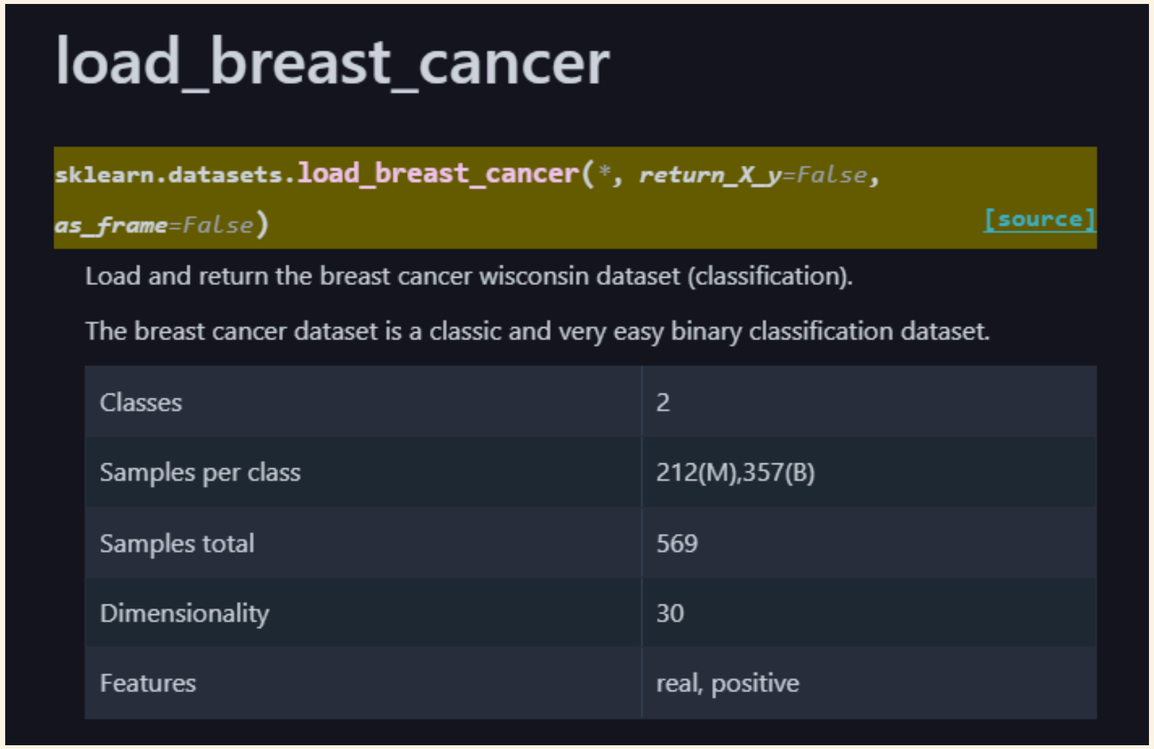
Classification

BREAST CANCER DIAGNOSIS USING DECISION TREE & LOGISTIC REGRESSION

BY RANIAH MARITZA

DATASET DESCRIPTION

Dataset **Breast Cancer Wisconsin (Diagnostic)** merupakan dataset yang diambil dari **Scikit-Learn** untuk membedakan antara tumor payudara ganas (malignant) dan jinak (benign).



load_breast_cancer

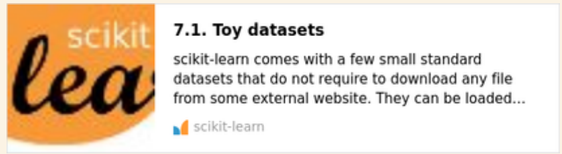
```
sklearn.datasets.load_breast_cancer(*, return_X_y=False,
as_frame=False)                                          [source]
```

Load and return the breast cancer wisconsin dataset (classification).

The breast cancer dataset is a classic and very easy binary classification dataset.

Classes	2
Samples per class	212(M),357(B)
Samples total	569
Dimensionality	30
Features	real, positive

SOURCE:



7.1. Toy datasets

scikit-learn comes with a few small standard datasets that do not require to download any file from some external website. They can be loaded...

scikit-learn

Dataset ini berisi 569 sampel dengan 30 fitur numerik.

- **Target (Y):**
  - 0 -> Malignant (Ganas)
  - 1 -> Benign (Jinak)
- **Fitur (X):**
  Fitur ini dikelompokkan menjadi 3 kategori utama, yaitu
  1. Mean (Rata-rata)
  2. Standard error
  3. Worst (Nilai terburuk atau terbesar)

  Terdapat 10 karakteristik pada masing-masing 3 kategori utama ini (radius, texture, dll) sehingga total terdapat 30 fitur numerik

TOOLS USED

table of CONTENT

1. IMPORT LIBRARY

Beberapa library yang digunakan dalam model klasifikasi ini

```
[1]  import numpy as np
     import pandas as pd
     import matplotlib.pyplot as plt
     from sklearn import datasets
     from sklearn.preprocessing import StandardScaler
     from sklearn.model_selection import train_test_split
     from sklearn.tree import DecisionTreeClassifier
     from sklearn.linear_model import LogisticRegression
     from sklearn import tree
     from sklearn.metrics import accuracy_score
```

2. LOAD DATASET

```python
# Mengambil dataset mengenai breast cancer dari scikit-learn dan mengonversi ke DataFrame
breastCancer = datasets.load_breast_cancer()

x = breastCancer.data #inputan
y = breastCancer.target #output

# Mengonversi data fitur dan target menjadi DataFrame
df_x = pd.DataFrame(x, columns=breastCancer.feature_names)
df_y = pd.Series(y, name='target')

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

df.head(10)
```

	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
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	0
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	0
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	0
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	0
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	0
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	0
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	0
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	0
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	0
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	0

10 rows × 31 columns

3. EXPLORATORY DATA ANALYSIS

```
[ ] 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])
```

```
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
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
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
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
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
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
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
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
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

8 rows × 31 columns

4. STANDARDIZATION & SPLIT DATA

```
# Standarisasi dengan StancadardScaler
scaler = StandardScaler()
x_scaled = scaler.fit_transform(df_x)

# Membagi data menjadi data train dan test
x_train, x_test, y_train, y_test = train_test_split(x_scaled, df_y, test_size=0.2, random_state=50)
```

5. TRAINING CLASSIFICATION MODELS

```
# Membuat dan melatih model Decision Tree
tree_model = DecisionTreeClassifier(random_state=50)
tree_model.fit(x_train, y_train)
```

DecisionTreeClassifier

DecisionTreeClassifier(random_state=50)

DECISION TREE MODEL

```
# Membuat dan melatih model Logistic Regression
log_reg = LogisticRegression(solver='liblinear')
log_reg.fit(x_train, y_train)
```

LogisticRegression

LogisticRegression(solver='liblinear')

LOGISTIC REGRESSION MODEL

6. EVALUATION

```
# Prediksi dengan Decision Tree
y_pred_tree = tree_model.predict(x_test)
acc_tree = accuracy_score(y_test, y_pred_tree)

# Prediksi dengan Logistic Regression
y_pred_log = log_reg.predict(x_test)
acc_log = accuracy_score(y_test, y_pred_log)

print("LAPORAN KLASIFIKASI")
print(f"Akurasi Decision Tree: {acc_tree*100:.2f}%")
print(f"Akurasi Logistic Regression: {acc_log * 100:.2f}%")
```

```
LAPORAN KLASIFIKASI
Akurasi Decision Tree: 91.23%
Akurasi Logistic Regression: 100.00%
```

Hasil dari evaluasi kedua model klasifikasi untuk data kanker payudara:

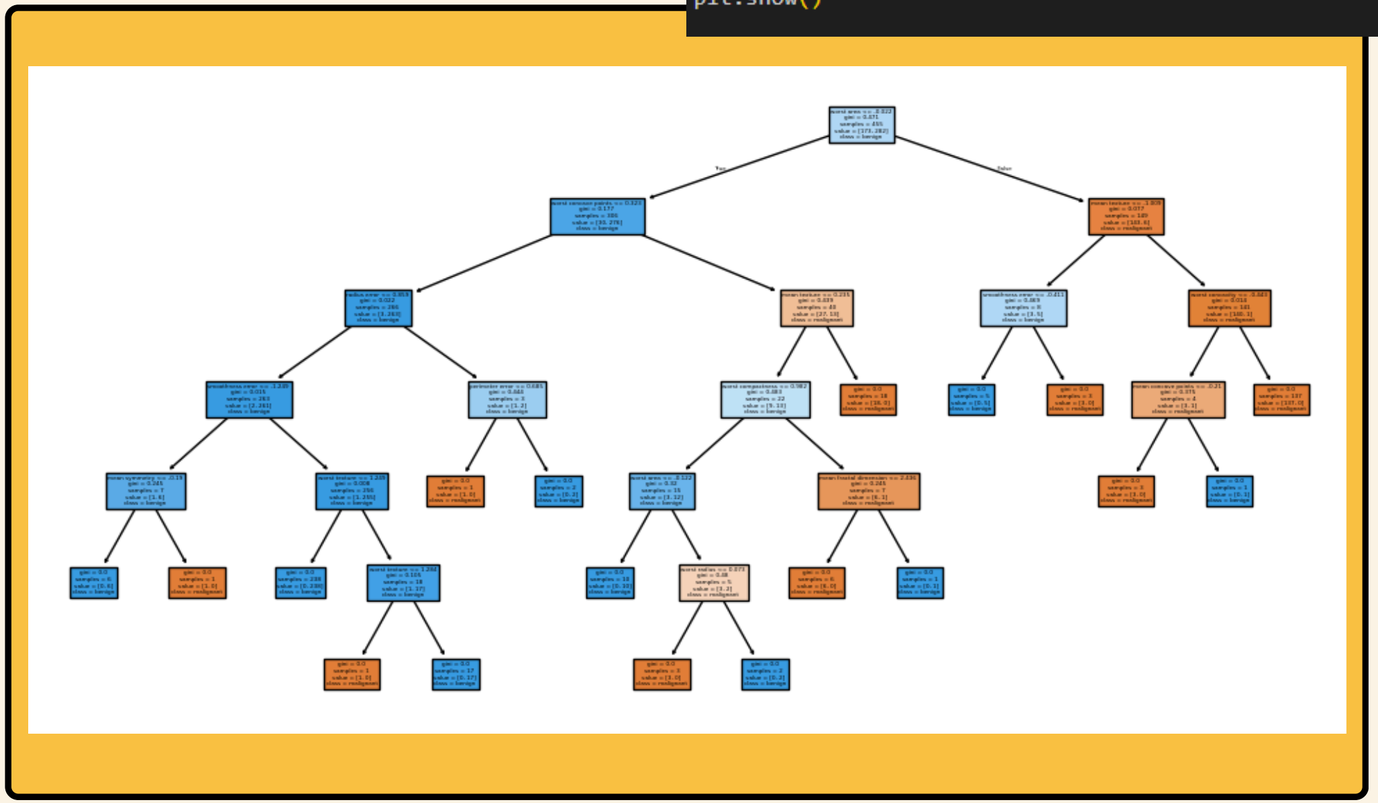
- **Decision Tree** mencapai akurasi 91,23%
- **Logistic Regression** mencapai akurasi 100,00%

Pada Logistic Regression memiliki performa lebih tinggi dibandingkan dengan Decision Tree. Hal ini menunjukkan bahwa pada model Logistic Regression mampu memisahkan kelas dengan sangat baik dalam dataset ini.

VISUALIZATION

Decision Tree

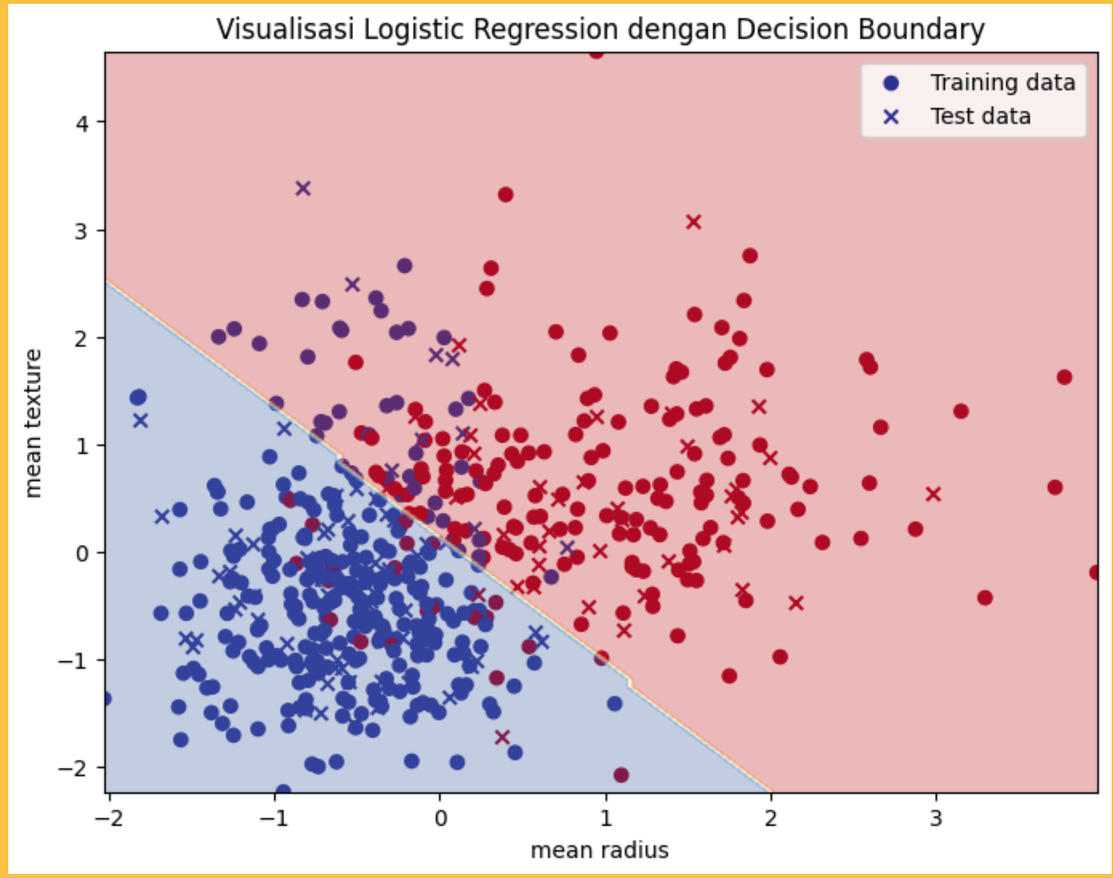
```
# Visualisasi Decision Tree
plt.figure(figsize=(12, 6))
tree.plot_tree(tree_model, filled=True, feature_names=breastCancer.feature_names, class_names=breastCancer.target_names)
plt.show()
```



VISUALIZATION

Logistic Regression

```python
# Visualisasi Koefisien Logistic Regression
# Pilih dua fitur untuk visualisasi
feature1 = 0  # mean radius
feature2 = 1  # mean texture

# Membuat scatter plot
plt.figure(figsize=(8, 6))

# Plot data training
plt.scatter(x_train[:, feature1], x_train[:, feature2], c=y_train, cmap=plt.cm.RdYlBu, marker='o', label='Training data')
plt.scatter(x_test[:, feature1], x_test[:, feature2], c=y_test, cmap=plt.cm.RdYlBu, marker='x', label='Test data')

# Membuat meshgrid untuk plot decision boundary
xx, yy = np.meshgrid(np.linspace(x_train[:, feature1].min(), x_train[:, feature1].max(), 100),
                     np.linspace(x_train[:, feature2].min(), x_train[:, feature2].max(), 100))

# Membuat prediksi untuk setiap titik pada grid
# Menambahkan nilai-nilai fitur lainnya dari rata-rata fitur
Z = np.c_[xx.ravel(), yy.ravel()]
Z = np.hstack([Z, np.tile(x_train[:, 2:].mean(axis=0), (Z.shape[0], 1))])

# Memastikan data memiliki jumlah fitur yang benar
Z = log_reg.predict(Z)
Z = Z.reshape(xx.shape)

# Plot decision boundary
plt.contourf(xx, yy, Z, alpha=0.3, cmap=plt.cm.RdYlBu)

# Menambahkan label dan judul
plt.title('Visualisasi Logistic Regression dengan Decision Boundary')
plt.xlabel(breastCancer.feature_names[feature1])
plt.ylabel(breastCancer.feature_names[feature2])
plt.legend(loc='best')

plt.show()
```



introduce

ABOUT ME

RANIAH MARITZA

Mahasiswa semester 4 jurusan Teknik Informatika Universitas Negeri Surabaya yang memiliki ketertarikan dalam Data Science & Front-End Development. Memiliki keterampilan manajemen waktu, kerja sama tim, dan tanggung jawab yang baik, didukung oleh pengalaman aktif dalam organisasi.

Thank YOU

https://www.linkedin.com/in/raniah-maritza/

https://github.com/Raniah1403

@raaniahrm