BAB IV

HASIL DAN PEMBAHASAN

4.1 Hasil Penelitian

Pada penelitian ini menggunakan algoritma *Naive Bayes* untuk mengklasifikasikan tingkat kerontokan rambut yang terdiri dari 9 fitur seperti begadang, tingkat tekanan, konsumsi kopi, durasi kerja otak, tingkat stres, berenang, minyak rambut, ketombe, libido (hormon) dan 4 label yang terdiri dari few, medium, many, a lot. Dataset yang digunakan terdiri dari 400 data sampel yang telah melalui proses prapemrosesan seperti pembersihan data, pembobotan fitur dan diimplementasikan pada *tools Google Collabs* sehingga memperoleh tingkat akurasi 76% dengan rasio 70% data training dan 30% data testing.

4.2 Pembahasan

Implementasi dengan *Tools Google Collabs.*

IMPORT LIBRARY

```
[130] import pandas as pd
      from sklearn.naive_bayes import GaussianNB
      from sklearn import metrics
      from sklearn.metrics import classification_report
      from sklearn.metrics import ConfusionMatrixDisplay
      from sklearn.model_selection import train_test_split
```

Gambar 4.1 Import Library

MENAMPILKAN DATASET

```
[131] df_milk = pd.read_excel('hair_loss_dataset3.xlsx')
      df_milk
```

	stay_up_late	pressure_level	coffee_consumed	brain_working_duration	stress_level	swimming	hair_grease	dandruff	libido	hair_loss
0	2	0	0	1	0	0	3	0	1	Few
1	0	0	0	3	0	0	1	0	1	Few
2	3	0	1	0	0	1	2	0	2	Medium
3	2	0	0	1	0	0	3	0	3	Few
4	2	0	0	1	0	0	1	0	2	Few
...	...	...	...	...	...	...	...	...	...	...
395	1	0	1	2	0	0	1	0	5	Medium
396	1	0	0	3	0	1	2	0	1	Few
397	1	0	1	1	0	0	2	0	5	Medium
398	0	0	1	1	0	0	2	0	5	Medium
399	1	0	0	2	0	1	2	0	1	Few

400 rows × 10 columns

Gambar 4.2 Menampilkan Dataset

```
df_milk.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 400 entries, 0 to 399
Data columns (total 10 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   stay_up_late            400 non-null    int64
 1   pressure_level          400 non-null    int64
 2   coffee_consumed         400 non-null    int64
 3   brain_working_duration  400 non-null    int64
 4   stress_level            400 non-null    int64
 5   swimming                400 non-null    int64
 6   hair_grease             400 non-null    int64
 7   dandruff                400 non-null    int64
 8   libido                  400 non-null    int64
 9   hair_loss               400 non-null    object
dtypes: int64(9), object(1)
memory usage: 31.4+ KB
```

Gambar 4.3 Manampilkan info dataset

MENAMPILKAN PROBABILITAS DATA CLASS/LABEL

```
df_milk['hair_loss'].value_counts()
```

hair_loss	count
Few	169
Medium	167
Many	42
A lot	22

dtype: int64

Gambar 4.4 Mencari Probabilitas Data Class/Label

Featur Scaling dilakukan jika jarak antara value-value pada setiap fitur terlihat berjauhan, maka dilakukan penskalaan agar terlihat tidak jauh jarak antar nilainya.

FEATUR SCALING (JIKA DIPERLUKAN)

```
[134] from sklearn.preprocessing import StandardScaler

sc = StandardScaler()
x_train = sc.fit_transform(x_train)
x_test = sc.fit_transform(x_test)

[136] print(x_train)
```

```
[[ 0.79322978 -0.56635211 -0.02157282 ... -1.21898252 -0.53073452
  -1.01980776]
 [ 0.79322978 -0.56635211 -0.57069905 ...  0.42195549 -0.53073452
  -1.01980776]
 [ 0.13220496 -0.56635211 -0.57069905 ... -0.39851352 -0.53073452
  -1.01980776]
 ...
 [ 0.79322978 -0.56635211 -0.57069905 ...  0.42195549 -0.53073452
  -0.46962246]
 [-1.18984467 -0.56635211 -0.57069905 ... -0.39851352 -0.53073452
  -0.46962246]
 [ 4.09835386  2.93170506  4.92056329 ...  2.0628935   2.88548769
  -1.56999307]]
```

Gambar 4.5 Featur Scaling

Memisahkan variable Independent yang terdiri dari 9 fitur dan variable dependent yang terdiri dari 4 Label untuk diproses.

MEMISAHKAN VARIABEL INDEPENDENT DAN DEPENDENT (LABEL)

```
[137] x = df_milk.iloc[:,:9].values
      y = df_milk.iloc[:,9].values
```

Gambar 4.6 Memisahkan variable Independent dan dependent

OUTPUT INDEFENDENT VARIABEL

```
[138] x
```

```
array([[2, 0, 0, ..., 3, 0, 1],
       [0, 0, 0, ..., 1, 0, 1],
       [3, 0, 1, ..., 2, 0, 2],
       ...,
       [1, 0, 1, ..., 2, 0, 5],
       [0, 0, 1, ..., 2, 0, 5],
       [1, 0, 0, ..., 2, 0, 1]])
```

Gambar 4.7 Output Variable Independent

OUTPUT DEFENDENT VARIABEL

```
y
```

```
array(['Few', 'Few', 'Medium', 'Few', 'Few', 'Few', 'Medium', 'Few',
       'Few', 'Medium', 'Medium', 'Medium', 'Medium', 'Medium', 'Medium',
       'Few', 'Few', 'Few', 'Few', 'Few', 'Few', 'Few', 'Few', 'Few',
       'Few', 'Few', 'Medium', 'Medium', 'Medium', 'Medium', 'Medium',
       'Few', 'Few', 'Medium', 'Medium', 'Few', 'Few', 'Few', 'Medium',
       'Medium', 'Medium', 'Medium', 'Medium', 'Few', 'Medium', 'Medium',
       'Medium', 'Few', 'Few', 'Few', 'Few', 'Few', 'Medium', 'Medium',
       'Medium', 'Medium', 'Medium', 'Few', 'Few', 'Few', 'Few', 'Few',
       'Medium', 'Few', 'Medium', 'Few', 'Few', 'Few', 'Medium', 'Medium',
       'Medium', 'Medium', 'Few', 'Few', 'Few', 'Few', 'Few', 'Medium',
       'Few', 'Few', 'Medium', 'Medium', 'Medium', 'Medium', 'Few',
       'Medium', 'Medium', 'Medium', 'Few', 'Few', 'Few', 'Few', 'Medium',
       'Medium', 'Medium', 'Medium', 'Medium', 'Medium', 'Many', 'Many',
       'Many', 'Many', 'Many', 'Many', 'Many', 'Many', 'A lot', 'A lot',
       'A lot', 'A lot', 'A lot', 'A lot', 'A lot', 'Many', 'Medium',
       'Medium', 'Medium', 'Many', 'Medium', 'Medium', 'Many', 'Many',
       'Many', 'Many', 'Many', 'Medium', 'Medium', 'Medium', 'Medium',
       'Medium', 'Few', 'Few', 'Few', 'Few', 'Medium', 'Few', 'Few',
       'Few', 'Few', 'Few', 'Medium', 'Few', 'Few', 'Few', 'Medium',
       'Medium', 'Medium', 'Few', 'Few', 'Few', 'Few', 'Few', 'Few',
       'Medium', 'Few', 'Few', 'Medium', 'Few', 'Few', 'Few', 'Few',
       'Medium', 'Medium', 'Medium', 'Medium', 'Medium', 'Few', 'Few',
```

Gambar 4.8 Output Variabel Dependent

SPLIT DATA TRANING DAN TESTING

```
x_train, x_test, y_train, y_test = train_test_split(x,y,test_size=0.3,random_state=4)
```

Gambar 4.9 membagi data traning dan testing

NAIVE BAYES

```
nb_clas = GaussianNB()
nb_clas.fit(x_train,y_train)
```

```
GaussianNB
GaussianNB()
```

Gambar 4.10 membuat model / rumus naïve bayes

BUAT VARIABEL PREDIKSI DARI DATA TESTING (RANDOM)

```
y_predict = nb_clas.predict(x_test)
y_predict
```

```
array(['Medium', 'Medium', 'Few', 'Many', 'Many', 'Medium', 'Few',
       'A lot', 'Many', 'Many', 'Few', 'Few', 'Medium', 'Many', 'A lot',
       'Few', 'Medium', 'A lot', 'Medium', 'Few', 'Few', 'Few', 'Medium',
       'Medium', 'Few', 'Few', 'Medium', 'Many', 'Few', 'Few', 'Medium',
       'Few', 'Few', 'Many', 'Many', 'Many', 'Few', 'Medium', 'Medium',
       'Medium', 'Many', 'Few', 'Medium', 'Few', 'Few', 'Few', 'Medium',
       'Medium', 'Few', 'Medium', 'A lot', 'Few', 'Few', 'Few', 'Medium', 'Few',
       'Many', 'Medium', 'Many', 'Many', 'Few', 'Few', 'Medium', 'Medium',
       'Many', 'Medium', 'Few', 'Medium', 'Few', 'Medium', 'Few', 'Few',
       'Few', 'Medium', 'A lot', 'A lot', 'Many', 'Few', 'Many', 'Many',
       'Medium', 'Medium', 'Medium', 'Few', 'Few', 'A lot', 'Few', 'Few',
       'Few', 'Many', 'Many', 'Many', 'Few', 'Few', 'Many', 'A lot',
       'Medium', 'Few', 'A lot', 'A lot', 'Few', 'Few', 'Few', 'Medium',
       'Many', 'A lot', 'Few', 'A lot', 'Few', 'Many', 'A lot', 'Few',
       'Medium', 'Few', 'Medium', 'Medium', 'Many', 'Medium', 'Few',
       'Medium', 'Medium'], dtype='<U6')
```

Gambar. 4.11 membuat variabel testing (random)

Dari hasil evaluasi dibawah ini mendapatkan tingkat akurasi sebesar 76% dengan data testing 30% yaitu 120 data, terbagi menjadi 7 data a lot, 50 data few, 15 data many dan 48 data medium.

EVALUASI

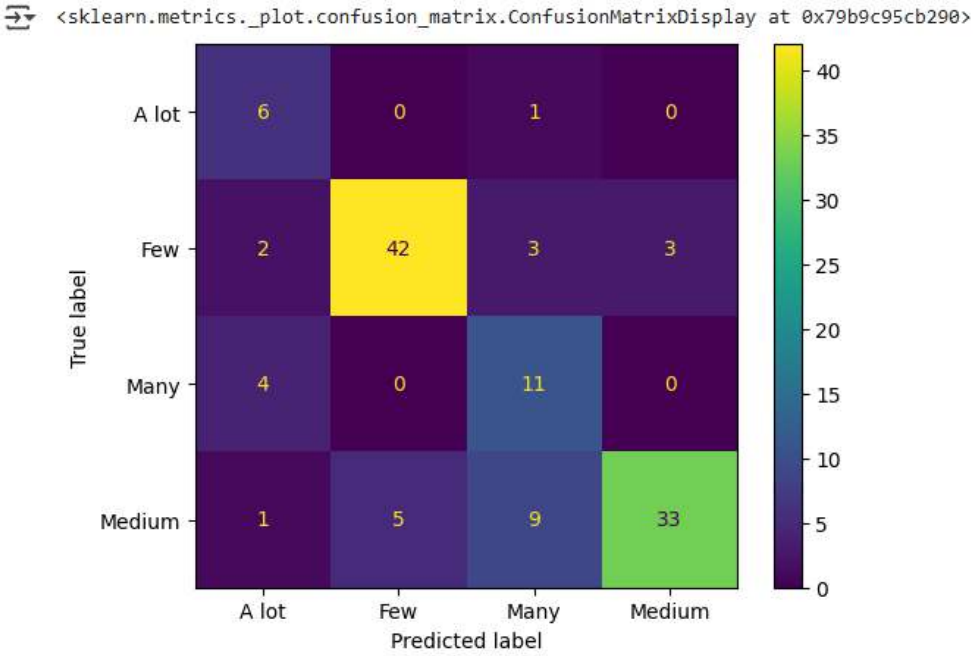
```
print("Test Accuracy : ", metrics.accuracy_score(y_test,y_predict))
print(metrics.classification_report(y_test,y_predict))
ConfusionMatrixDisplay.from_predictions(y_test,y_predict)
```

```
Test Accuracy :  0.7666666666666667
              precision    recall  f1-score   support

       A lot       0.46      0.86      0.60         7
         Few       0.89      0.84      0.87        50
        Many       0.46      0.73      0.56        15
      Medium       0.92      0.69      0.79        48

    accuracy                           0.77       120
   macro avg       0.68      0.78      0.70       120
weighted avg       0.82      0.77      0.78       120
```

Gambar 4.12 Evaluasi

Hasil pada diagram metric dibawah ini menunjukan bahwa:

a. dari 7 data a lot memperoleh prediksi 6 data sesuai dan 1 data masuk pada label many.
b. pada label few terdapat 50 data yang sesuai sebanyak 42 data dan 2 data masuk label a lot, 3 data masuk label many dan 3 data masuk label medium.
c. pada label many terdapat 15 data yang sesuai sebanyak 11 data dan 4 data masuk label a lot
d. pada label medium terdapat 48 data, yang sesuai sebanyak 33 data, 1 data masuk label a lot, 5 data masuk label few dan 9 data masuk label many.



Gambar 4.13 Diagram Metrik

INPUT DATA BARU

```
[144] new_data = [[3,0,1,0,0,0,2,0,4]]
      new_data
```

```
[[3, 0, 1, 0, 0, 0, 2, 0, 4]]
```

```
[145] predictions = nb_clas.predict(new_data)
      print("prdick result:",predictions)
```

```
prdick result: ['Medium']
```

```
[146] new_data = [[8,3,8,18,3,0,5,2,4]]
      new_data
```

```
[[8, 3, 8, 18, 3, 0, 5, 2, 4]]
```

```
[147] predictions = nb_clas.predict(new_data)
      print("predick result :",predictions)
```

```
predick result : ['A lot']
```

Gambar 4.14 Input contoh data baru

Pada metode Naive Bayes, sebenarnya model tidak secara eksplisit memberikan "feature importance" seperti pada Decision Tree atau Random Forest. Namun, kita masih bisa mencari atribut/variabel yang paling berperan dalam proses klasifikasi dengan beberapa cara:

1. **Melihat *Conditional Probability* (P(x|y))**

- Naive Bayes menghitung peluang tiap atribut terhadap kelas.
- Variabel yang punya distribusi probabilitas paling berbeda antar kelas biasanya paling berpengaruh.
- Misalnya, jika pada atribut Jenis Rambut:
  - P(Rambut Rontok | Many) = 0.8
  - P(Tidak Rontok | Few) = 0.2

    maka atribut ini sangat berperan karena perbedaannya jelas.

2. Menggunakan *Mutual Information* (MI)

- MI mengukur seberapa besar informasi sebuah variabel memberi tahu tentang kelas**.**
- Bisa dihitung dengan mutual_info_classif dari sklearn.feature_selection.
- Variabel dengan nilai MI tinggi → paling berkontribusi.

```
from sklearn.feature_selection import mutual_info_classif

mi = mutual_info_classif(X, y)
for col, score in zip(X.columns, mi):
    print(col, ":", score)
```

Gambar 4.15 Mutual Information

3. Menggunakan *Feature Selection*

- Terapkan metode seleksi fitur seperti:
    - **Chi-Square Test** → untuk data kategorikal.
    - **ANOVA F-test** → untuk data numerik.
- Fitur dengan nilai statistik tertinggi berarti paling berperan.

```
from sklearn.feature_selection import chi2
import pandas as pd

chi_scores = chi2(X, y)
scores = pd.Series(chi_scores[0], index=X.columns)
print(scores.sort_values(ascending=False))
```

Gambar 4.16 Feature Selection Chi-Square

4. Melihat *Likelihood Ratio* antar kelas

- Hitung perbandingan probabilitas tiap atribut antar kelas.
- Semakin besar perbedaannya, semakin kuat peran atribut itu.