KLASIFIKASI NAÏVE BAYES MENGGUNAKAN PYTHON

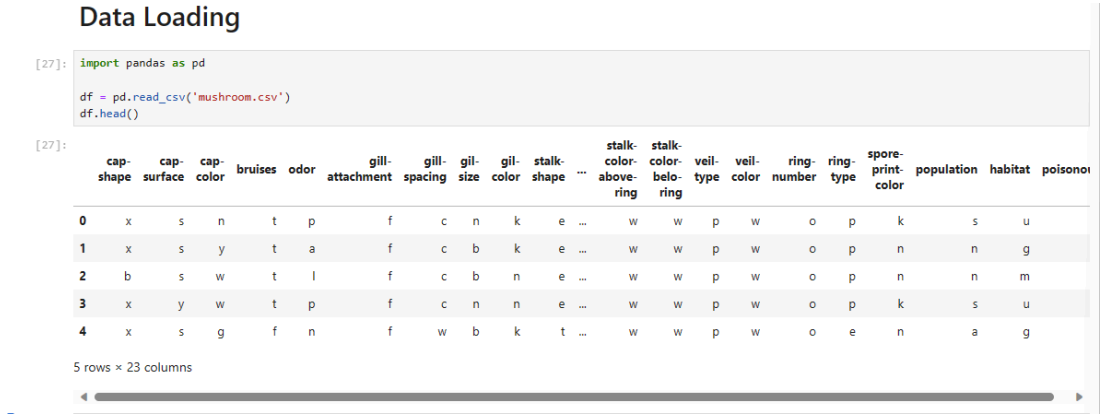
disusun untuk memenuhi tugas
mata kuliah Pembelajaran
Mesin

Oleh:

M. Syahidal Akbar Zas
2208107010045
Muhammad Raihan
2208107010021
Ammar Qurthuby
2208107010031
Azri Harniza
2208107010034

JURUSAN INFORMATIKA
FAKULTAS METEMATIKA DAN ILMU PENGETAHUAN ALAM
UNIVERSITAS SYIAH KUALA
DARUSSALAM, BANDA ACEH
2025

1. Cari dataset
   Dataset yang digunakan adalah dataset Mushroom (Jamur) yang mengklasifikasi apakah Mushroom tersebut poisonous (beracun) atau edible (layak dimakan.
   Sumber [Mushroom - UCI Machine Learning Repository]
2. Task Klasifikasi menggunakan metode Naïve Bayes
   Kali ini digunakan Bahasa pemrograman Python untuk melakukan klasifikasi data dan untuk memudahkan dalam proses tersebut digunakan Jupyter Notebook untuk mengeksekusi cell atau subprogram secara terpisah. Berikut Langkah beserta source code-nya untuk task Klasifikasi menggunakan metode Naïve Bayes menggunakan Bahasa pemrogramman python di Jupyter Notebook :
   1. Data Loading
      Pada klasifikasi kali ini, kita akan load dataset Mushroom dari UCI Machine Learning Repository menggunakan **pandas 'read_csv'** function dan **'head'** function untuk menampilkan sedikit data.



Data Loading

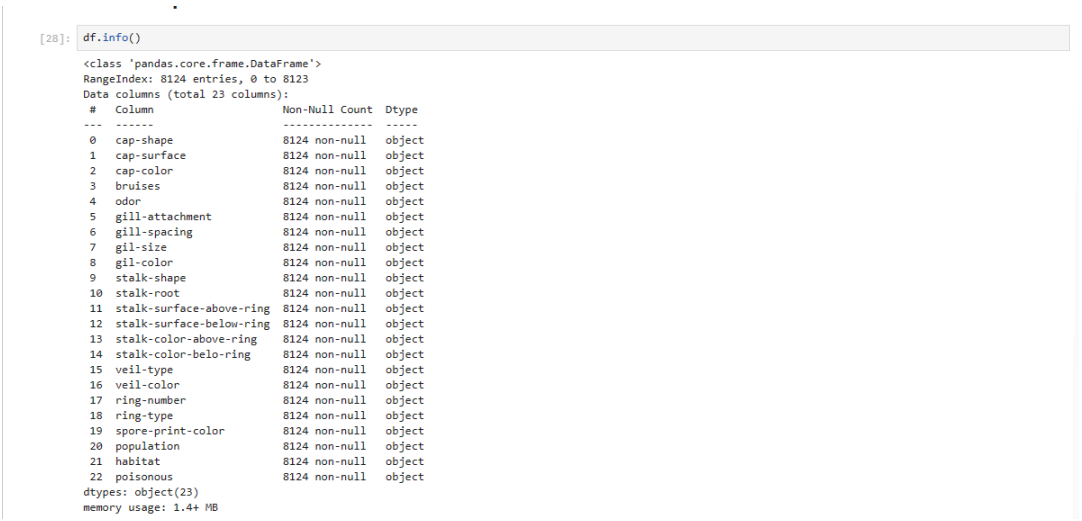
```
[27]: import pandas as pd

      df = pd.read_csv('mushroom.csv')
      df.head()
```

[27]:

	cap-shape	cap-surface	cap-color	bruises	odor	gill-attachment	gill-spacing	gil-size	gil-color	stalk-shape	...	stalk-color-above-ring	stalk-color-belo-ring	veil-type	veil-color	ring-number	ring-type	spore-print-color	population	habitat	poisonou
0	x	s	n	t	p	f	c	n	k	e	...	w	w	p	w	o	p	k	s	u	
1	x	s	y	t	a	f	c	b	k	e	...	w	w	p	w	o	p	n	n	g	
2	b	s	w	t	l	f	c	b	n	e	...	w	w	p	w	o	p	n	n	m	
3	x	y	w	t	p	f	c	n	n	e	...	w	w	p	w	o	p	k	s	u	
4	x	s	g	f	n	f	w	b	k	t	...	w	w	p	w	o	e	n	a	g	

5 rows × 23 columns

   2. Data Exploration
      Untuk melihat tipe data dan informasi lainya yang berguna untuk klasifikasi ini digunakan fungsi **'info'**

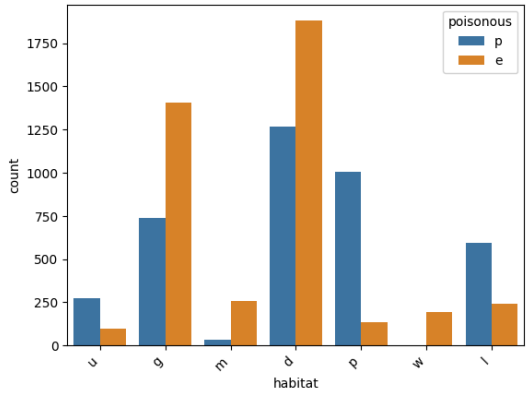


```
[28]: df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8124 entries, 0 to 8123
Data columns (total 23 columns):
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
 0   cap-shape                 8124 non-null   object
 1   cap-surface               8124 non-null   object
 2   cap-color                 8124 non-null   object
 3   bruises                   8124 non-null   object
 4   odor                      8124 non-null   object
 5   gill-attachment           8124 non-null   object
 6   gill-spacing              8124 non-null   object
 7   gil-size                  8124 non-null   object
 8   gil-color                 8124 non-null   object
 9   stalk-shape               8124 non-null   object
 10  stalk-root                8124 non-null   object
 11  stalk-surface-above-ring  8124 non-null   object
 12  stalk-surface-below-ring  8124 non-null   object
 13  stalk-color-above-ring    8124 non-null   object
 14  stalk-color-belo-ring     8124 non-null   object
 15  veil-type                 8124 non-null   object
 16  veil-color                8124 non-null   object
 17  ring-number               8124 non-null   object
 18  ring-type                 8124 non-null   object
 19  spore-print-color         8124 non-null   object
 20  population                8124 non-null   object
 21  habitat                   8124 non-null   object
 22  poisonous                 8124 non-null   object
dtypes: object(23)
memory usage: 1.4+ MB
```

Pada dataset ini akan diprediksi apakah sebuah Mushroom mengandung racun atau layak dimakan, untuk itu kita perlu contoh grafik berikut yaitu grafik poisonous terhadap habitat.

```
[29]: # Contoh grafik Label poisonous terhadap habitat
import seaborn as sns
import matplotlib.pyplot as plt

sns.countplot(data=df,x='habitat',hue=df['poisonous'])
plt.xticks(rotation=45, ha='right');
```



3. Data Procesing
   a. Split Data

      Tujuan dari split data adalah untuk memisahkan kolom class label dengan kolom-kolom lainya. Ini memudahkan untuk proses perhitungan data.

Split Data

```
[30]: X=df.iloc[:,:22].values
y=df.iloc[:, 22].values
```

```
[6]: print(X)
```

```
[['x' 's' 'n' ... 'k' 's' 'u']
 ['x' 's' 'y' ... 'n' 'n' 'g']
 ['b' 's' 'w' ... 'n' 'n' 'm']
 ...
 ['f' 's' 'n' ... 'b' 'c' 'l']
 ['k' 'y' 'n' ... 'w' 'v' 'l']
 ['x' 's' 'n' ... 'o' 'c' 'l']]
```

```
[7]: print(y)
```

```
['p' 'e' 'e' ... 'e' 'p' 'e']
```

   b. Transform TypeData Kategorical to Numerik

      Tipe data kategori di ubah menjadi numerik dengan tujuan mempermudah proses klasifikasi Naïve Bayes.

Transform typedata Kategorical to Numerik

```
[8]: from sklearn.preprocessing import LabelEncoder

# Create a LabelEncoder instance
le = LabelEncoder()

# Iterate through columns 0 to 21
for col in range(0, 22):
    X[:, col] = le.fit_transform(X[:, col])

y = le.fit_transform(y)
```

```
[71]: print(df)
```

```
     cap-shape cap-surface cap-color bruises odor gill-attachment  \
0            5           2         4       1    6               1
1            5           2         9       1    0               1
2            0           2         8       1    3               1
3            5           3         8       1    6               1
4            5           2         3       0    5               1
...        ...         ...       ...     ...  ...             ...
```

c. Checiking if the Data has a NaN Value

Agar data menjadi lebih baik untuk diproses, maka harus cek kesuluruhan data apakah mengandung NaN value atau tidak.

Checking if the Data has a NaN Value

```
[9]: print(df.isnull().sum())
```

```
cap-shape                 0
cap-surface               0
cap-color                 0
bruises                   0
odor                      0
gill-attachment           0
gill-spacing              0
gil-size                  0
gil-color                 0
stalk-shape               0
stalk-root                0
stalk-surface-above-ring  0
stalk-surface-below-ring  0
stalk-color-above-ring    0
stalk-color-belo-ring     0
veil-type                 0
veil-color                0
ring-number               0
ring-type                 0
spore-print-color         0
population                0
habitat                   0
poisonous                 0
dtype: int64
```

d. Checkin if the Data is imbalanced

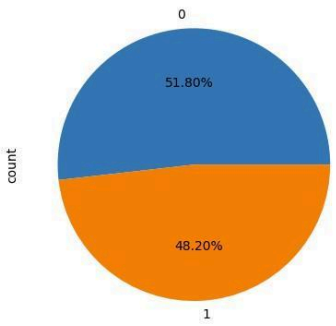
Ini bertujuan agar data yang ada mewakili keseluruhan data agar menghindari data terdistribusi skewed

Checkinf if the Data is Imbalanced

```
[10]: import numpy as np

# Assuming 'y' is a NumPy array
unique_values, counts = np.unique(y, return_counts=True)

# 'unique_values' will contain the unique values in 'y', and 'counts' will contain their corresponding counts
for value, count in zip(unique_values, counts):
    print(f"Value: {value}, Count: {count}")
```

```
Value: 0, Count: 4208
Value: 1, Count: 3916
```

```
[11]: import pandas as pd
import matplotlib.pyplot as plt

# Assuming 'y' is a NumPy array
y_series = pd.Series(y)

# Use value_counts and plot the pie chart
y_series.value_counts().plot.pie(autopct='%.2f%%')

plt.show()
```



e. Labeling the Data and Characteristics

Bertujuan untuk memisahkan data training dan data testing, dan mentransformasi isi data dalam dataset menjadi satuan yang standar atau setara.

Labeling the Data and Characteristics

```
[12]: from sklearn.model_selection import train_test_split

      X_train, X_test, y_train, y_test = train_test_split(
          X, y, test_size=0.33, random_state=125
      )
```

```
[13]: from sklearn.preprocessing import StandardScaler
      scaler = StandardScaler()
      scaler.fit(X_train)
      X_train = scaler.transform(X_train)
      X_test = scaler.transform(X_test)
```

```
[14]: X_train
```

```
[14]: array([[-0.83427256, -1.50847428,  1.75382193, ..., -1.08419784,
              1.07753997, -0.87289376],
             [ 1.0418928 ,  0.12824963,  1.36342226, ..., -0.66405767,
              -1.29729472,  0.87793319],
             [ 1.0418928 ,  0.94661159, -0.97897577, ..., -0.2439175 ,
              1.07753997, -0.87289376],
             ...,
             [ 1.0418928 ,  0.94661159, -0.19817643, ...,  1.43664319,
              0.28592841,  0.29432421],
             [ 1.0418928 , -1.50847428, -0.97897577, ..., -0.2439175 ,
              0.28592841, -0.87289376],
             [-0.83427256, -1.50847428, -0.5885761 , ..., -0.2439175 ,
              1.07753997, -0.87289376]])
```

```
[15]: X_test
```

```
[15]: array([[ 1.0418928 , -1.50847428,  1.36342226, ..., -0.2439175 ,
              -2.88051785, -0.28928478],
             [-0.83427256,  0.12824963,  1.36342226, ..., -1.08419784,
              -0.50568315, -0.28928478],
             [ 1.0418928 , -1.50847428, -1.36937544, ...,  1.43664319,
              0.28592841,  0.29432421],
             ...,
             [-0.83427256,  0.94661159, -0.19817643, ...,  1.43664319,
              0.28592841,  1.46154217],
             [-2.08504947, -1.50847428,  1.36342226, ...,  1.43664319,
              -0.50568315, -0.28928478],
             [ 1.0418928 ,  0.12824963,  1.36342226, ...,  1.43664319,
              -1.29729472, -0.28928478]])
```

4. Model Building and Training

   Pada model Building and Training dataset kali ini adalah menggunakan klasifikasi Naïve Bayes. Langkah-langkahnya adalah sebagai berikut:

   a. Model Building and Training

      Membuat model baru pada data training.

Model Building and Training

```
[16]: from sklearn.naive_bayes import GaussianNB

      model = GaussianNB()

      model.fit(X_train, y_train);
```

   b. Model Evaluation

      Berdasarkan model yang telah di Building dan Training sebelumnya, selanjutnya model akan di Evaluation untuk mennetukan Accuracy dan F1 Score.

Model Evaluation

```
[17]: from sklearn.metrics import (
          accuracy_score,
          confusion_matrix,
          ConfusionMatrixDisplay,
          f1_score,
          classification_report,
      )

      y_pred = model.predict(X_test)

      accuray = accuracy_score(y_pred, y_test)
      f1 = f1_score(y_pred, y_test, average="weighted")

      print("Accuracy:", accuray)
      print("F1 Score:", f1)
```

```
Accuracy: 0.9198060425214473
F1 Score: 0.9197574782178044
```

5. Confusion Matrix

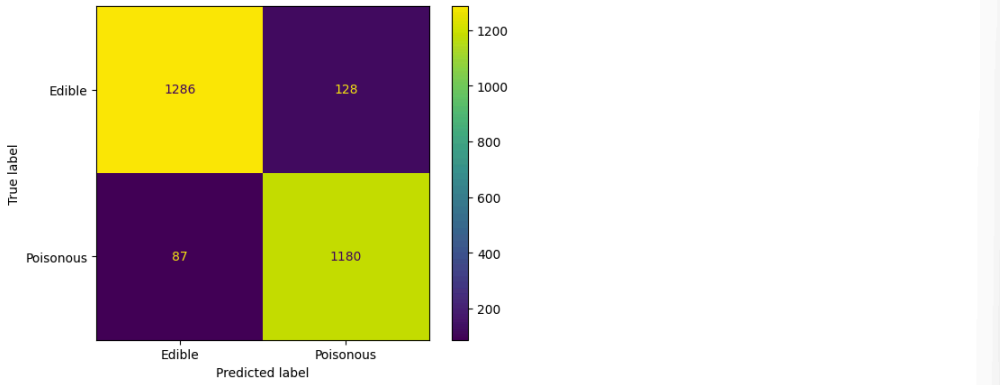
   Pada proses kali ini akan menampilkan Confusion Matrix dari data Mushroom dan menampilkan recall, precision, accuracy dan f1-score dari data.

Confusion Matrix

```
[18]: import seaborn as sn
      import matplotlib.pyplot as plt
      from sklearn.metrics import confusion_matrix

[19]: cm_nb = confusion_matrix(y_test,y_pred)

[20]: cm_nb

[20]: array([[1286,  128],
             [  87, 1180]], dtype=int64)

[21]: tn, fp, fn, tp = cm_nb.ravel()

      recall = tp / (tp + fn)
      precision = tp / (tp + fp)
      accuracy = (tp + tn) / (tn + fp + fn + tp)
      f1 = 2 * (precision * recall) / (precision + recall)

[22]: print("recall: ", recall)
      print("precision: ", precision)
      print("accuracy: ", accuracy)
      print("f1-score: ", f1)

      recall:  0.9313338595106551
      precision:  0.9021406727828746
      accuracy:  0.9198060425214473
      f1-score:  0.9165048543689321
```

Berikut adalah gambar dari Confusion Matrix :

```
[23]: labels = ["Edible", "Poisonous"]
      cm = confusion_matrix(y_test, y_pred)
      disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=labels)
      disp.plot();
```



Interpretasi:

Berdasarkan hasil evaluasi model yang ditunjukkan oleh confusion matrix, model memiliki performa yang cukup baik dalam mengklasifikasikan jamur sebagai *Edible* atau *Poisonous*. Dari 2569 sampel data uji, model berhasil mengklasifikasikan 1286 sampel *Edible* dengan benar (True Negative) dan 1180 sampel *Poisonous* dengan benar (True Positive). Namun, terdapat 87 kasus False Positive, di mana jamur yang sebenarnya *Poisonous* salah diklasifikasikan sebagai *Edible*. Kesalahan ini dapat berbahaya dalam konteks nyata karena dapat menyebabkan konsumsi jamur beracun. Selain itu, terdapat 126 kasus False Negative, di mana jamur yang sebenarnya *Edible* salah diklasifikasikan sebagai *Poisonous*, yang dapat menyebabkan kesalahan dalam identifikasi dan pemborosan sumber daya.

Dari hasil perhitungan metrik evaluasi, model memiliki nilai **recall** sebesar 93.13%, yang menunjukkan bahwa model mampu mengidentifikasi sebagian besar jamur *Poisonous* dengan benar. **Precision** sebesar 93.10% menunjukkan bahwa dari semua sampel yang diprediksi sebagai *Poisonous*, sebanyak 93.10% benar-benar *Poisonous*. **Akurasi** model mencapai 91.00%, yang berarti model secara keseluruhan mampu mengklasifikasikan sampel dengan benar dalam 91% dari semua kasus. Sementara itu, **F1-score** sebesar 91.60% menunjukkan keseimbangan yang baik antara precision dan recall.

Link Penjelasan Video: https://youtu.be/PoMysNaaZS8
Link GitHub: twosecondz/Kelompok_1_Tugas01_Data_Preparation: Tugas 1 MK Machine Learning B