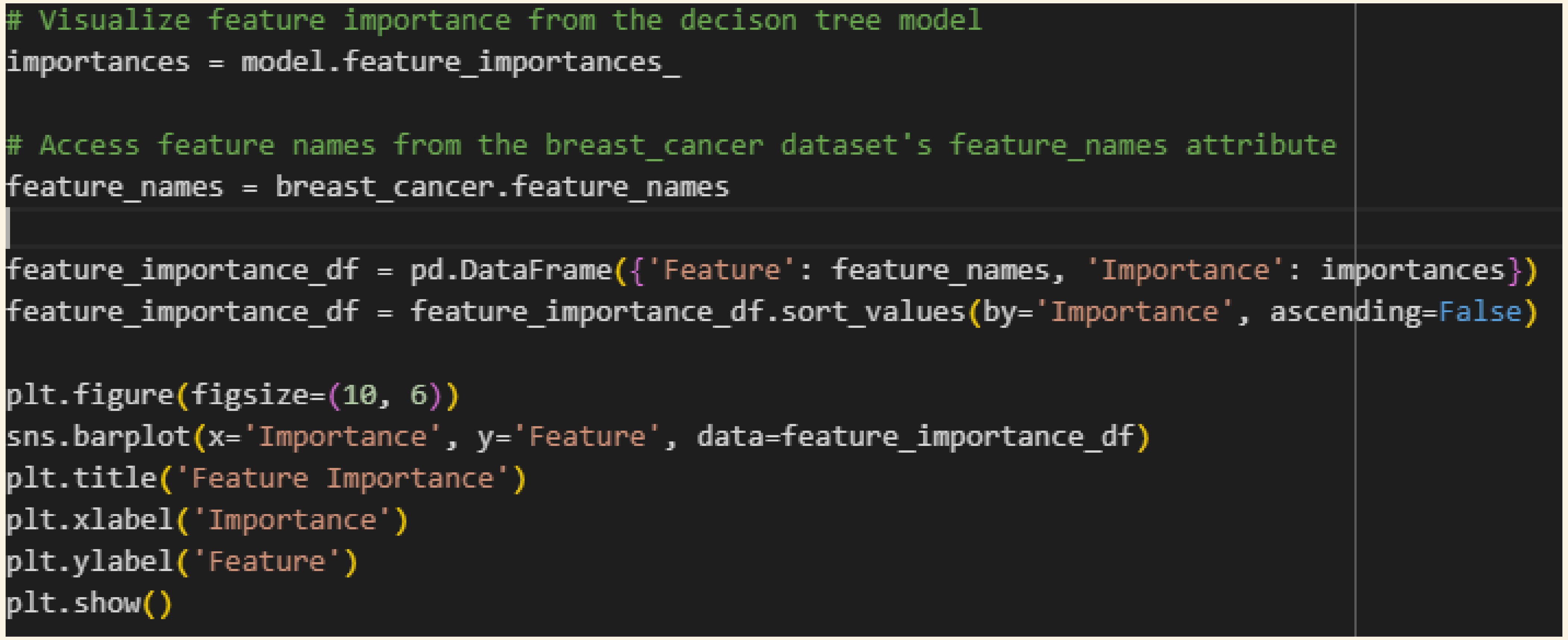


```
# Visualize the correlation matrix
plt.figure(figsize=(12, 10))
sns.heatmap(df.corr(), annot=True, cmap='coolwarm')
plt.title('Correlation Matrix')
plt.show()
```

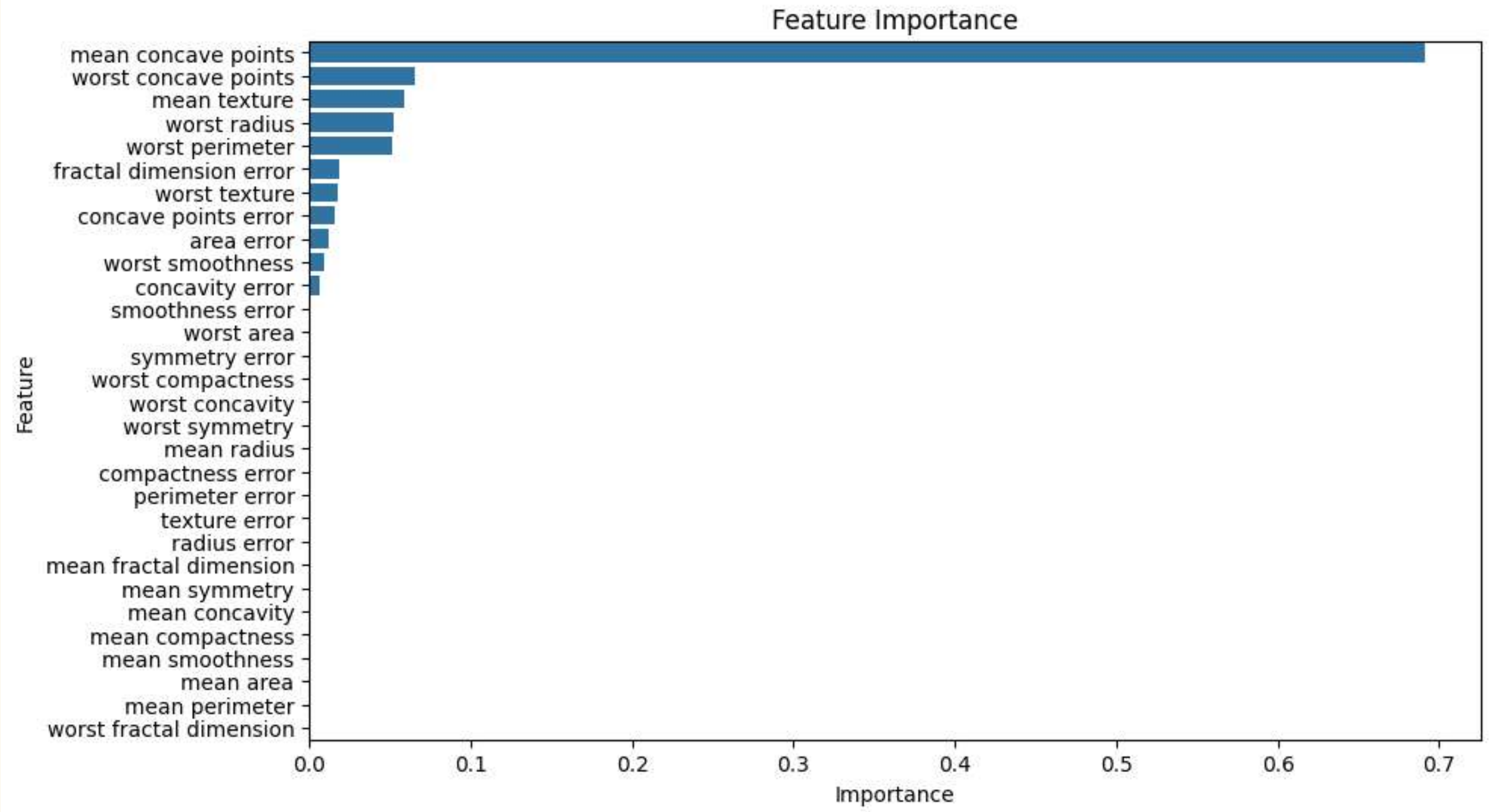


→

DATA VISUALIZATION



```
# Visualize feature importance from the decison tree model
importances = model.feature_importances_

# Access feature names from the breast_cancer dataset's feature_names attribute
feature_names = breast_cancer.feature_names

feature_importance_df = pd.DataFrame({'Feature': feature_names, 'Importance': importances})
feature_importance_df = feature_importance_df.sort_values(by='Importance', ascending=False)

plt.figure(figsize=(10, 6))
sns.barplot(x='Importance', y='Feature', data=feature_importance_df)
plt.title('Feature Importance')
plt.xlabel('Importance')
plt.ylabel('Feature')
plt.show()
```



THANKS

085715964578

www.linkedin.com/in/pajar-saputra-88563b2a6

https://github.com/pajarbeuy