POLITEKNIK ELEKTRONIKA
NEGERI SURABAYA

MACHINE
LEARNING OPS

LAPORAN PRAKTIKUM

ML SELECTION AND EVALUATION

Dosen Pengampu :
Ibu Tri Hadiah Muliawati S.ST., M.Kom.

Disusun Oleh :
Reyna Aisyana (3321600002)

PROGRAM STUDI
SAINS DATA TERAPAN
2022/2023

PRAKTIKUM KE – 2

1. DATASET

- **Judul** : Milk Quality Prediction
- **Link** : https://www.kaggle.com/datasets/cpluzshrijayan/milkquality

pH	Temprature	Taste	Odor	Fat	Turbidity	Colour	Grade
6.6	35	1	0	1	0	254	high
6.6	36	0	1	0	1	253	high
8.5	70	1	1	1	1	246	low
9.5	34	1	1	0	1	255	low
6.6	37	0	0	0	0	255	medium
6.6	37	1	1	1	1	255	high
5.5	45	1	0	1	1	250	low
4.5	60	0	1	1	1	250	low
8.1	66	1	0	1	1	255	low

Analisis

Data yang digunakan memiliki 7 feature, yaitu pH, Temperature, Taste, Odor, Fat, Turbidity, dan Colour. Serta satu class label, yaitu Grade yang di dalamnya terbagi menjadi tiga kelas, yaitu low, medium, dan high.

Masalah yang akan diselesaikan kali ini adalah untuk mengetahui kualitas susu dari ketujuh feature yang ada, yaitu pH, Temperature, Taste, Odor, Fat, Turbidity, dan Colour. Berdasarkan permasalahan tersebut, maka yang akan diprediksi pada kasus kali ini adalah kelas pada data. Dimana outputnya nanti bersifat bilangan diskrit yang terbatas dalam tiga kelas. Oleh karena itu, pemodelan yang akan dilakukan terhadap data di atas adalah menggunakan **algoritma klasifikasi**.

2. MODEL KLASIFIKASI

Model klasifikasi yang dirasa cocok dengan data di atas adalah **Decision Tree.** Hal ini dikarenakan algoritma decision tree akan menghasilkan sebuah keputusan sebab-akibat dengan pernyataan tertentu.

a) Entry Data

```
import pandas as pd
data = pd.read_csv('milknew.csv')
data.head(5)
```

✓ 0.9s Python

	pH	Temprature	Taste	Odor	Fat	Turbidity	Colour	Grade
0	6.6	35	1	0	1	0	254	high
1	6.6	36	0	1	0	1	253	high
2	8.5	70	1	1	1	1	246	low
3	9.5	34	1	1	0	1	255	low
4	6.6	37	0	0	0	0	255	medium

b) Analyze Data

```
data.shape
```

✓ 0.0s Python

```
(1059, 8)
```

```
data.info()
```

✓ 0.1s Python

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1059 entries, 0 to 1058
Data columns (total 8 columns):
 #   Column      Non-Null Count  Dtype  
---  ------      --------------  -----  
 0   pH          1059 non-null   float64
 1   Temprature  1059 non-null   int64  
 2   Taste       1059 non-null   int64  
 3   Odor        1059 non-null   int64  
 4   Fat         1059 non-null   int64  
 5   Turbidity   1059 non-null   int64  
 6   Colour      1059 non-null   int64  
 7   Grade       1059 non-null   object 
dtypes: float64(1), int64(6), object(1)
memory usage: 66.3+ KB
```

```
data.describe()
```

✓ 0.1s Python

	pH	Temprature	Taste	Odor	Fat	Turbidity	Colour
count	1059.000000	1059.000000	1059.000000	1059.000000	1059.000000	1059.000000	1059.000000
mean	6.630123	44.226629	0.546742	0.432483	0.671388	0.491029	251.840415
std	1.399679	10.098364	0.498046	0.495655	0.469930	0.500156	4.307424
min	3.000000	34.000000	0.000000	0.000000	0.000000	0.000000	240.000000
25%	6.500000	38.000000	0.000000	0.000000	0.000000	0.000000	250.000000
50%	6.700000	41.000000	1.000000	0.000000	1.000000	0.000000	255.000000
75%	6.800000	45.000000	1.000000	1.000000	1.000000	1.000000	255.000000
max	9.500000	90.000000	1.000000	1.000000	1.000000	1.000000	255.000000

Analisis

Secara kesuluruhan, data yang digunakan berdimensi 1059×8 dengan satu feature bertipe float, enam feature bertipe integer, dan satu feature bertipe object yang merupakan class label. Analisis deskripsi data juga ditampilkan seperti output tabel di atas yang menunjukkan nilai mean, standard deviasi, nilai minimum, nilai maksimum, dan quartile pada masing-masing feature.

c) Cleaning Data

- **Menganalisis fitur bertipe numerik dan kategorikal**

```
"""
1. Analyze the numerical and categorical features, and convert
categorical feature into numerical.
"""
data['Grade'].unique()
```

✓ 0.0s Python

```
array(['high', 'low', 'medium'], dtype=object)
```

Analisis

Pada tahap ini dilakukan pengecekan feature yang memiliki data bertipe kategorikal. Berdasarkan output diketahui bahwa feature 'Grade' merupakan data bertipe kategorikal yang memiliki tiga nilai di dalamnya, yaitu high, low, dan medium. Data bertipe kategorikal ini nantinya akan dikonversi menjadi data bertipe numerik agar dapat dibaca oleh mesin dan diolah dalam pemodelan.

- **Memeriksa missing value**

```
# 2. Check for missing values and handle them.
data.isnull().sum()
```

✓ 0.0s Python

```
pH            0
Temprature    0
Taste         0
Odor          0
Fat           0
Turbidity     0
Colour        0
Grade         0
dtype: int64
```

Analisis

Pengecekan missing value dilakukan agar tidak mempengaruhi proses analisis dan hasil akhir. Berdasarkan output di atas diketahui bahwa tidak ditemukan

missing value pada dataset yang digunakan. Sehingga tahap pengolahan data selanjutnya bisa langsung dilakukan.

d) Modelling

- **Konversi class label**

```
from sklearn import preprocessing
label_encoder = preprocessing.LabelEncoder()
data['Grade']= label_encoder.fit_transform(data['Grade'])
data['Grade'].unique()
```

✓ 0.0s Python

```
array([0, 1, 2])
```

Analisis

Label encoding dilakukan untuk mengkonversi class label yang sebelumnya bertipe kategorikal menjadi numerik. Berdasarkan output di atas maka terjadi perubahan class label seperti berikut:

- High → 0
- Low → 1
- Medium → 2

Sehingga didapatkan transform class label seperti berikut:

Grade
high
high
low
low
medium

Grade
0
0
1
1
2

- **Splitting data**

```
from sklearn.model_selection import train_test_split
X = data.drop(['Grade'],axis=1)
y = data['Grade']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.3, random_state = 3)
```

✓ 0.0s

Analisis

Digunakan metode Hold Out untuk melakukan splitting data. Data dibagi menjadi train data dan test dengan perbandingan 70:30, yang mana data akan terlebih dahulu digenarate random sebanyak 3 kali sebelum akhirnya displit menjadi train dan test data. Masing-masing train dan test data yang telah

terbentuk akan dipisahkan dengan class label mereka untuk mendapatkan class label hasil prediksi yang baru.

- **Pemodelan menggunakan decision tree**

```
from sklearn.tree import DecisionTreeClassifier

decision_tree = DecisionTreeClassifier()
decision_tree.fit(X_train, y_train)
```

✓ 0.0s Python

```
DecisionTreeClassifier()
```

```
class_result = decision_tree.predict(X_test)
class_result.shape
```

✓ 0.0s Python

```
(318,)
```

```
class_result
```

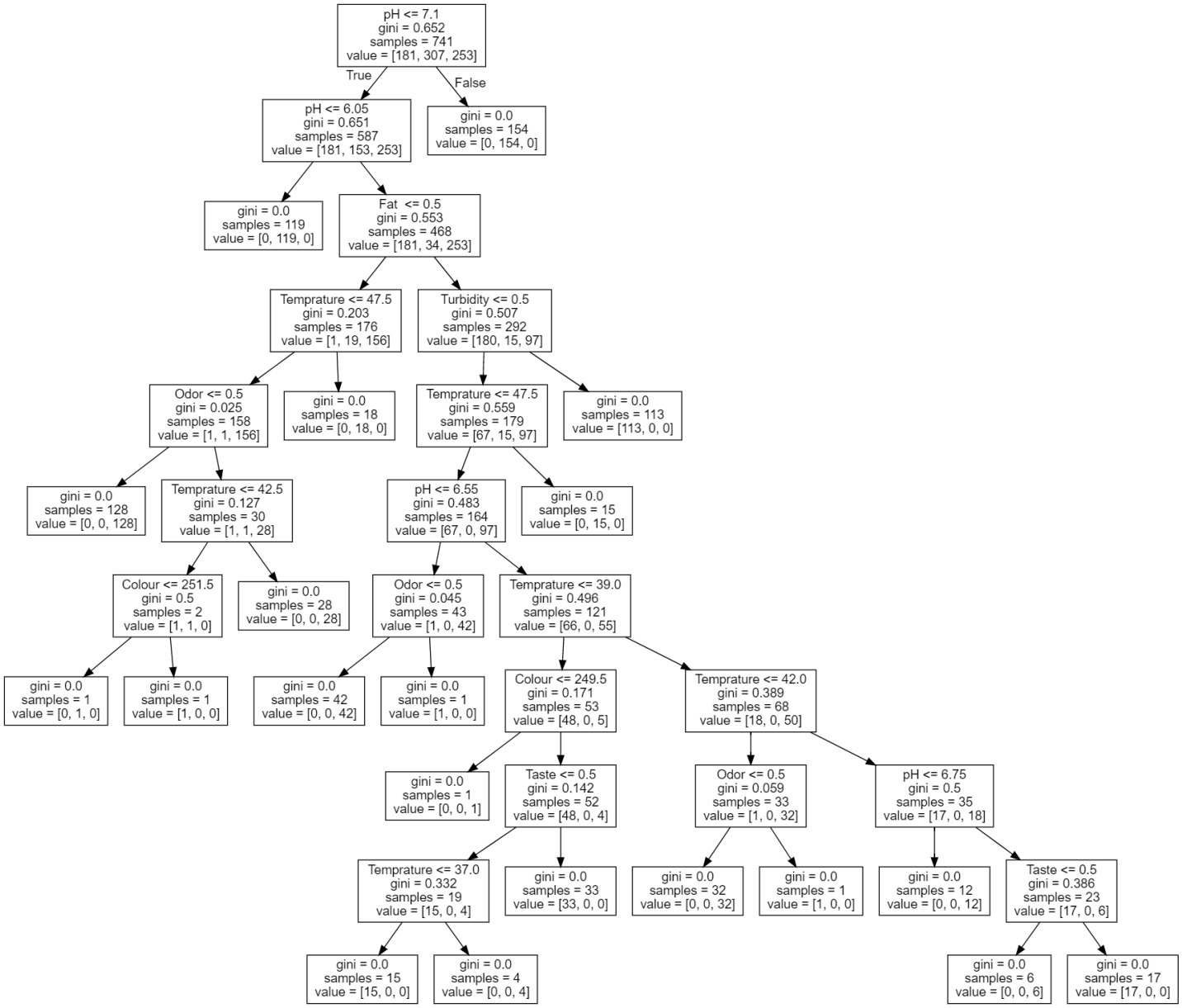
✓ 0.0s Python

```
array([1, 2, 1, 1, 1, 0, 1, 1, 2, 0, 1, 0, 1, 0, 1, 2, 0, 1, 0, 1, 2, 2,
       1, 1, 1, 1, 0, 1, 1, 1, 1, 2, 2, 0, 1, 2, 2, 1, 0, 0, 2, 2, 2, 0,
       1, 1, 2, 1, 0, 0, 1, 2, 2, 1, 2, 1, 2, 2, 1, 1, 1, 0, 2, 2, 2, 2,
       2, 0, 1, 0, 1, 2, 2, 2, 0, 1, 2, 1, 0, 2, 1, 2, 0, 1, 2, 2, 0, 0,
       0, 1, 1, 0, 0, 0, 2, 0, 1, 1, 0, 2, 1, 0, 0, 0, 0, 1, 1, 2, 1, 2,
       2, 1, 2, 1, 1, 0, 1, 0, 2, 0, 2, 0, 2, 1, 0, 1, 0, 1, 2, 0, 2, 1,
       2, 1, 0, 2, 1, 2, 2, 1, 2, 2, 1, 1, 1, 2, 1, 2, 1, 2, 1, 1, 2, 1,
       2, 0, 1, 2, 2, 0, 1, 2, 2, 2, 1, 1, 2, 1, 1, 1, 2, 2, 2, 1, 0, 0,
       2, 2, 0, 2, 0, 1, 2, 2, 1, 2, 2, 1, 1, 2, 0, 1, 1, 2, 1, 0, 2, 2,
       0, 2, 1, 1, 0, 1, 0, 1, 2, 2, 1, 2, 1, 2, 2, 2, 2, 1, 1, 1, 2, 2,
       1, 0, 2, 1, 2, 0, 0, 2, 0, 0, 2, 2, 1, 1, 1, 1, 1, 2, 1, 1, 1, 0,
       1, 2, 2, 1, 0, 2, 1, 1, 0, 2, 2, 1, 0, 1, 2, 1, 0, 2, 0, 0, 1, 0,
       2, 2, 1, 2, 1, 0, 2, 0, 2, 1, 2, 2, 1, 2, 2, 0, 1, 2, 0, 1, 1, 2,
       0, 2, 2, 0, 0, 2, 1, 2, 2, 2, 1, 0, 0, 2, 2, 1, 0, 1, 2, 1, 1, 0,
       2, 0, 2, 0, 1, 0, 2, 2, 2, 1])
```

Analisis

Digunakan bantuan library DecisionTreeClassifier dalam pemodelan. Train data yang terbagi menjadi X_train dan y_train akan dilatih menggunakan algoritma decision tree untuk mendapatkan sebuah model. Setelah berhasil, akan dilakukan tahap testing model menggunakan test data tanpa melibatkan class label dari test data. Tahap prediksi menggunakan test data akan menghasilkan class label baru yang nantinya akan dibandingkan dengan class label lama, sehingga dapat diketahui ketepatan prediksi berdasarkan model yang telah dibangun.

- **Hierarchical**



3. PERFORMA PEMODELAN

- **Confusion Matrix**

```
from sklearn.metrics import classification_report, confusion_matrix
print('-CONFUSION MATRIX-')
print(confusion_matrix(y_test, class_result))
```

✓ 0.0s Python

```
-CONFUSION MATRIX-
[[ 75   0   0]
 [  2 120   0]
 [  0   0 121]]
```

Berdasarkan output confusion matrix di atas, dapat digambarkan juga sebagai berikut:

PREDICTED CLASS \ TRUE CLASS	High	Low	Medium
High	75	0	0
Low	2	120	0
Medium	0	0	121

Dari confusion matrix tersebut dapat dilihat bahwa hasil dari class prediksi hampir memiliki keakuratan sempurna. Kesalahan terjadi pada dua prediksi kelas, data yang seharusnya memiliki class label 'High' salah diprediksi menjadi class label 'Low'. Secara detail hasil klasifikasi dapat dibagi sebagai berikut:

High Class

	(True Class) High	**(True Class) Non High**
(Predicted Class) High	75 (TP)	0 (FP)
(Predicted Class) Non High	2 (FN)	120+121 = 241 (TN)

Low Class

	(True Class) Low	**(True Class) Non Low**
(Predicted Class) Low	120 (TP)	2 (FP)
(Predicted Class) Non Low	0 (FN)	75+121 = 196 (TN)

Medium Class

	(True Class) Medium	(True Class) Non Medium
(Predicted Class) Medium	121 (TP)	2 (FP)
(Predicted Class) Non Medium	0 (FN)	75+2+120 = 197 (TN)

- **Accuracy, Precision, Recall**

```
from sklearn.metrics import accuracy_score
accuracy = accuracy_score(class_result, y_test)
accuracy
```

✓ 0.0s Python

```
0.9937106918238994
```

```
print(classification_report(y_test, class_result))
```

✓ 0.0s Python

```
              precision    recall  f1-score   support

           0       0.97      1.00      0.99        75
           1       1.00      0.98      0.99       122
           2       1.00      1.00      1.00       121

    accuracy                           0.99       318
   macro avg       0.99      0.99      0.99       318
weighted avg       0.99      0.99      0.99       318
```

Analisis

Class label hasil prediksi akan dibandingkan dengan class label asli, kemudian akan dihitung nilai akurasinya. Berdasarkan pengujian, didapatkan hasil akurasi model sebesar 99,3%. Yang mana hasil tersebut sudah menunjukkan keakuratan hasil yang tinggi pada model. Namun, kita juga perlu memperhatikan nilai uji lainnya, yaitu **Precision**.

Nilai **Precision** dipilih dengan alasan kesalahan prediksi False Positif lebih merugikan terhadap sistem. Dimana apabila kualitas susu 'Low' dan salah diprediksi menjadi 'High', maka baik customer ataupun perusahaan akan sama-

sama dirugikan. Oleh karena itu, nilai uji Precision menjadi patokan terhadap keberhasilan model.

Rata-rata nilai **Precision** pada ketiga class label bernilai 99%, yang mana ini menunjukkan hasil **pemodelan data Milk Quality menggunakan algoritma Decision Tree dapat dikatakan berhasil.** Dikarenakan nilai akurasi dan precision yang sudah tinggi, maka pemodelan terhadap data Milk Quality dirasa tidak perlu diubah dan telah cukup.

KESIMPULAN

- Digunakan data Milk Quality yang memiliki tujuh feature dan satu class label yang terdiri dari tiga kelas, yaitu High, Low, dan Medium. Berdasarkan data, diputuskan akan dilakukan pemodelan menggunakan metode klasifikasi karena yang akan diprediksi pada permasalahan kali ini adalah kelas data.
- Algoritma klasifikasi yang digunakan adalah Decision Tree, hal ini dikarenakan algoritma decision tree akan menghasilkan sebuah keputusan sebab-akibat dengan pernyataan tertentu yang dipengaruhi oleh ketujuh feature yang ada.
- Pada performa pemodelan selain nilai akurasi, nilai precision digunakan sebagai patokan keberhasilan pemodelan. Dikarenakan pada permasalahan yang diuji, kesalahan prediksi False Positif lebih merugikan sistem.
- Hasil pemodelan menggunakan algoritma decision tree memiliki nilai akurasi dan precision yang tinggi, yaitu sebesar 99%. Oleh karena itu, pemodelan terhadap data Milk Quality tidak perlu diubah lagi dan dirasa telah cukup.

REFRENSI

1. https://www.pengalaman-edukasi.com/2021/01/perbedaan-antara-algoritma-dan-model.html?m=1
2. https://ilham1012.com/2019/06/11/regresi-vs-klasifikasi-apa-bedanya/
3. https://stackabuse.com/decision-trees-in-python-with-scikit-learn/