Tugas Kecil 1 Machine Learning

Dataset breast cancer

Jon Felix Germinian - 13518025

Filbert Wijaya - 13518077

Import semua library yang diperlukan

agak banyak library yang diminta, dimaklumin

In [1]:
```
import pandas as pd
import sklearn
import sklearn.datasets
import sklearn.model_selection
import sklearn.metrics
```

Import dataset yang digunakan

In [2]:
```
breast_cancer = sklearn.datasets.load_breast_cancer()
```

Make dataframe for given dataset

In [3]:
```
df_breast_cancer = pd.DataFrame(breast_cancer.data, columns=breast_cancer.feature_names
df_breast_cancer.head()
```

Out[3]:

	mean radius	mean texture	mean perimeter	mean area	mean smoothness	mean compactness	mean concavity	mean concave points	mean symmetry	m fra dimen
0	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.3001	0.14710	0.2419	0.07
1	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.0869	0.07017	0.1812	0.05
2	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.1974	0.12790	0.2069	0.05
3	11.42	20.38	77.58	386.1	0.14250	0.28390	0.2414	0.10520	0.2597	0.09
4	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.1980	0.10430	0.1809	0.05

5 rows × 30 columns

In [4]:
```
df_breast_cancer.columns
```

Out[4]:
```
Index(['mean radius', 'mean texture', 'mean perimeter', 'mean area',
       'mean smoothness', 'mean compactness', 'mean concavity',
       'mean concave points', 'mean symmetry', 'mean fractal dimension',
       'radius error', 'texture error', 'perimeter error', 'area error',
```

```
       'smoothness error', 'compactness error', 'concavity error',
       'concave points error', 'symmetry error', 'fractal dimension error',
       'worst radius', 'worst texture', 'worst perimeter', 'worst area',
       'worst smoothness', 'worst compactness', 'worst concavity',
       'worst concave points', 'worst symmetry', 'worst fractal dimension'],
      dtype='object')
```

Feature engineering

In [5]:

```
X, y = sklearn.datasets.load_breast_cancer(return_X_y=True)
X_train, X_test, y_train, y_test = sklearn.model_selection.train_test_split(X, y, rando
```

Training

Decision Tree

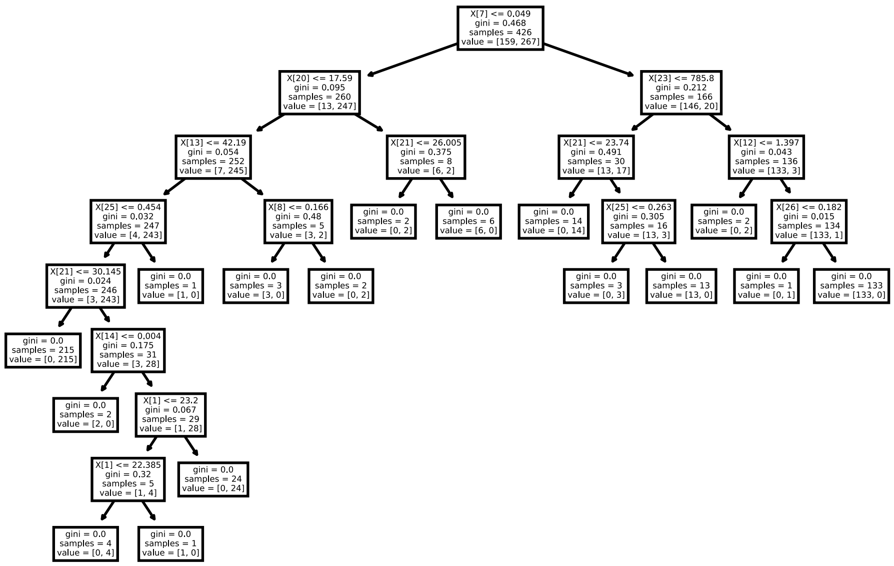
In [6]:

```
from sklearn import tree
clf1 = tree.DecisionTreeClassifier()
clf1 = clf1.fit(X_train, y_train)
tree.plot_tree(clf1)
```

Out[6]:

```
[Text(187.32857142857142, 205.35999999999999, 'X[7] <= 0.049\ngini = 0.468\nsamples = 42
6\nvalue = [159, 267]'),
 Text(119.57142857142857, 181.2, 'X[20] <= 17.59\ngini = 0.095\nsamples = 260\nvalue =
[13, 247]'),
 Text(79.71428571428572, 157.04, 'X[13] <= 42.19\ngini = 0.054\nsamples = 252\nvalue =
[7, 245]'),
 Text(47.82857142857143, 132.88, 'X[25] <= 0.454\ngini = 0.032\nsamples = 247\nvalue =
[4, 243]'),
 Text(31.885714285714286, 108.72, 'X[21] <= 30.145\ngini = 0.024\nsamples = 246\nvalue =
[3, 243]'),
 Text(15.942857142857143, 84.56, 'gini = 0.0\nsamples = 215\nvalue = [0, 215]'),
 Text(47.82857142857143, 84.56, 'X[14] <= 0.004\ngini = 0.175\nsamples = 31\nvalue = [3,
28]'),
 Text(31.885714285714286, 60.400000000000006, 'gini = 0.0\nsamples = 2\nvalue = [2,
0]'),
 Text(63.77142857142857, 60.400000000000006, 'X[1] <= 23.2\ngini = 0.067\nsamples = 29\n
value = [1, 28]'),
 Text(47.82857142857143, 36.24000000000001, 'X[1] <= 22.385\ngini = 0.32\nsamples = 5\nv
alue = [1, 4]'),
 Text(31.885714285714286, 12.079999999999984, 'gini = 0.0\nsamples = 4\nvalue = [0,
4]'),
 Text(63.77142857142857, 12.079999999999984, 'gini = 0.0\nsamples = 1\nvalue = [1, 0]'),
 Text(79.71428571428572, 36.24000000000001, 'gini = 0.0\nsamples = 24\nvalue = [0, 2
4]'),
 Text(63.77142857142857, 108.72, 'gini = 0.0\nsamples = 1\nvalue = [1, 0]'),
 Text(111.6, 132.88, 'X[8] <= 0.166\ngini = 0.48\nsamples = 5\nvalue = [3, 2]'),
 Text(95.65714285714286, 108.72, 'gini = 0.0\nsamples = 3\nvalue = [3, 0]'),
 Text(127.54285714285714, 108.72, 'gini = 0.0\nsamples = 2\nvalue = [0, 2]'),
 Text(159.42857142857144, 157.04, 'X[21] <= 26.005\ngini = 0.375\nsamples = 8\nvalue =
[6, 2]'),
 Text(143.4857142857143, 132.88, 'gini = 0.0\nsamples = 2\nvalue = [0, 2]'),
 Text(175.37142857142857, 132.88, 'gini = 0.0\nsamples = 6\nvalue = [6, 0]'),
 Text(255.0857142857143, 181.2, 'X[23] <= 785.8\ngini = 0.212\nsamples = 166\nvalue = [1
46, 20]'),
 Text(223.2, 157.04, 'X[21] <= 23.74\ngini = 0.491\nsamples = 30\nvalue = [13, 17]'),
 Text(207.25714285714287, 132.88, 'gini = 0.0\nsamples = 14\nvalue = [0, 14]'),
 Text(239.14285714285714, 132.88, 'X[25] <= 0.263\ngini = 0.305\nsamples = 16\nvalue =
```

```
[13, 3]'),
 Text(223.2, 108.72, 'gini = 0.0\nsamples = 3\nvalue = [0, 3]'),
 Text(255.0857142857143, 108.72, 'gini = 0.0\nsamples = 13\nvalue = [13, 0]'),
 Text(286.9714285714286, 157.04, 'X[12] <= 1.397\ngini = 0.043\nsamples = 136\nvalue =
[133, 3]'),
 Text(271.0285714285714, 132.88, 'gini = 0.0\nsamples = 2\nvalue = [0, 2]'),
 Text(302.9142857142857, 132.88, 'X[26] <= 0.182\ngini = 0.015\nsamples = 134\nvalue =
[133, 1]'),
 Text(286.9714285714286, 108.72, 'gini = 0.0\nsamples = 1\nvalue = [0, 1]'),
 Text(318.8571428571429, 108.72, 'gini = 0.0\nsamples = 133\nvalue = [133, 0]')]
```



In [7]:
```python
print("Training set score: %f" % clf1.score(X_train, y_train))
print("Test set score: %f" % clf1.score(X_test, y_test))
# overfit?
```

```
Training set score: 1.000000
Test set score: 0.881119
```

In [8]:
```python
print("Accuracy score: %f" % sklearn.metrics.accuracy_score(y_test, clf1.predict(X_test
print("F1 Score : %f" % sklearn.metrics.f1_score(y_test, clf1.predict(X_test),average='
```

```
Accuracy score: 0.881119
F1 Score : 0.882642
```

Id3Estimator

In [9]:
```python
import six
import sys
sys.modules['sklearn.externals.six'] = six
import mlrose
```

In [10]:
```python
import id3
estimator = id3.Id3Estimator()
estimator = estimator.fit(X_train, y_train)
tree = id3.export_graphviz(estimator.tree_, 'tree.dot', breast_cancer.feature_names)
```

In [11]:

```
print("Accuracy score: %f" % sklearn.metrics.accuracy_score(y_test, estimator.predict(X
print("F1 Score : %f" % sklearn.metrics.f1_score(y_test, estimator.predict(X_test),aver
```

```
Accuracy score: 0.944056
F1 Score : 0.944437
```

KMeans

In [12]:

```
from sklearn.cluster import KMeans

kmeans = KMeans(n_clusters=2).fit(df_breast_cancer)

centroids = kmeans.cluster_centers_
print(centroids)
```

```
[[1.25562991e+01 1.85703653e+01 8.11234703e+01 4.96061872e+02
  9.48844977e-02 9.10998174e-02 6.24377642e-02 3.34325434e-02
  1.78057991e-01 6.34540183e-02 3.04190868e-01 1.21515320e+00
  2.15288059e+00 2.37852922e+01 7.17326256e-03 2.34746895e-02
  2.87455128e-02 1.06363242e-02 2.06135799e-02 3.74750297e-03
  1.40439018e+01 2.47095434e+01 9.19375114e+01 6.19647945e+02
  1.29959110e-01 2.23311758e-01 2.19214947e-01 9.13298425e-02
  2.83553653e-01 8.32819406e-02]
 [1.93799237e+01 2.16945802e+01 1.28231298e+02 1.18592977e+03
  1.01294580e-01 1.48612977e-01 1.76939466e-01 1.00698779e-01
  1.91539695e-01 6.06029008e-02 7.42803817e-01 1.22253817e+00
  5.25058015e+00 9.56781679e+01 6.59868702e-03 3.21766947e-02
  4.24197710e-02 1.56739847e-02 2.03039695e-02 3.95338931e-03
  2.37094656e+01 2.89126718e+01 1.58496183e+02 1.75302290e+03
  1.40424733e-01 3.57757710e-01 4.49306107e-01 1.92431069e-01
  3.11881679e-01 8.61654962e-02]]
```

In [13]:

```
print("Accuracy score: %f" % sklearn.metrics.accuracy_score(y_test, kmeans.predict(X_te
print("F1 Score : %f" % sklearn.metrics.f1_score(y_test, kmeans.predict(X_test),average
```

```
Accuracy score: 0.153846
F1 Score : 0.098834
```

Logistic Regression

In [14]:

```
from sklearn.linear_model import LogisticRegression
clf3 = LogisticRegression(random_state=0).fit(X_train, y_train)
```

In [15]:

```
print("Training set score: %f" % clf3.score(X_train, y_train))
print("Test set score: %f" % clf3.score(X_test, y_test))
```

```
Training set score: 0.946009
Test set score: 0.944056
```

In [16]:

```
print("Accuracy score: %f" % sklearn.metrics.accuracy_score(y_test, clf3.predict(X_test
print("F1 Score : %f" % sklearn.metrics.f1_score(y_test, clf3.predict(X_test),average='
```

```
Accuracy score: 0.944056
F1 Score : 0.944437
```

Neural Network

In [17]:
```
from sklearn.neural_network import MLPClassifier
clf4 = MLPClassifier(random_state=1, max_iter=300).fit(X_train, y_train)
```

In [18]:
```
print("Training set score: %f" % clf4.score(X_train, y_train))
print("Test set score: %f" % clf4.score(X_test, y_test))
```

```
Training set score: 0.927230
Test set score: 0.895105
```

In [19]:
```
print("Accuracy score: %f" % sklearn.metrics.accuracy_score(y_test, clf4.predict(X_test
print("F1 Score : %f" % sklearn.metrics.f1_score(y_test, clf4.predict(X_test),average='
```

```
Accuracy score: 0.895105
F1 Score : 0.896232
```

SVM

In [20]:
```
from sklearn.pipeline import make_pipeline
from sklearn.preprocessing import StandardScaler
from sklearn.svm import SVC
clf5 = make_pipeline(StandardScaler(), SVC(gamma='auto'))
clf5.fit(X_train, y_train)
```

Out[20]:
```
Pipeline(steps=[('standardscaler', StandardScaler()),
                ('svc', SVC(gamma='auto'))])
```

In [21]:
```
print("Training set score: %f" % clf5.score(X_train, y_train))
print("Test set score: %f" % clf5.score(X_test, y_test))
```

```
Training set score: 0.985915
Test set score: 0.965035
```

In [22]:
```
print("Accuracy score: %f" % sklearn.metrics.accuracy_score(y_test, clf5.predict(X_test
print("F1 Score : %f" % sklearn.metrics.f1_score(y_test, clf5.predict(X_test),average='
```

```
Accuracy score: 0.965035
F1 Score : 0.964965
```

Abalysis

Dapat dilihat bahwa skor akurasi dari terbaik ke terburuk adalah SVM, ID3, Logistic Regression, Decision Tree, Neural Network dan KMeans.
Untuk nilai F1, skor dari terbaik ke terburuk adalah SVM, ID3, Logistic Regression, Decision Tree, Neural Network dan KMeans.

SVM mungkin terbaik dalam kasus data breast cancer karena nilai-nilai dari dataframe tersebut linearly seperable.
ID3, Logistic Regression, Decision Tree memiliki nilai baik mungkin dikarenakan algoritmanya yang cocok digunakan untuk klasifikasi dengan banyak parameter (ada 30 kolom untuk dataset ini).
Performa neural network kurang dibandingkan algoritma lain karena ukuran sampel yang kurang besar.

KMeans memiliki performa yang sangat buruk untuk dataset ini, mungkin dikarenakan variasi data yang besar dan dimensi dataset.