```
import pandas as pd
from sklearn.datasets import load_breast_cancer
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.tree import DecisionTreeClassifier, plot_tree
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
import seaborn as sns
import matplotlib.pyplot as plt


# Memuat dataset kanker payudara
data = load_breast_cancer()


# Mengonversi dataset menjadi DataFrame pandas
df = pd.DataFrame(data.data, columns=data.feature_names)
df['target'] = data.target


# Menampilkan lima baris pertama dataset
print("Lima baris pertama dataset:\n")
display(df.head())
```

```
Lima baris pertama dataset:
```

	mean radius	mean texture	mean perimeter	mean area	mean smoothness	mean compactness	mean concavity	mean concave points	mean symmetry	mean fractal dimension
0	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.3001	0.14710	0.2419	0.07871
1	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.0869	0.07017	0.1812	0.05667
2	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.1974	0.12790	0.2069	0.05999
3	11.42	20.38	77.58	386.1	0.14250	0.28390	0.2414	0.10520	0.2597	0.09744
4	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.1980	0.10430	0.1809	0.05883

5 rows × 31 columns

```
# Menampilkan informasi dataset
print("Informasi Dataset:")
df.info()
```

```
Informasi Dataset:
<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
```

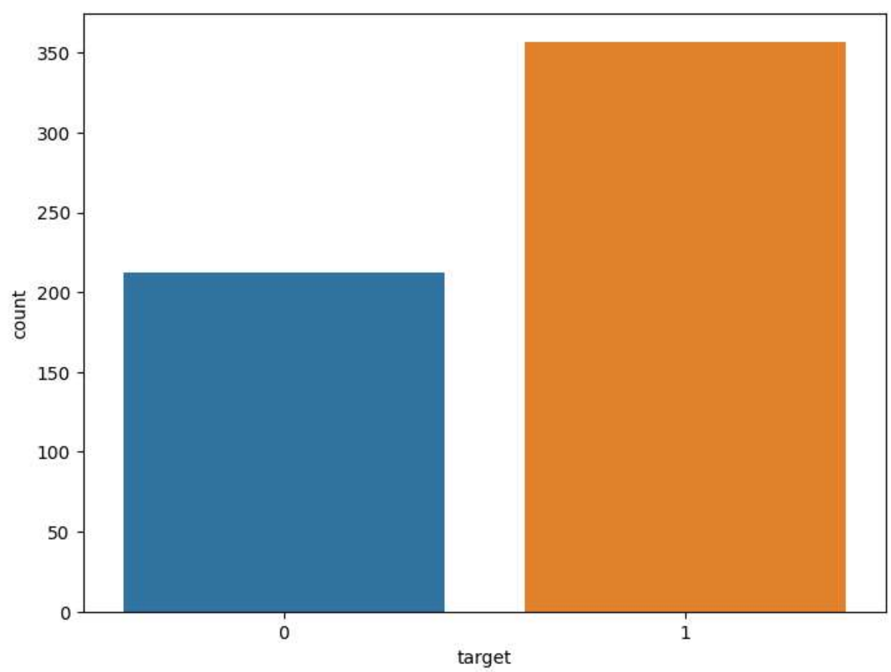
```
# Menampilkan ringkasan statistik dataset
print("\nStatistik Dataset:")
display(df.describe())
```

Statistik Dataset:

	mean radius	mean texture	mean perimeter	mean area	mean smoothness	mean compactness	mear concavity
count	569.000000	569.000000	569.000000	569.000000	569.000000	569.000000	569.00000C
mean	14.127292	19.289649	91.969033	654.889104	0.096360	0.104341	0.08879S
std	3.524049	4.301036	24.298981	351.914129	0.014064	0.052813	0.07972C
min	6.981000	9.710000	43.790000	143.500000	0.052630	0.019380	0.00000C
25%	11.700000	16.170000	75.170000	420.300000	0.086370	0.064920	0.02956C
50%	13.370000	18.840000	86.240000	551.100000	0.095870	0.092630	0.06154C
75%	15.780000	21.800000	104.100000	782.700000	0.105300	0.130400	0.13070C
max	28.110000	39.280000	188.500000	2501.000000	0.163400	0.345400	0.42680C

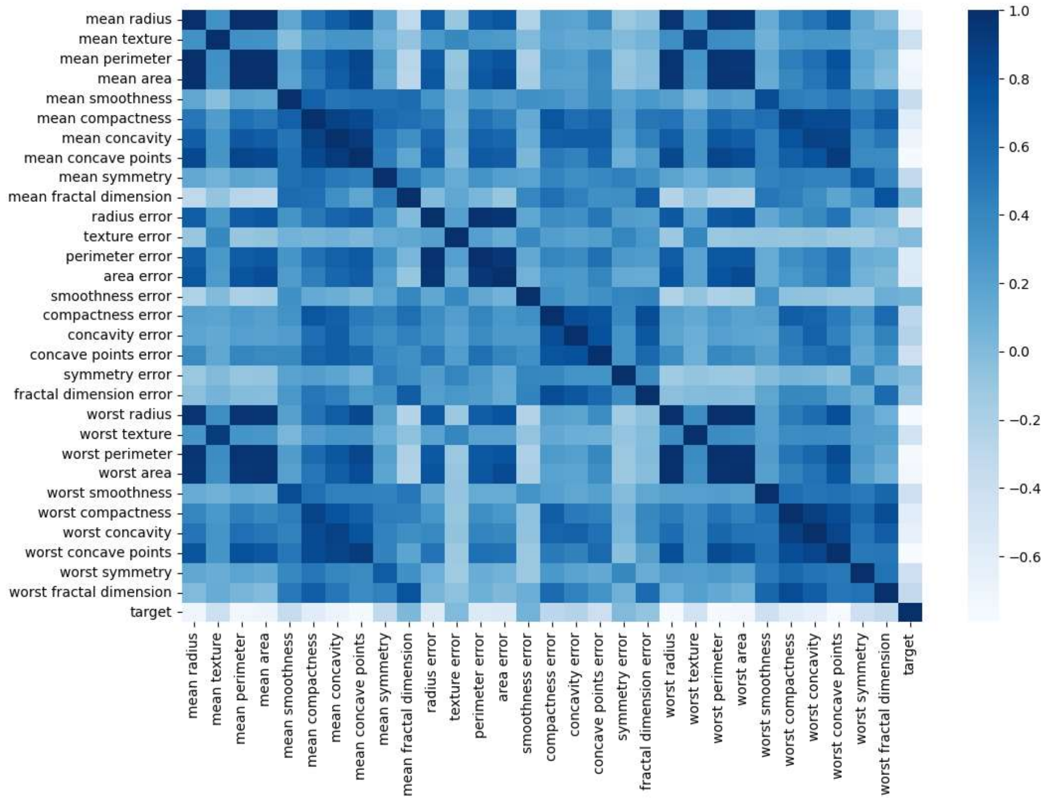
8 rows × 31 columns

```
# Menampilkan distribusi variabel target
plt.figure(figsize=(8, 6))
sns.countplot(x='target', data=df)
plt.show()
```




```
# Menampilkan matriks korelasi dengan menggunakan fungsi pairplot dari Seaborn
sns.pairplot(df, hue='target', diag_kind='hist')
plt.show()
```

```
# Menampilkan matriks korelasi menggunakan fungsi heatmap dari Seaborn
plt.figure(figsize=(12, 8))
sns.heatmap(df.corr(), cmap='Blues')
plt.show()
```




```
# Membagi dataset menjadi set pelatihan dan pengujian
X = df.drop('target', axis=1)
y = df['target']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)
```

```
# Melakukan Normalisasi data menggunakan StandardScaler
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

```
# Melatih model self training
model1 = LogisticRegression()
model1.fit(X_train_scaled, y_train)
```

```
▾ LogisticRegression
LogisticRegression()
```

```
# Melatih model random forest
model2 = RandomForestClassifier(n_estimators=100, random_state=42)
model2.fit(X_train_scaled, y_train)
```

```
▾         RandomForestClassifier
RandomForestClassifier(random_state=42)
```

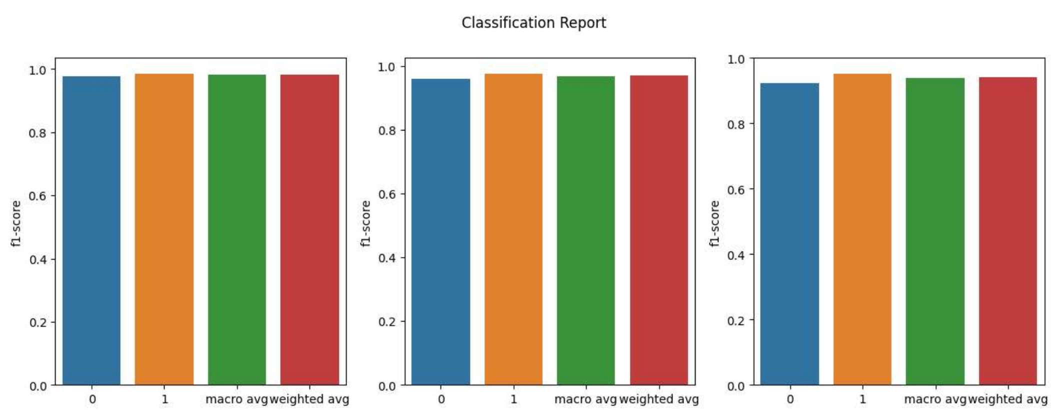
```
# Melatih model decission tree
model3 = DecisionTreeClassifier(random_state=42)
model3.fit(X_train, y_train)
```

```
▾ DecisionTreeClassifier
```

```
# Mengevaluasi model pada set pengujian
y_pred1 = model1.predict(X_test_scaled)
y_pred2 = model2.predict(X_test_scaled)
y_pred3 = model3.predict(X_test)

accuracy1 = accuracy_score(y_test, y_pred1)
accuracy2 = accuracy_score(y_test, y_pred2)
accuracy3 = accuracy_score(y_test, y_pred3)

print(f"Akurasi model self training: {accuracy1}")
print(f"Akurasi model random forest: {accuracy2}")
print(f"Akurasi model decission tree: {accuracy3}")
```

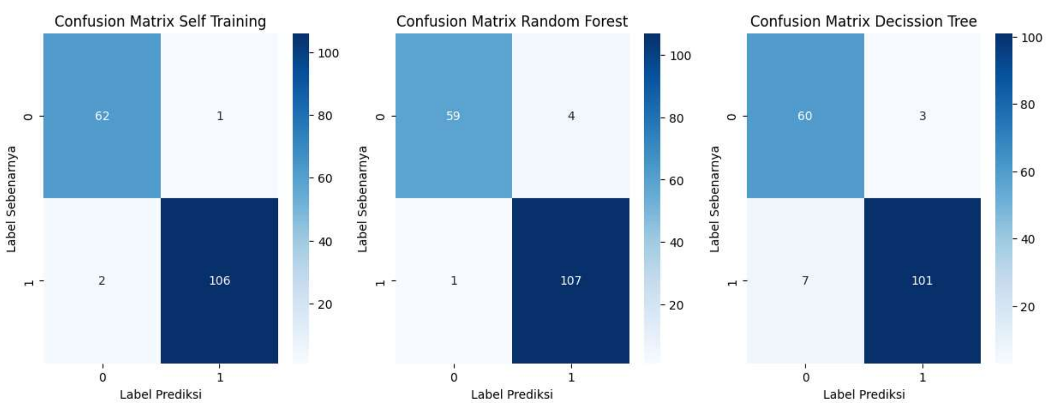
```
Akurasi model self training: 0.9824561403508771
Akurasi model random forest: 0.9707602339181286
Akurasi model decission tree: 0.9415204678362573
```

```
# Menampilkan classification report
report1 = classification_report(y_test, y_pred1, output_dict=True)
report2 = classification_report(y_test, y_pred2, output_dict=True)
report3 = classification_report(y_test, y_pred3, output_dict=True)

df_report1 = pd.DataFrame(report1).transpose()
df_report1.drop('support', axis=1, inplace=True)
df_report1.drop('accuracy', axis=0, inplace=True)

df_report2 = pd.DataFrame(report2).transpose()
df_report2.drop('support', axis=1, inplace=True)
df_report2.drop('accuracy', axis=0, inplace=True)

df_report3 = pd.DataFrame(report3).transpose()
df_report3.drop('support', axis=1, inplace=True)
df_report3.drop('accuracy', axis=0, inplace=True)

fig, axs = plt.subplots(1, 3, figsize=(15,5))
fig.suptitle("Classification Report")

sns.barplot(x=df_report1.index, y=df_report1['f1-score'], ax=axs[0])
sns.barplot(x=df_report2.index, y=df_report2['f1-score'], ax=axs[1])
sns.barplot(x=df_report3.index, y=df_report3['f1-score'], ax=axs[2])

plt.show()
```




```
# Menampilkan confusion matrix
cm1 = confusion_matrix(y_test, y_pred1)
cm2 = confusion_matrix(y_test, y_pred2)
cm3 = confusion_matrix(y_test, y_pred3)
```

```
plt.figure(figsize=(15,5))
plt.subplot(1, 3, 1)
sns.heatmap(cm1, annot=True, cmap='Blues', fmt='.0f')
plt.title("Confusion Matrix Self Training")
plt.xlabel("Label Prediksi")
plt.ylabel("Label Sebenarnya")

plt.subplot(1, 3, 2)
sns.heatmap(cm2, annot=True, cmap='Blues', fmt='.0f')
plt.title("Confusion Matrix Random Forest")
plt.xlabel("Label Prediksi")
plt.ylabel("Label Sebenarnya")

plt.subplot(1, 3, 3)
sns.heatmap(cm3, annot=True, cmap='Blues', fmt='.0f')
plt.title("Confusion Matrix Decission Tree")
plt.xlabel("Label Prediksi")
plt.ylabel("Label Sebenarnya")

plt.show()
```




```
# Visualisasi tingkat kepentingan fitur
feature_importance = pd.Series(model2.feature_importances_, index=X.columns)
feature_importance.nlargest(10).plot(kind='barh')
plt.title("10 Fitur Terpenting Random Forest")
plt.show()
```

```
# Visualisasi pohon keputusan
plt.figure(figsize=(20,10))
plot_tree(model3, feature_names=X.columns, class_names=['ganas', 'jinak'], filled=True)
plt.show()
```