naive_bayes

October 18, 2024

0.1 1. Mengimpor Pustaka yang Diperlukan

[1]:

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import seaborn as sns
```

0.2 2. Mengimpor Dataset

[2]:

```
dataset = pd.read_csv('Social_Network_Ads.csv')
X = dataset.iloc[:, [2, 3]].values
y = dataset.iloc[:, -1].values
```

[3]:

```
print(dataset.head())
```

```
    User ID  Gender  Age  EstimatedSalary  Purchased
0  15624510    Male   19            19000          0
1  15810944    Male   35            20000          0
2  15668575  Female   26            43000          0
3  15603246  Female   27            57000          0
4  15804002    Male   19            76000          0
```

[4]:

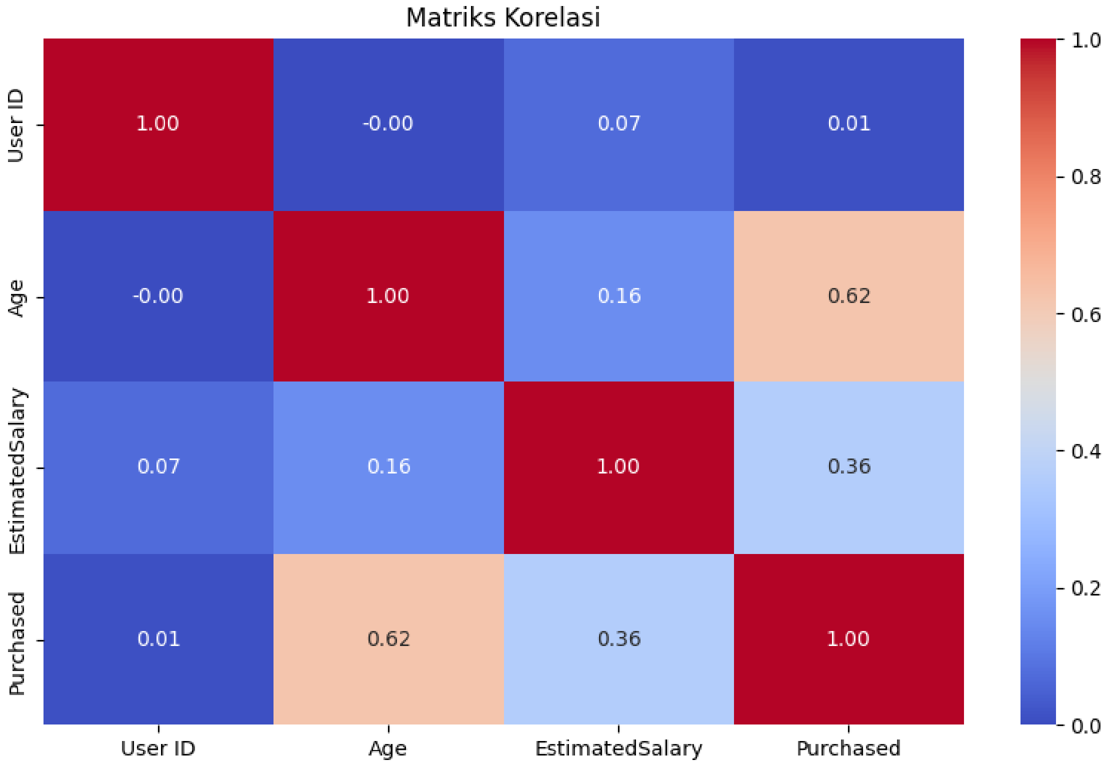
```
print(dataset.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 400 entries, 0 to 399
Data columns (total 5 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   User ID          400 non-null    int64
 1   Gender           400 non-null    object
 2   Age              400 non-null    int64
 3   EstimatedSalary  400 non-null    int64
 4   Purchased        400 non-null    int64
dtypes: int64(4), object(1)
memory usage: 15.8+ KB
None
```

[5]:
```
# periksa nilai yang hilang di setiap kolom
print(dataset.isnull().sum())
```

```
User ID            0
Gender             0
Age                0
EstimatedSalary    0
Purchased          0
dtype: int64
```

[6]:
```
# Memilih hanya kolom numerik
numerical_data = dataset.select_dtypes(include=['int64', 'float64'])

# Menghitung korelasi
correlation_matrix = numerical_data.corr()

# Visualisasi korelasi
plt.figure(figsize=(10, 6))
sns.heatmap(correlation_matrix, annot=True, fmt=".2f", cmap='coolwarm')
plt.title('Matriks Korelasi')
plt.show()
```



0.3 3. Pembagian Dataset

[7]:

```
# Membagi dataset menjadi data pelatihan dan data pengujian
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.25,␣
 ↪random_state = 0)
```

[8]:

```
# Menampilkan nilai X_train dan y_train
print("\nFitur Pelatihan (X_train):")
print(X_train)
print("\nTarget Pelatihan (y_train):")
print(y_train)
```

```
Fitur Pelatihan (X_train):
[[    44  39000]
 [    32 120000]
 [    38  50000]
 [    32 135000]
 [    52  21000]
 [    53 104000]
 [    39  42000]
 [    38  61000]
 [    36  50000]
 [    36  63000]
 [    35  25000]
 [    35  50000]
 [    42  73000]
 [    47  49000]
 [    59  29000]
 [    49  65000]
 [    45 131000]
 [    31  89000]
 [    46  82000]
 [    47  51000]
 [    26  15000]
 [    60 102000]
 [    38 112000]
 [    40 107000]
 [    42  53000]
 [    35  59000]
 [    48  41000]
 [    48 134000]
 [    38 113000]
 [    29 148000]
 [    26  15000]
 [    60  42000]
 [    24  19000]
 [    42 149000]
 [    46  96000]
 [    28  59000]
 [    39  96000]
 [    28  89000]
 [    41  72000]
 [    45  26000]
 [    33  69000]
 [    20  82000]
 [    31  74000]
 [    42  80000]
 [    35  72000]
 [    33 149000]
 [    40  71000]
 [    51 146000]
 [    46  79000]
 [    35  75000]
 [    38  51000]
 [    36  75000]
 [    37  78000]
 [    38  61000]
 [    60 108000]
 [    20  82000]
 [    57  74000]
 [    42  65000]
 [    26  80000]
 [    46 117000]
 [    35  61000]
 [    21  68000]
 [    28  44000]
 [    41  87000]
 [    37  33000]
 [    27  90000]
 [    39  42000]
 [    28 123000]
 [    31 118000]
 [    25  87000]
 [    35  71000]
 [    37  70000]
 [    35  39000]
 [    47  23000]
 [    35 147000]
 [    48 138000]
 [    26  86000]
 [    25  79000]
 [    52 138000]
 [    51  23000]
 [    35  60000]
 [    33 113000]
 [    30 107000]
 [    48  33000]
 [    41  80000]
 [    48  96000]
 [    31  18000]
 [    31  71000]
 [    43 129000]
 [    59  76000]
 [    18  44000]
 [    36 118000]
 [    42  90000]
 [    47  30000]
 [    26  43000]
 [    40  78000]
 [    46  59000]
 [    59  42000]
 [    46  74000]
 [    35  91000]
 [    28  59000]
 [    40  57000]
 [    59 143000]
 [    57  26000]
 [    52  38000]
 [    47 113000]
 [    53 143000]
 [    35  27000]
 [    58 101000]
 [    45  45000]
 [    23  82000]
 [    46  23000]
 [    42  65000]
 [    28  84000]
 [    38  59000]
 [    26  84000]
 [    29  28000]
 [    37  71000]
 [    22  55000]
 [    48  35000]
 [    49  28000]
 [    38  65000]
 [    27  17000]
 [    46  28000]
 [    48 141000]
 [    26  17000]
 [    35  97000]
 [    39  59000]
 [    24  27000]
 [    32  18000]
 [    46  88000]
 [    35  58000]
 [    56  60000]
 [    47  34000]
 [    40  72000]
 [    32 100000]
 [    19  21000]
 [    25  90000]
 [    35  88000]
 [    28  32000]
 [    50  20000]
 [    40  59000]
 [    50  44000]
 [    35  72000]
 [    40 142000]
 [    46  32000]
 [    39  71000]
 [    20  74000]
 [    29  75000]
 [    31  76000]
 [    47  25000]
 [    40  61000]
 [    34 112000]
 [    38  80000]
 [    42  75000]
 [    47  47000]
 [    39  75000]
 [    19  25000]
 [    37  80000]
 [    36  60000]
 [    41  52000]
 [    36 125000]
 [    48  29000]
 [    36 126000]
 [    51 134000]
 [    27  57000]
 [    38  71000]
 [    39  61000]
 [    22  27000]
 [    33  60000]
 [    48  74000]
 [    58  23000]
 [    53  72000]
 [    32 117000]
 [    54  70000]
 [    30  80000]
 [    58  95000]
 [    26  52000]
 [    45  79000]
 [    24  55000]
 [    40  75000]
 [    33  28000]
 [    44 139000]
 [    22  18000]
 [    33  51000]
 [    43 133000]
 [    24  32000]
 [    46  22000]
 [    35  55000]
 [    54 104000]
 [    48 119000]
 [    35  53000]
 [    37 144000]
 [    23  66000]
 [    37 137000]
 [    31  58000]
 [    33  41000]
 [    45  22000]
 [    30  15000]
 [    19  19000]
 [    49  74000]
 [    39 122000]
 [    35  73000]
 [    39  71000]
 [    24  23000]
 [    41  72000]
 [    29  83000]
 [    54  26000]
 [    35  44000]
 [    37  75000]
 [    29  47000]
 [    31  68000]
 [    42  54000]
 [    30 135000]
 [    52 114000]
 [    50  36000]
 [    56 133000]
 [    29  61000]
 [    30  89000]
 [    26  16000]
 [    33  31000]
 [    41  72000]
 [    36  33000]
 [    55 125000]
 [    48 131000]
 [    41  71000]
 [    30  62000]
 [    37  72000]
 [    41  63000]
 [    58  47000]
 [    30 116000]
 [    20  49000]
 [    37  74000]
 [    41  59000]
 [    49  89000]
 [    28  79000]
 [    53  82000]
 [    40  57000]
 [    60  34000]
 [    35 108000]
 [    21  72000]
 [    38  71000]
 [    39 106000]
 [    37  57000]
 [    26  72000]
 [    35  23000]
 [    54 108000]
 [    30  17000]
 [    39 134000]
 [    29  43000]
 [    33  43000]
 [    35  38000]
 [    41  45000]
 [    41  72000]
 [    39 134000]
 [    27 137000]
 [    21  16000]
 [    26  32000]
 [    31  66000]
 [    39  73000]
 [    41  79000]
 [    47  50000]
 [    41  30000]
 [    37  93000]
 [    60  46000]
 [    25  22000]
 [    28  37000]
 [    38  55000]
 [    36  54000]
 [    20  36000]
 [    56 104000]
 [    40  57000]
 [    42 108000]
```

```
 [    20  23000]
 [    40  65000]
 [    47  20000]
 [    18  86000]
 [    35  79000]
 [    57  33000]
 [    34  72000]
 [    49  39000]
 [    27  31000]
 [    19  70000]
 [    39  79000]
 [    26  81000]
 [    25  80000]
 [    28  85000]
 [    55  39000]
 [    50  88000]
 [    49  88000]
 [    52 150000]
 [    35  65000]
 [    42  54000]
 [    34  43000]
 [    37  52000]
 [    48  30000]
 [    29  43000]
 [    36  52000]
 [    27  54000]
 [    26 118000]]

Target Pelatihan (y_train):
[0 1 0 1 1 1 0 0 0 0 0 0 1 1 1 0 1 0 0 1 0 1 0 1 0 0 1 1 1 1 0 1 0 1 0 0 1
 0 0 1 0 0 0 0 0 1 1 1 1 0 0 0 1 0 1 0 1 0 0 1 0 0 0 1 0 0 0 1 1 0 0 1 0 1
 1 1 0 0 1 1 0 0 1 1 0 1 0 0 1 1 0 1 1 1 0 0 0 0 0 1 0 0 1 1 1 1 1 0 1 1 0
 1 0 0 0 0 0 0 0 1 1 0 0 1 0 0 1 0 0 0 1 0 1 1 0 1 0 0 0 0 1 0 0 0 1 1 0 0
 0 0 1 0 1 0 0 0 1 0 0 0 0 1 1 1 0 0 0 0 0 0 1 1 1 1 1 0 1 0 0 0 0 0 1 0 0
 0 0 0 0 1 1 0 1 0 1 0 0 1 0 0 0 1 0 0 0 0 0 1 0 0 0 0 0 1 0 1 1 0 0 0 0 0
 0 1 1 0 0 0 0 1 0 0 0 0 1 0 1 0 1 0 0 0 1 0 0 0 1 0 1 0 0 0 0 0 1 1 0 0 0
 0 0 1 0 1 1 0 0 0 0 0 1 0 1 0 0 1 0 0 1 0 1 0 0 0 0 0 0 1 1 1 1 0 0 0 0 1
 0 0 0 0]
```

[9]:

```
# Menampilkan nilai X_test dan y_test
print("\nFitur Pengujian (X_test):")
print(X_test)
print("\nTarget Pengujian (y_test):")
print(y_test)
```

```
Fitur Pengujian (X_test):
[[    30  87000]
```

```
[    38  50000]
[    35  75000]
[    30  79000]
[    35  50000]
[    27  20000]
[    31  15000]
[    36 144000]
[    18  68000]
[    47  43000]
[    30  49000]
[    28  55000]
[    37  55000]
[    39  77000]
[    20  86000]
[    32 117000]
[    37  77000]
[    19  85000]
[    55 130000]
[    35  22000]
[    35  47000]
[    47 144000]
[    41  51000]
[    47 105000]
[    23  28000]
[    49 141000]
[    28  87000]
[    29  80000]
[    37  62000]
[    32  86000]
[    21  88000]
[    37  79000]
[    57  60000]
[    37  53000]
[    24  58000]
[    18  52000]
[    22  81000]
[    34  43000]
[    31  34000]
[    49  36000]
[    27  88000]
[    41  52000]
[    27  84000]
[    35  20000]
[    43 112000]
[    27  58000]
[    37  80000]
[    52  90000]
[    26  30000]
[    49  86000]
[    57 122000]
[    34  25000]
[    35  57000]
[    34 115000]
[    59  88000]
[    45  32000]
[    29  83000]
[    26  80000]
[    49  28000]
[    23  20000]
[    32  18000]
[    60  42000]
[    19  76000]
[    36  99000]
[    19  26000]
[    60  83000]
[    24  89000]
[    27  58000]
[    40  47000]
[    42  70000]
[    32 150000]
[    35  77000]
[    22  63000]
[    45  22000]
[    27  89000]
[    18  82000]
[    42  79000]
[    40  60000]
[    53  34000]
[    47 107000]
[    58 144000]
[    59  83000]
[    24  55000]
[    26  35000]
[    58  38000]
[    42  80000]
[    40  75000]
[    59 130000]
[    46  41000]
[    41  60000]
[    42  64000]
[    37 146000]
[    23  48000]
[    25  33000]
[    24  84000]
[    27  96000]
[    23  63000]
```

```
 [    48  33000]
 [    48  90000]
 [    42 104000]]

Target Pengujian (y_test):
[0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 1 0 0 1 0 1 0 1 0 0 0 0 0 1 1 0 0 0 0
 0 0 1 0 0 0 0 1 0 0 1 0 1 1 0 0 0 1 1 0 0 1 0 0 1 0 1 0 1 0 0 0 0 1 0 0 1
 0 0 0 0 1 1 1 0 0 0 1 1 0 1 1 0 0 1 0 0 0 1 0 1 1 1]
```

0.4 4. Standardisasi Fitur

[10]:
```python
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
```

0.5 5. Pelatihan model Naive Bayes pada data pelatihan

[11]:
```python
from sklearn.naive_bayes import GaussianNB
classifier = GaussianNB()
classifier.fit(X_train, y_train)
```

[11]:
```
GaussianNB()
```

0.6 6. Prediksi hasil data pengujian

[12]:
```python
y_pred = classifier.predict(X_test)
```

0.7 7. Evaluasi Model

[13]:
```python
# Menghitung dan menampilkan confusion matrix dan akurasi
from sklearn.metrics import confusion_matrix, accuracy_score
cm = confusion_matrix(y_test, y_pred)
print("\nConfusion Matrix:")
print(cm)
print("\nAkurasi:", accuracy_score(y_test, y_pred))
```

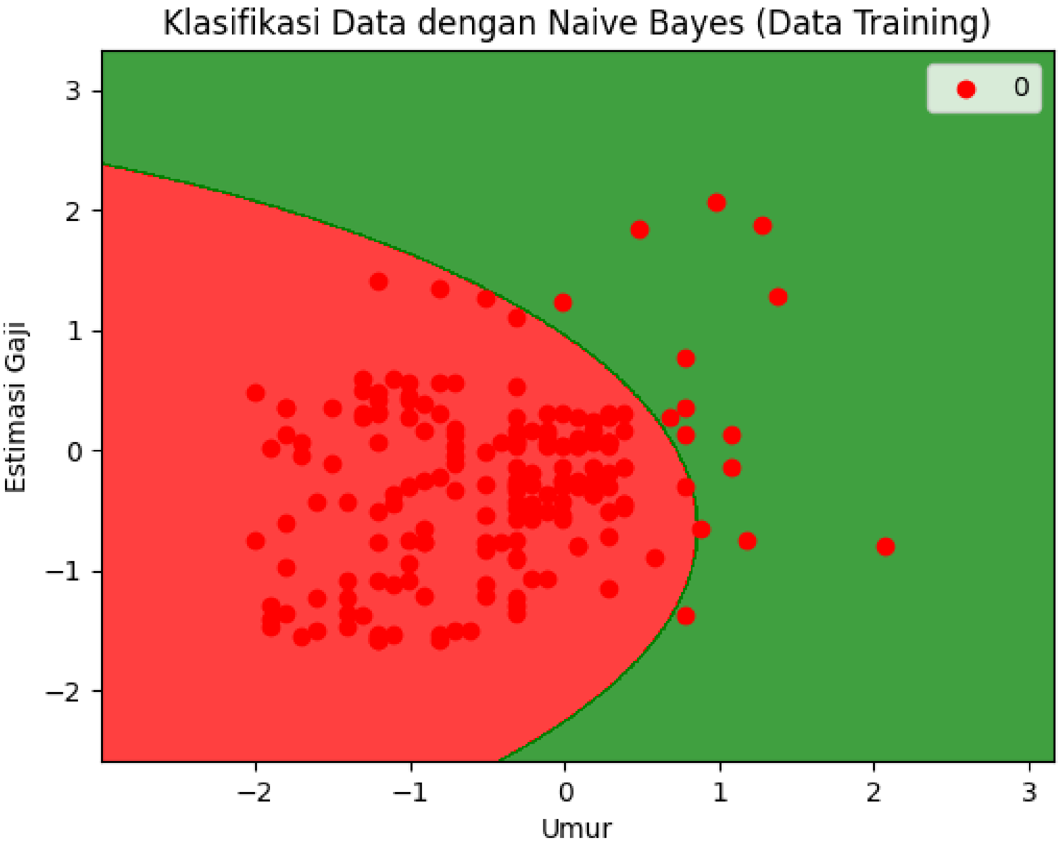
```
Confusion Matrix:
[[65  3]
 [ 7 25]]

Akurasi: 0.9
```

0.8 8. Visualisasi hasil pelatihan

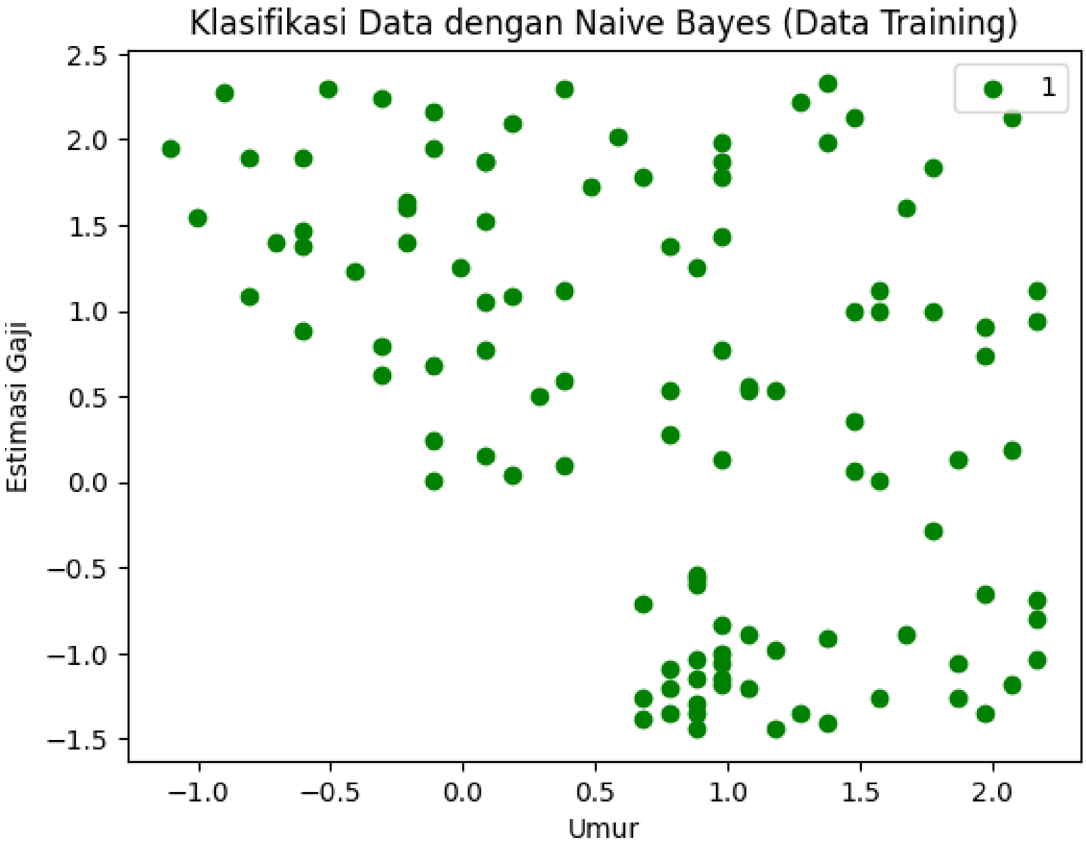
[16]:

```
from matplotlib.colors import ListedColormap
X_set, y_set = X_train, y_train
X1, X2 = np.meshgrid(np.arange(start = X_set[:, 0].min() - 1, stop = X_set[:,␣
 ↪0].max() + 1, step = 0.01),
                     np.arange(start = X_set[:, 1].min() - 1, stop = X_set[:,␣
 ↪1].max() + 1, step = 0.01))
plt.contourf(X1, X2, classifier.predict(np.array([X1.ravel(), X2.ravel()]).T).
 ↪reshape(X1.shape),
             alpha = 0.75, cmap = ListedColormap(('red', 'green')))
plt.xlim(X1.min(), X1.max())
plt.ylim(X2.min(), X2.max())
for i, j in enumerate(np.unique(y_set)):
    plt.scatter(X_set[y_set == j, 0], X_set[y_set == j, 1],
                c = ListedColormap(('red', 'green'))(i), label = j)
    plt.title('Klasifikasi Data dengan Naive Bayes (Data Training)')
    plt.xlabel('Umur')
    plt.ylabel('Estimasi Gaji')
    plt.legend()
    plt.show()
```

```
C:\Users\RESTU\AppData\Local\Temp\ipykernel_26372\3165894093.py:10: UserWarning:
*c* argument looks like a single numeric RGB or RGBA sequence, which should be
avoided as value-mapping will have precedence in case its length matches with
*x* & *y*.  Please use the *color* keyword-argument or provide a 2D array with a
single row if you intend to specify the same RGB or RGBA value for all points.
  plt.scatter(X_set[y_set == j, 0], X_set[y_set == j, 1],
```



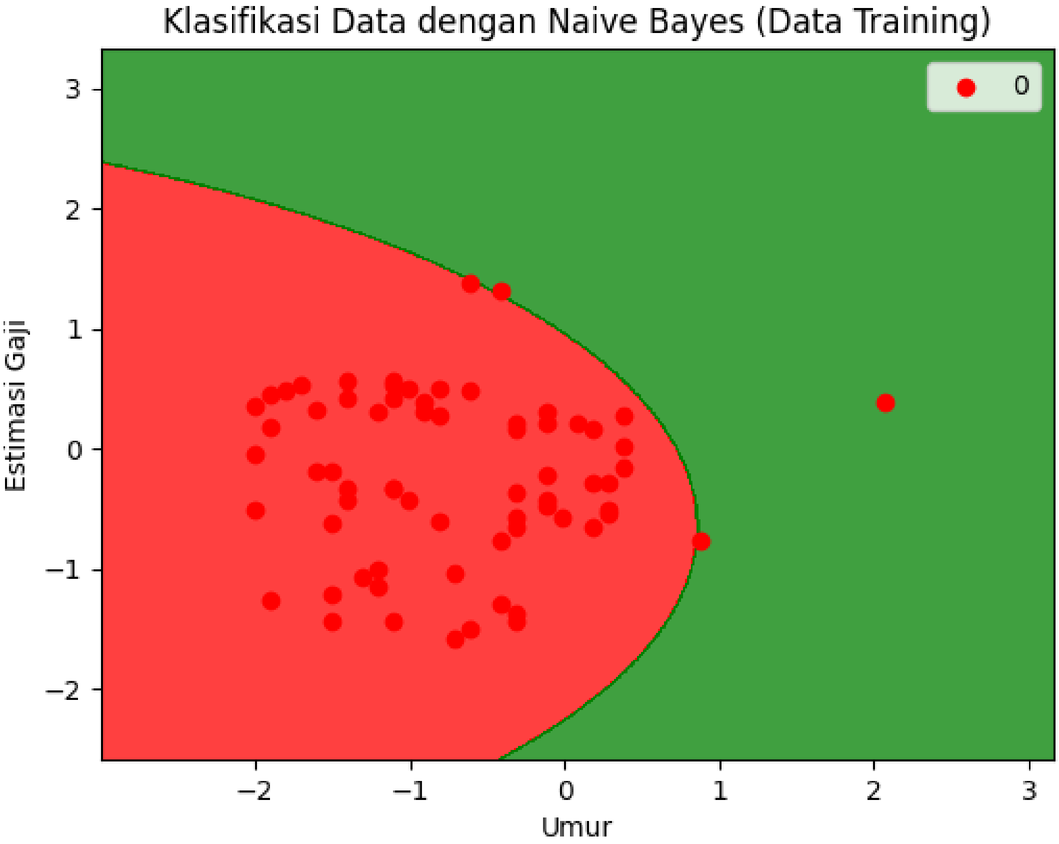
Klasifikasi Data dengan Naive Bayes (Data Training)
Estimasi Gaji
Umur
0
3
2
1
0
−1
−2
−2
−1
0
1
2
3

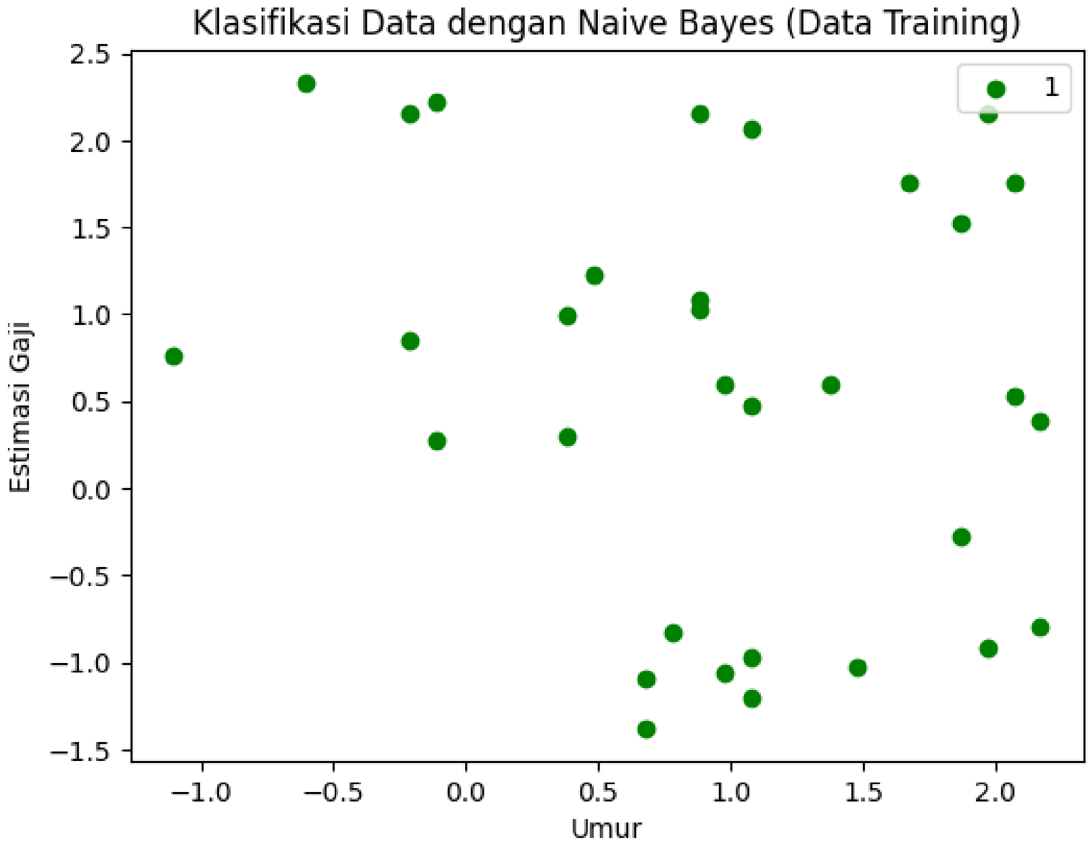


0.9 9. Visualisasi hasil pengujian

```
[15]: from matplotlib.colors import ListedColormap
X_set, y_set = X_test, y_test
X1, X2 = np.meshgrid(np.arange(start = X_set[:, 0].min() - 1, stop = X_set[:,␣
 ↪0].max() + 1, step = 0.01),
                     np.arange(start = X_set[:, 1].min() - 1, stop = X_set[:,␣
 ↪1].max() + 1, step = 0.01))
plt.contourf(X1, X2, classifier.predict(np.array([X1.ravel(), X2.ravel()]).T).
 ↪reshape(X1.shape),
             alpha = 0.75, cmap = ListedColormap(('red', 'green')))
plt.xlim(X1.min(), X1.max())
plt.ylim(X2.min(), X2.max())
for i, j in enumerate(np.unique(y_set)):
    plt.scatter(X_set[y_set == j, 0], X_set[y_set == j, 1],
                c = ListedColormap(('red', 'green'))(i), label = j)
    plt.title('Klasifikasi Data dengan Naive Bayes (Data Training)')
    plt.xlabel('Umur')
    plt.ylabel('Estimasi Gaji')
    plt.legend()
    plt.show()
```

```
C:\Users\RESTU\AppData\Local\Temp\ipykernel_26372\1992769346.py:10: UserWarning:
*c* argument looks like a single numeric RGB or RGBA sequence, which should be
avoided as value-mapping will have precedence in case its length matches with
*x* & *y*.  Please use the *color* keyword-argument or provide a 2D array with a
single row if you intend to specify the same RGB or RGBA value for all points.
  plt.scatter(X_set[y_set == j, 0], X_set[y_set == j, 1],
```




Klasifikasi Data dengan Naive Bayes (Data Training)
1
Estimasi Gaji
2.5
2.0
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0
Umur

[ ]:

[ ]: