LAPORAN UTS KECERDASAN KOMPUTASIONAL

“Classification using Decicsion Tree, Confused Matrix, Accuration on Breast Cancer Dataset”

Nama: Ilham Adikusuma

NIM: 202410103034

Prodi: Informatika

Kelas: Kecerdasan Komputasional B

I. Penjelasan mengenai alur algoritma beserta contoh perhitungannya

1. Mengimport library yang diperlukan

```
#mengimport library
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import confusion_matrix, accuracy_score
from sklearn.tree import DecisionTreeClassifier
from sklearn import tree
import pydotplus
from IPython.display import Image
import itertools
# import warnings
# warnings.filterwarnings(action='ignore')
```

2. Mengupload breast-cancer.csv . Data dibaca menggunakan pandas, dengan separator “;”

```
from google.colab import files              #memanggil library google.colab
uploaded = files.upload()                   #upload dataframe 'breast-cancer.csv'
```

Choose Files breast-cancer.csv

- **breast-cancer.csv**(application/vnd.ms-excel) - 19252 bytes, last modified: 10/14/2021 - 100% done

```
Saving breast-cancer.csv to breast-cancer (2).csv
```

```
data = pd.read_csv('breast-cancer.csv', sep = ';')
data
```

	Class	Age	Menopause	tumor-size	inv-node	node-capes	deg-malig	breast	breast-quad	irradiat
0	no-recurrence-events	30-39	premeno	30-34	0-2	no	3	left	left_low	no
1	no-recurrence-events	40-49	premeno	20-24	0-2	no	2	right	right_up	no
2	no-recurrence-events	40-49	premeno	20-24	0-2	no	2	left	left_low	no
3	no-recurrence-events	60-69	ge40	15-19	0-2	no	2	right	left_up	no
4	no-recurrence-events	40-49	premeno	0-4	0-2	no	2	right	right_low	no
...	...	...	...	...	...	...	...	...	...	...

3. Mengecek **Data** apakah ada yang **kosong**. Ternyata tidak ada maka proses **Cleaning** dilewati

```
#menghitung total data yang kosong di dataframe
data.isna().sum()
```

```
Class          0
Age            0
Menopause      0
tumor-size     0
inv-node       0
node-capes     0
deg-malig      0
breast         0
breast-quad    0
irradiat       0
dtype: int64
```

4. Lalu ubah data di masing-masing kolom yang masih berupa **string** ke **integer** agar proses klasifikasi lebih mudah. Proses ini disebut **Prepocessing**.
   Class tidak diikut sertakan karena nantinya akan menjadi sebuah hasil

```
[39] # mengubah data berupa kategori menjadi dalam bentuk angka
     from sklearn.preprocessing import LabelEncoder
     le = LabelEncoder()

     data['Age'] = le.fit_transform(data['Age'])
     data['Menopause'] = le.fit_transform(data['Menopause'])
     data['tumor-size'] = le.fit_transform(data['tumor-size'])
     data['inv-node'] = le.fit_transform(data['inv-node'])
     data['node-capes'] = le.fit_transform(data['node-capes'])
     data['breast'] = le.fit_transform(data['breast'])
     data['breast-quad'] = le.fit_transform(data['breast-quad'])
     data['irradiat'] = le.fit_transform(data['irradiat'])
```

	Class	Age	Menopause	...	breast	breast-quad	irradiat
0	no-recurrence-events	1	2	...	0	2	0
1	no-recurrence-events	2	2	...	1	5	0
2	no-recurrence-events	2	2	...	0	2	0
3	no-recurrence-events	4	0	...	1	3	0
4	no-recurrence-events	2	2	...	1	4	0
..	...	...	...	...	...	...	...
281	recurrence-events	1	2	...	0	3	0
282	recurrence-events	1	2	...	0	3	1
283	recurrence-events	4	0	...	1	3	0
284	recurrence-events	2	0	...	0	2	0
285	recurrence-events	3	0	...	0	2	0

[286 rows x 10 columns]

5. Split Data menjadi
   - Data test :
     - X_test = Class
     - Y_test = Data random
   - Data train :
     - X_train = Data
     - Y_train = ½ Data

   Ket:
   Test_size = 0.5 -> mengambil ½ data dari keseluruhan
   Random state = 124 -> untuk setiap 124 data, data akan di shuffle

```
[74] # Memisahkan kolom Class dengan kolom lainnya
     x=data.drop(['Class'], axis=1)
     y=data['Class']


     # Membagi data menjadi data train dan data test
     x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.5, random_state= 143)
```

6. Disini saya membuat 2 cara dalam mencari Decision Tree. Pertama dengan menggunakan library, dan Kedua cara manual

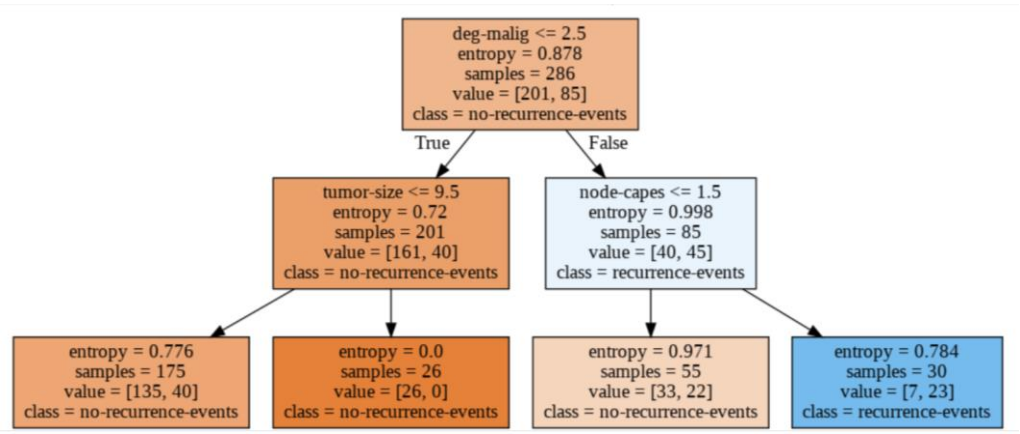
7. Menggunakan Library
    a. fitting **X_train** dan **Y_train**
    b. membuat variabel **Prediksi X_test**
    c. lalu **accuracy_score(Y_test, Prediksi)** untuk mendapatkan hasil akurasinya
    d. lalu **confusion_matrix(Y_test, Prediksi)** untuk membuat confusion matriks

```
[77] DTC = DecisionTreeClassifier()

     # Membuat model svm berdasarkan data train yang diberikan
     DTC.fit(x_train,y_train)
     # Membuat prediksi
     predictions = DTC.predict(x_test)
     # Mendapatkan score akurasi prediksi
     acc_score = accuracy_score(y_test, predictions)

     print("Akurasi model Decision Tree: {}".format(format(acc_score,'.3f')), '\n')
     print("Confusion matriks : \n", confusion_matrix(y_test, predictions),"\n")

     Akurasi model Decision Tree: 0.650

     Confusion matriks :
      [[79 24]
      [26 14]]
```

8. Cara Manual
    a. Proses pada **Decision Tree** adalah mengubah bentuk data (tabel) menjadi model pohon, mengubah model pohon menjadi rule, dan menyederhanakan rule. **Entropi** adalah nilai informasi yang menyatakan ukuran ketidakpastian(impurity) dari attribut dari suatu kumpulan obyek data dalam satuan bit. **Range Entropi** ada di antara 0 dan 1, semakin mendekati 1 semakin tidak pasti.

```
[62] #maximal kedalaman tree adalah 2
     clf = tree.DecisionTreeClassifier(criterion='entropy',max_depth=2)
     clf = clf.fit(x, y)
```

```
[63] dot_data = tree.export_graphviz(clf, feature_names=x.columns, class_names=['no-recurrence-events','recurrence-events'], filled=True,
                                     out_file=None)

     graph = pydotplus.graph_from_dot_data(dot_data)
     Image(graph.create_png())
```

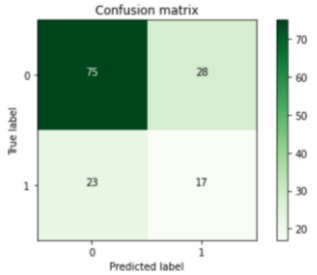


b. **Visualisasikan** Data dalam bentuk **Confusion Matriks**. Jika Data ingin diperlihatkan dalam **skala normal**, Normalize di set **False**. Jika Data ingin diperlihatkan dalam **skala range 0-1**, Normalize di set **True**

```
[67] def plot_confusion_matrix(cm, classes,
                               normalize=False,
                               title='Confusion matrix',
                               cmap=plt.cm.Greens):

        if normalize:
            cm = cm.astype('float') / cm.sum(axis=1)[:, np.newaxis]
            print("Normalized confusion matrix")
        else:
            print('Confusion matrix, without normalization')

        plt.imshow(cm, interpolation='nearest', cmap=cmap)
        plt.title(title)
        plt.colorbar()
        tick_marks = np.arange(len(classes))
        plt.xticks(tick_marks, classes)
        plt.yticks(tick_marks, classes)
```

```
        fmt = '.2f' if normalize else 'd'
        thresh = cm.max() / 2.
        for i, j in itertools.product(range(cm.shape[0]), range(cm.shape[1])):
            plt.text(j, i, format(cm[i, j], fmt),
                     horizontalalignment="center",
                     color="white" if cm[i, j] > thresh else "black")

        plt.ylabel('True label')
        plt.xlabel('Predicted label')
        plt.tight_layout()


    clf = DecisionTreeClassifier()
    clf.fit(x_train, y_train)
    y_pred = clf.predict(x_test)
    decision_tree_cm = confusion_matrix(y_test, y_pred)
    plot_confusion_matrix(decision_tree_cm, [0, 1])
    plt.show()
```

```
Confusion matrix, without normalization
```



c. Menghitung **akurasi** berdasarkan Confusuion Matriks.

```
arr = decision_tree_cm

TP = arr[0,0]
FP = arr[0,1]
FN = arr[1,0]
TN = arr[1,1]

akurasi = (TP + TN) / (TP + FP + FN + TN)
print("Akurasi :", akurasi)
```

```
Akurasi : 0.6433566433566433
```

Ket=> TP :True Positive

FP :False Positive

FN :False Negative

TN :True Negative

II. Jika algoritma yang digunakan terdapat proses training maka jelaskan apa yang didapatkan dari proses training

Hal yang perlu dilakukan sebelum proses training yaitu menyiapkan data tes terlebih dahulu. Lalu proses training bisa dilakukan dengan cara/mekanisme dari yang sudah dilakukan di data tes.

III. Penjelasan mengenai hasil evaluasi menggunakan confussion matrix dan accuracy

[TP FP

FN TN]

a. Di dalam confussion matrix terdapat TP, TN, FP, FN (Hal. 5) .
   - TP diprediksi benar dan kesimpulannya positif. Contoh: wanita dinyatakan hamil
   - FP diprediksi benar dan kesimpulannya negatif. Contoh: pria dinyatakan hamil
   - FN diprediksi salah dan kesimpulannya positif. Contoh: wanita tidak dinyatakan hamil
   - TN diprediksi salah dan kesimpulannya negatif. Contoh: pria tidak dinyatakan hamil

Untuk kasus pada breast cancer ini bisa seperti ini:

- TP: diprediksi reccurance dan kesimpulannya positif
- FP: diprediksi reccurance dan kesimpulannya negatif
- FN:diprediksi no reccurance dan kesimpulannya positif
- TN:diprediksi no reccurance dan kesimpulannya negatif

Misalkan ambil data dan ditampilkan sebagai berikut:

```
testData=  [["recurrence-events",1,2,5,0,1,2,0,3,0],
            ["recurrence-events",1,2,0,3,0,1,3,2,1],
            ["recurrence-events",4,0,3,0,1,1,1,3,0],
            ["no-recurrence-events",2,2,3,0,1,2,1,5,0],
            ["no-recurrence-events",2,2,3,0,1,2,0,2,0],]
testData = pd.DataFrame(testData, columns=data.columns)
testData
```

	Class	Age	Menopause	tumor-size	inv-node	node-capes	deg-malig	breast	breast-quad	irradiat
0	recurrence-events	1	2	5	0	1	2	0	3	0
1	recurrence-events	1	2	0	3	0	1	3	2	1
2	recurrence-events	4	0	3	0	1	1	1	3	0
3	no-recurrence-events	2	2	3	0	1	2	1	5	0
4	no-recurrence-events	2	2	3	0	1	2	0	2	0

Lalu dites menghasilkan sebagai berikut:

```
testY = testData['Class']
testX = testData.drop(['Class'],axis=1)

predY = clf.predict(testX)
predictions = pd.concat([pd.Series(predY,name='Predicted Class')], axis=1)
predictions
```

	Predicted Class
0	no-recurrence-events
1	recurrence-events
2	recurrence-events
3	no-recurrence-events
4	no-recurrence-events

data[0] merupakan FN

data[1] merupakan TP

data[2] merupakan TP

data[3] merupakan TN

data[4] merupakan TN

b. Akurasi bisa didapatkan dengan rumus sebagai berikut:

akurasi = (TP + TN) / (TP + FP + FN + TN)

Dari total 286 diambil setengah menjadi 143 karena test_size = 0.5
Lalu 143 data ini di train dan dijadikan confusion matriksnya
Didapatkan data di TP = 75, ,FP=28, FN=23, TN=17
Di aplikasikan ke rumus menghasilkan 64%