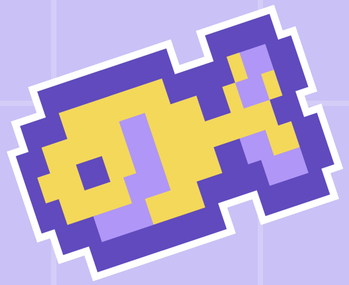
LAPORAN PRAKTIKUM UTS MACHINE LEARNING

Klasifikasi dengan Studi Kasus Water Potability

Dosen Pengampu : Entin Martiana Kusumaningtyas S.Kom, M.Kom.

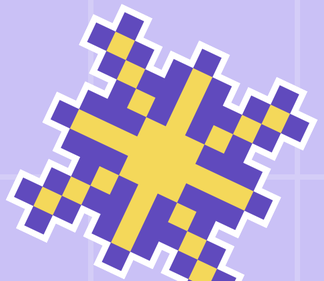
Bayu Kurniawan / 3322600019

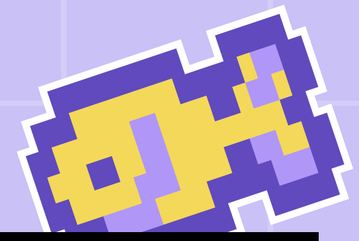
Start



```python
# Assignment 1
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.feature_selection import RFE
from sklearn.metrics import accuracy_score
from sklearn.naive_bayes import GaussianNB as GNB
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier as dtc
from sklearn.impute import SimpleImputer

# menampilkan data water_potability.csv
dataset = pd.read_csv("C:/Users/bayuk/OneDrive/Documents/AI/pens/smtr3/Machine Learning/Data/water_potability.csv")
dataset
```

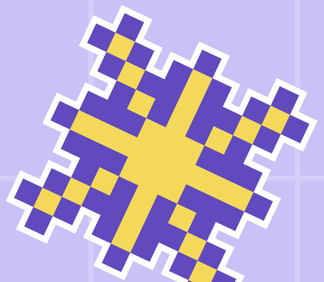
✓ 0.0s





	ph	Hardness	Solids	Chloramines	Sulfate	Conductivity	Organic_carbon	Trihalomethanes	Turbidity	Potability
0	NaN	204.890456	20791.31898	7.300212	368.516441	564.308654	10.379783	86.990970	2.963135	0
1	3.716080	129.422921	18630.05786	6.635246	NaN	592.885359	15.180013	56.329076	4.500656	0
2	8.099124	224.236259	19909.54173	9.275884	NaN	418.606213	16.868637	66.420093	3.055934	0
3	8.316766	214.373394	22018.41744	8.059332	356.886136	363.266516	18.436525	100.341674	4.628771	0
4	9.092223	181.101509	17978.98634	6.546600	310.135738	398.410813	11.558279	31.997993	4.075075	0
...	...	...	...	...	...	...	...	...	...	...
3271	4.668102	193.681736	47580.99160	7.166639	359.948574	526.424171	13.894419	66.687695	4.435821	1
3272	7.808856	193.553212	17329.80216	8.061362	NaN	392.449580	19.903225	NaN	2.798243	1
3273	9.419510	175.762646	33155.57822	7.350233	NaN	432.044783	11.039070	69.845400	3.298875	1
3274	5.126763	230.603758	11983.86938	6.303357	NaN	402.883113	11.168946	77.488213	4.708658	1
3275	7.874671	195.102299	17404.17706	7.509306	NaN	327.459761	16.140368	78.698446	2.309149	1

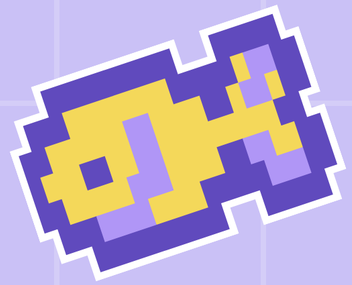
3276 rows × 10 columns



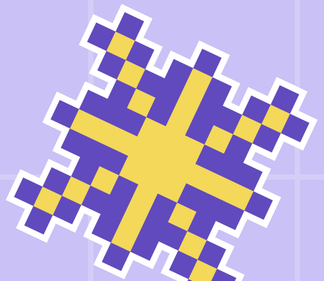
```python
# Assignment 2
# Pilih fitur yang memiliki korelasi tinggi dengan target 'Potability' menggunakan RFE

# Memilih atribut yang akan digunakan
selected_features = ['ph', 'Hardness', 'Solids', 'Chloramines', 'Sulfate', 'Conductivity', 'Organic_carbon', 'Trihalomethanes', 'Turbidity']

# Memisahkan fitur dan target
X = dataset[selected_features]
y = dataset['Potability']

# Langkah 1: Lakukan pengisian missing value jika ada atribut yang kosong
imputer = SimpleImputer(strategy='mean')
X = imputer.fit_transform(X)

# Langkah 2: Inisialisasi model Decision Tree
model = dtc()

# Langkah 3: Inisialisasi RFE
rfe = RFE(model, n_features_to_select=7)  # Mengambil 7 fitur terbaik

# Langkah 4: Melatih RFE
fit = rfe.fit(X, y)

# Mengambil indeks dari fitur-fitur yang terpilih
selected_features_indices = fit.support_

# Mendapatkan nama fitur yang terpilih
selected_features = dataset[selected_features].columns[selected_features_indices]

print("Fitur yang terpilih:")
print(selected_features)
```

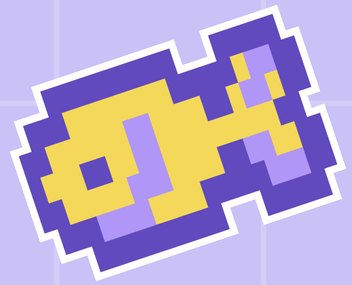
✓ 0.0s

```
Fitur yang terpilih:
Index(['ph', 'Hardness', 'Solids', 'Chloramines', 'Sulfate', 'Conductivity',
       'Trihalomethanes'],
      dtype='object')
```



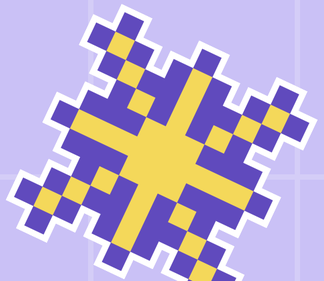
```
Fitur yang terpilih:
Index(['ph', 'Hardness', 'Solids', 'Chloramines', 'Sulfate', 'Conductivity',
       'Trihalomethanes'],
      dtype='object')
```





```
# assignment 3
# Validasi model Hold-out Method (70%-30%)
datalabel = dataset.loc[:,['Potability']]
xtrain, xtest, ytrain, ytest = train_test_split(dataset, datalabel, test_size = 0.30, random_state = 100)
```

✓ 0.0s



```python
# assignment 4
sel_dataset = dataset[['ph', 'Hardness', 'Solids', 'Chloramines', 'Sulfate', 'Conductivity', 'Trihalomethanes', 'Potability']]
dataset = sel_dataset.fillna(sel_dataset.groupby("Potability").transform("mean"))
print("\n Dataset setelah pengisian mising value\n", dataset)
```

✓ 0.0s

```
 Dataset setelah pengisian mising value
            ph    Hardness       Solids  Chloramines     Sulfate  \
0     7.085378  204.890456  20791.31898     7.300212  368.516441
1     3.716080  129.422921  18630.05786     6.635246  334.564290
2     8.099124  224.236259  19909.54173     9.275884  334.564290
3     8.316766  214.373394  22018.41744     8.059332  356.886136
4     9.092223  181.101509  17978.98634     6.546600  310.135738
...        ...         ...          ...          ...         ...
3271  4.668102  193.681736  47580.99160     7.166639  359.948574
3272  7.808856  193.553212  17329.80216     8.061362  332.566990
3273  9.419510  175.762646  33155.57822     7.350233  332.566990
3274  5.126763  230.603758  11983.86938     6.303357  332.566990
3275  7.874671  195.102299  17404.17706     7.509306  332.566990

      Conductivity  Trihalomethanes  Potability
0       564.308654        86.990970           0
1       592.885359        56.329076           0
2       418.606213        66.420093           0
3       363.266516       100.341674           0
4       398.410813        31.997993           0
...            ...              ...         ...
3271    526.424171        66.687695           1
3272    392.449580        66.539684           1
3273    432.044783        69.845400           1
3274    402.883113        77.488213           1
3275    327.459761        78.698446           1

[3276 rows x 8 columns]
```

```python
# assignment 5
train_data = np.array(xtrain)[:,:-1]
newmin = 0
newmax = 1
mindata = train_data.min()
maxdata = train_data.max()
train_data = ((train_data-mindata)*(newmax-newmin)/(maxdata-mindata))+newmin
train_label = ytrain
print("Train data : ", train_data)
```

✓ 0.0s

```
Train data :  [[1.15533354e-04 3.47403354e-03 4.18547357e-01 ... 5.53125991e-03
  5.39390005e-03 1.05299615e-03]
 [1.15722733e-04 3.48769740e-03 3.68132351e-01 ... 5.45576489e-03
  7.56851330e-03 1.13700862e-03]
 [1.40684192e-04 3.85358879e-03 3.40738385e-01 ... 6.18611891e-03
  6.83842405e-03 1.08291020e-03]
 ...
 [1.22179825e-04 1.91185163e-03 4.63159260e-01 ... 6.77567831e-03
  7.70996175e-03 1.24690638e-03]
 [1.30508316e-04 3.45085772e-03 4.40194299e-01 ... 5.43168742e-03
  1.04202091e-02 1.13339206e-03]
 [1.10694378e-04 3.38167941e-03 2.48727917e-01 ... 5.24929157e-03
  8.54590388e-03 1.40731531e-03]]
```

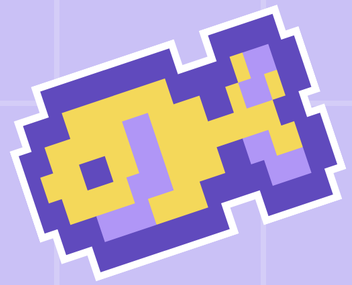
```python
# assignment 6
test_data = np.array(xtest)[:,:-1]
newmin = 0
newmax = 1
mindata = train_data.min()
maxdata = train_data.max()
test_data = ((test_data-mindata)*(newmax-newmin)/(maxdata-mindata))+newmin
test_label = ytest
print("Test data : ", test_data)
```

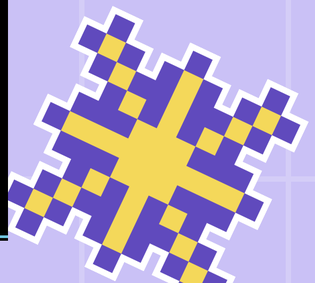
✓ 0.0s

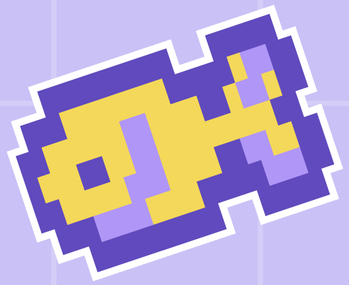
```
Test data :  [[8.07047713e+00 1.98865948e+02 1.82666177e+04 ... 3.96619510e+02
  3.76710304e+02 8.73795608e+01]
 [6.60253977e+00 1.74632977e+02 2.16074832e+04 ... 3.08931421e+02
  6.57570422e+02 6.88270473e+01]
 [8.54589035e+00 2.55324872e+02 3.10565788e+04 ... 3.34564290e+02
  4.26883367e+02 3.44482932e+01]
 ...
 [1.31754017e+01 4.74320000e+01 1.92379497e+04 ... 3.75147315e+02
  5.00245952e+02 6.65396837e+01]
 [7.87467136e+00 1.95102299e+02 1.74041771e+04 ... 3.32566990e+02
  3.27459761e+02 7.86984463e+01]
 [7.30099013e+00 1.82447697e+02 2.91363387e+04 ... 3.32566990e+02
  3.07433303e+02 4.98953419e+01]]
```

```python
# Assignment 7
# Classfication using k-NN
kNN = KNeighborsClassifier(n_neighbors=3, weights='distance')
kNN.fit(train_data, train_label)
kNN_predict = kNN.predict(test_data)
acc_kNN = accuracy_score(test_label, kNN_predict)

# Classfication using bayesian
classifier = GNB()
classifier.fit(train_data, train_label.values.ravel())
byn_pradict = classifier.predict(test_data)
acc_byn = accuracy_score(test_label, byn_pradict)

# Classfication using Decision Tree
dtc = dtc()
dtc.fit(train_data, train_label)
dtc_predict = dtc.predict(test_data)
acc_dtc = 1 - accuracy_score(test_label, dtc_predict)
print("k-NN Error Rate :", acc_kNN)
print("Bayesian Error Rate :", acc_byn)
print("Decision Tree Error Rate: ", acc_dtc)
```

✓ 0.0s







```
k-NN Akurasi Rate : 0.6063072227873856
Bayesian Akurasi Rate : 0.3936927772126144
Decision Tree Akurasi Rate:  0.6063072227873856
C:\Users\bayuk\AppData\Local\Packages\PythonSoftwareFound
  return self._fit(X, y)
```