naive_bayes_pet

October 18, 2024

0.1 Mengimpor pustaka yang diperlukan

[1]:
```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
```

0.2 Mengimpor dataset

[2]:
```
dataset = pd.read_csv('pet_adoption_data.csv')
```

[3]:
```
print(dataset.head(10))
```

```
   PetID PetType             Breed  AgeMonths   Color    Size   WeightKg  \
0    500    Bird          Parakeet        131  Orange   Large   5.039768
1    501  Rabbit            Rabbit         73   White   Large  16.086727
2    502     Dog  Golden Retriever        136  Orange  Medium   2.076286
3    503    Bird          Parakeet         97   White   Small   3.339423
4    504  Rabbit            Rabbit        123    Gray   Large  20.498100
5    505     Dog          Labrador         70   Brown   Large  20.986261
6    506    Bird          Parakeet        169   Brown   Small  10.902613
7    507     Cat           Siamese         13  Orange   Large   7.252683
8    508    Bird          Parakeet         49   Brown  Medium  24.597598
9    509    Bird          Parakeet         60    Gray   Large   7.295994

   Vaccinated  HealthCondition  TimeInShelterDays  AdoptionFee  PreviousOwner  \
0           1                0                 27          140              0
1           0                0                  8          235              0
2           0                0                 85          385              0
3           0                0                 61          217              1
4           0                0                 28           14              1
5           0                0                 87          301              1
6           1                0                 70          440              1
7           1                0                  3          137              0
8           1                1                 69          405              0
9           0                0                 73          231              1

   AdoptionLikelihood
0                   0
```

```
1                   0
2                   0
3                   0
4                   0
5                   0
6                   0
7                   1
8                   0
9                   0
```

[4]:
```
print(dataset.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2007 entries, 0 to 2006
Data columns (total 13 columns):
 #   Column              Non-Null Count  Dtype
---  ------              --------------  -----
 0   PetID               2007 non-null   int64
 1   PetType             2007 non-null   object
 2   Breed               2007 non-null   object
 3   AgeMonths           2007 non-null   int64
 4   Color               2007 non-null   object
 5   Size                2007 non-null   object
 6   WeightKg            2007 non-null   float64
 7   Vaccinated          2007 non-null   int64
 8   HealthCondition     2007 non-null   int64
 9   TimeInShelterDays   2007 non-null   int64
 10  AdoptionFee         2007 non-null   int64
 11  PreviousOwner       2007 non-null   int64
 12  AdoptionLikelihood  2007 non-null   int64
dtypes: float64(1), int64(8), object(4)
memory usage: 204.0+ KB
None
```

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

```
PetID                 0
PetType               0
Breed                 0
AgeMonths             0
Color                 0
Size                  0
WeightKg              0
Vaccinated            0
HealthCondition       0
TimeInShelterDays     0
AdoptionFee           0
PreviousOwner         0
```

```
AdoptionLikelihood    0
dtype: int64
```

0.3 Encoding data kategori (Atribut)

[6]:
```
from sklearn.preprocessing import LabelEncoder

categorical_cols = dataset.select_dtypes(include=['object']).columns
label_encoders = {}

for col in categorical_cols:
    label_encoders[col] = LabelEncoder()
    dataset[col] = label_encoders[col].fit_transform(dataset[col])
```

[7]:
```
# definisikan kolom kategorikal dan numerik
# kolom_kategorikal = ['PetType', 'Breed', 'Color', 'Size']
# kolom_numerik = ['AgeMonths', 'WeightKg', 'TimeInShelterDays', 'AdoptionFee',␣
↪'PreviousOwner']
```

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

# encoding kolom kategorikal
# label_encoder = LabelEncoder()
# for kolom in kolom_kategorikal:
#     dataset[kolom] = label_encoder.fit_transform(dataset[kolom])
```

[9]:
```
print("Tipe data setelah encoding:")
print(dataset.dtypes)
```

```
Tipe data setelah encoding:
PetID                   int64
PetType                 int32
Breed                   int32
AgeMonths               int64
Color                   int32
Size                    int32
WeightKg              float64
Vaccinated              int64
HealthCondition         int64
TimeInShelterDays       int64
AdoptionFee             int64
PreviousOwner           int64
AdoptionLikelihood      int64
dtype: object
```

[10]:
```
# Menampilkan dataset yang sudah diolah
dataset.head()
```

[10]:

	PetID	PetType	Breed	AgeMonths	Color	Size	WeightKg	Vaccinated
0	500	0	2	131	3	0	5.039768	1
1	501	3	5	73	4	0	16.086727	0
2	502	2	0	136	3	1	2.076286	0
3	503	0	2	97	4	2	3.339423	0
4	504	3	5	123	2	0	20.498100	0

	HealthCondition	TimeInShelterDays	AdoptionFee	PreviousOwner
0	0	27	140	0
1	0	8	235	0
2	0	85	385	0
3	0	61	217	1
4	0	28	14	1

	AdoptionLikelihood
0	0
1	0
2	0
3	0
4	0

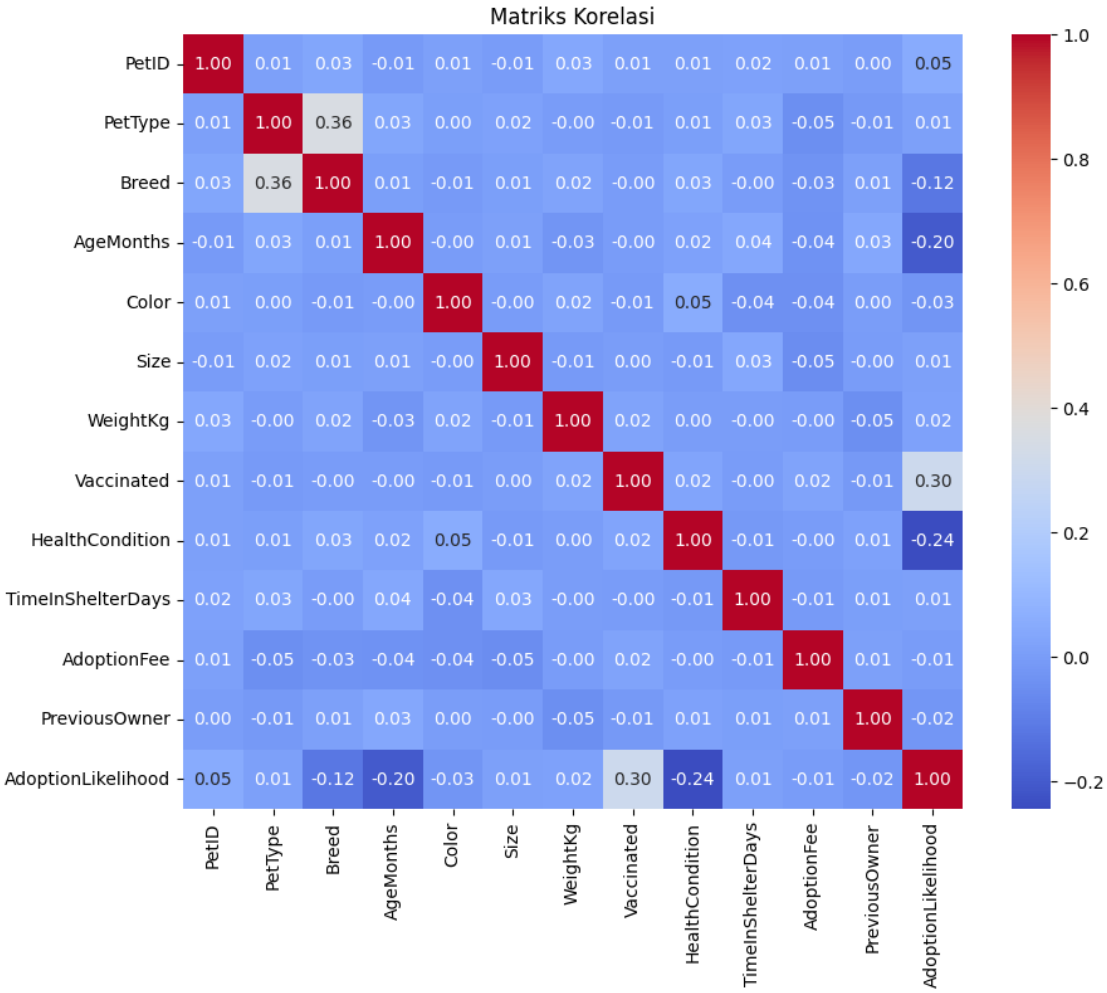
[11]:
```
# Memilih fitur dan label
X = dataset.iloc[:, [3, 9]].values
y = dataset.iloc[:, -1].values
```

[12]:
```
import seaborn as sns

numerical_data = dataset.select_dtypes(include=['number'])

# Menghitung matriks korelasi
correlation_matrix = dataset.corr()

# Visualisasi matriks korelasi
plt.figure(figsize=(10, 8))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt='.2f')
plt.title('Matriks Korelasi')
plt.show()
```



0.4 Pembagian dataset

[13]:
```
# 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 = 0.2,␣
 ↪random_state = 42)
```

[14]:
```
# 7. Memeriksa bentuk dan tipe data setelah pembagian
print("\nBentuk X_train:", X_train.shape)
print("Tipe data X_train:", X_train.dtype)
```

```
Bentuk X_train: (1605, 2)
Tipe data X_train: int64
```

[15]:
```
# Menampilkan hasil pembagian
print("Ukuran X_train:", X_train.shape)
print("Ukuran X_test:", X_test.shape)
print("Ukuran y_train:", y_train.shape)
print("Ukuran y_test:", y_test.shape)
```

```
Ukuran X_train: (1605, 2)
Ukuran X_test: (402, 2)
Ukuran y_train: (1605,)
Ukuran y_test: (402,)
```

[16]:
```
# 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):
[[ 68  74]
 [172  44]
 [169  43]
 …
 [ 19  31]
 [155  87]
 [ 34  83]]

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

[17]:
```
# 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):
[[116  71]
 [165  52]
 [106  22]
 [ 31  52]
 [125  39]
 [161  64]
 [  3  42]
 [ 56   2]
 [117  18]
 [  5  32]
 [151  19]
```

```
[172  31]
[ 59  20]
[120   1]
[  3  86]
[ 98   2]
[152  14]
[ 65   6]
[ 84   9]
[ 82   8]
[ 68  57]
[141  71]
[139  30]
[ 94  81]
[139  55]
[170  49]
[114  13]
[129  15]
[ 41  35]
[ 74  77]
[ 44  24]
[ 55  19]
[ 35  67]
[ 92  37]
[159  13]
[137  60]
[ 69  37]
[115  16]
[170  11]
[ 65  28]
[166   4]
[ 34  66]
[ 35  10]
[ 35  46]
[ 33  89]
[153  23]
[ 46  60]
[121  33]
[155  52]
[100  29]
[ 95  85]
[ 40  53]
[ 21  48]
[148  47]
[  4  78]
[165  34]
[ 16  15]
[148  48]
[ 65  29]
```

```
[142  13]
[149  31]
[149  81]
[ 44  59]
[128  53]
[  8  63]
[144  54]
[110   2]
[175  46]
[ 66  26]
[149  81]
[ 56   6]
[137  19]
[ 51   6]
[107  49]
[ 86  34]
[173  15]
[134  58]
[ 29   7]
[ 46  75]
[169  66]
[ 22   8]
[126  20]
[ 89  51]
[113  46]
[ 77  44]
[161  44]
[ 25  83]
[ 45  12]
[ 96  27]
[ 51   9]
[ 64  65]
[141  21]
[ 28  35]
[162  65]
[120  59]
[118  40]
[ 28   7]
[ 74  56]
[ 34  55]
[ 55  36]
[ 55  20]
[ 54   6]
[163  47]
[112  51]
[  6  23]
[ 41  35]
[116  72]
```

```
[125  39]
[ 64  66]
[172  14]
[ 82   8]
[137  37]
[170  84]
[ 86  60]
[ 47  78]
[ 92  43]
[ 87  10]
[  9  34]
[ 16  13]
[174  85]
[113  70]
[  3  27]
[107  76]
[177  50]
[122  35]
[109  65]
[169   4]
[129  57]
[122  45]
[125  40]
[147  76]
[ 42   3]
[ 42  21]
[179   3]
[106  15]
[ 49  58]
[ 70  74]
[162  78]
[ 19  52]
[156  11]
[ 41  19]
[149  62]
[129  25]
[ 65  88]
[ 24  56]
[ 78  77]
[146  66]
[ 78  26]
[ 73  57]
[109  83]
[162  18]
[ 70   8]
[171  23]
[ 40   5]
[ 30  20]
```

```
[ 33  12]
[154  68]
[ 87  31]
[ 98  33]
[ 34  89]
[ 21  79]
[136  73]
[101  57]
[ 88  14]
[  2   5]
[101  70]
[ 79  82]
[159   7]
[ 73  29]
[149  38]
[163  45]
[102  28]
[143  54]
[137  17]
[129  71]
[136  86]
[163  64]
[167  68]
[ 30  38]
[ 87  73]
[134  10]
[ 86  26]
[ 18  40]
[ 22  67]
[161   5]
[149  65]
[ 29  42]
[143   1]
[  4  74]
[ 74  38]
[116  30]
[173  10]
[153  66]
[119  51]
[120  22]
[175  60]
[124  31]
[112  18]
[108  34]
[148  85]
[172  70]
[161  84]
[140  45]
```

```
[ 58  34]
[131  84]
[128  77]
[137  13]
[138  66]
[ 41  47]
[ 82  21]
[149  51]
[  8  49]
[ 75  37]
[ 10   8]
[ 17  49]
[  8  89]
[ 81  14]
[ 17  42]
[ 86  30]
[140  72]
[121  86]
[ 43   6]
[124   7]
[ 82   1]
[  1  24]
[ 56  63]
[ 46  38]
[ 49  64]
[ 82   6]
[118  32]
[116  46]
[152  67]
[132  33]
[168   7]
[179  38]
[154  57]
[158  52]
[178  35]
[149   7]
[ 76  34]
[139  10]
[ 29  61]
[ 72  66]
[ 20  73]
[ 41  69]
[  8  41]
[  2  66]
[127  61]
[ 46  17]
[115  37]
[ 13  11]
```

```
[105  76]
[113  22]
[115  36]
[173  52]
[ 20  66]
[109  74]
[ 30  51]
[ 94   2]
[108  82]
[ 27  53]
[158  63]
[ 37  72]
[ 52  81]
[ 85  87]
[124  68]
[116  15]
[165  83]
[ 26  24]
[ 19  39]
[ 98  20]
[ 89  17]
[133  82]
[  3  56]
[ 98  61]
[175  42]
[ 82  74]
[ 51  42]
[121  10]
[ 84  57]
[145  57]
[ 47   7]
[ 66  14]
[160  62]
[ 70  17]
[ 28  63]
[ 17  54]
[141  89]
[130   2]
[  5   3]
[127  52]
[104  41]
[ 36  48]
[ 10  74]
[ 69  37]
[118  54]
[158  80]
[128  47]
[ 58  69]
```

```
[ 57  62]
[144   9]
[100  24]
[172  55]
[111  36]
[115  36]
[ 66  67]
[ 96  84]
[ 79  80]
[ 27  48]
[100  82]
[ 14  47]
[ 23   1]
[127  30]
[ 98   5]
[ 38  52]
[ 54   8]
[107  73]
[ 54  15]
[ 47  27]
[ 79  85]
[179  72]
[ 81  40]
[ 90  51]
[126  30]
[175   3]
[ 55  48]
[  4  52]
[159  18]
[126  49]
[ 57   2]
[ 12   6]
[ 57   2]
[163  68]
[118  43]
[ 77  59]
[ 54  54]
[176  61]
[ 12  81]
[ 16  54]
[ 24  61]
[  4   5]
[137  22]
[ 17  47]
[108  47]
[ 69  31]
[141  83]
[178  71]
```

```
[176  21]
[142  15]
[ 30  19]
[ 59  23]
[ 28  10]
[ 84  66]
[ 66   2]
[ 59   9]
[156  60]
[ 30  41]
[ 11  72]
[  4  87]
[ 14   7]
[ 13  17]
[ 41  59]
[102  53]
[109  51]
[178  79]
[127  81]
[153  73]
[169  44]
[178  10]
[ 49   2]
[ 88  61]
[  8  41]
[ 32  63]
[141  49]
[ 87  16]
[163   9]
[ 79  67]
[150  88]
[133   6]
[165   3]
[153  87]
[ 63  87]
[176   8]
[120  64]
[143  30]
[143  48]
[136  72]
[144  87]
[104  68]
[  3  70]
[ 19  25]
[119  48]
[170  42]
[ 87  87]
[126  66]
```

```
 [ 59  17]
 [ 61  69]
 [ 20  32]
 [  8  17]
 [163  43]
 [175  17]
 [ 49  38]]

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

0.5 Standardisasi fitur

[18]:
```
# Standarisasi kolom numerik
from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```

0.6 Pelatihan model Naive Bayes pada data pelatihan

[19]:
```
from sklearn.naive_bayes import GaussianNB
classifier = GaussianNB()
classifier.fit(X_train, y_train)
```

[19]:
```
GaussianNB()
```

0.7 Prediksi hasil data pengujian

[20]:
```
y_pred = classifier.predict(X_test)
```

0.8 Evaluasi model

[21]:
```
# Menghitung dan menampilkan confusion matrix
from sklearn.metrics import confusion_matrix, accuracy_score
cm = confusion_matrix(y_test, y_pred)
print("\nConfusion Matrix:")
```

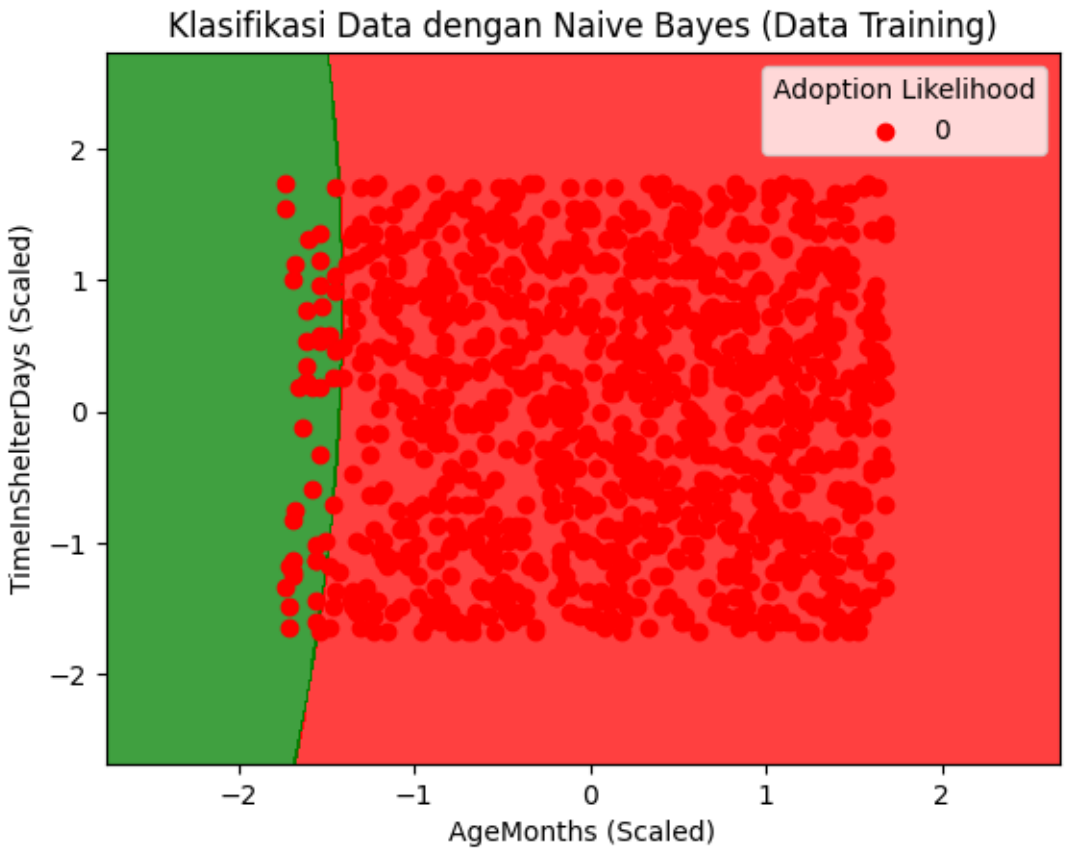
```
print(cm)
# Menghitung akurasi
accuracy = accuracy_score(y_test, y_pred)
print(f'Akurasi Model Naive Bayes: {accuracy * 100:.2f}%')
```

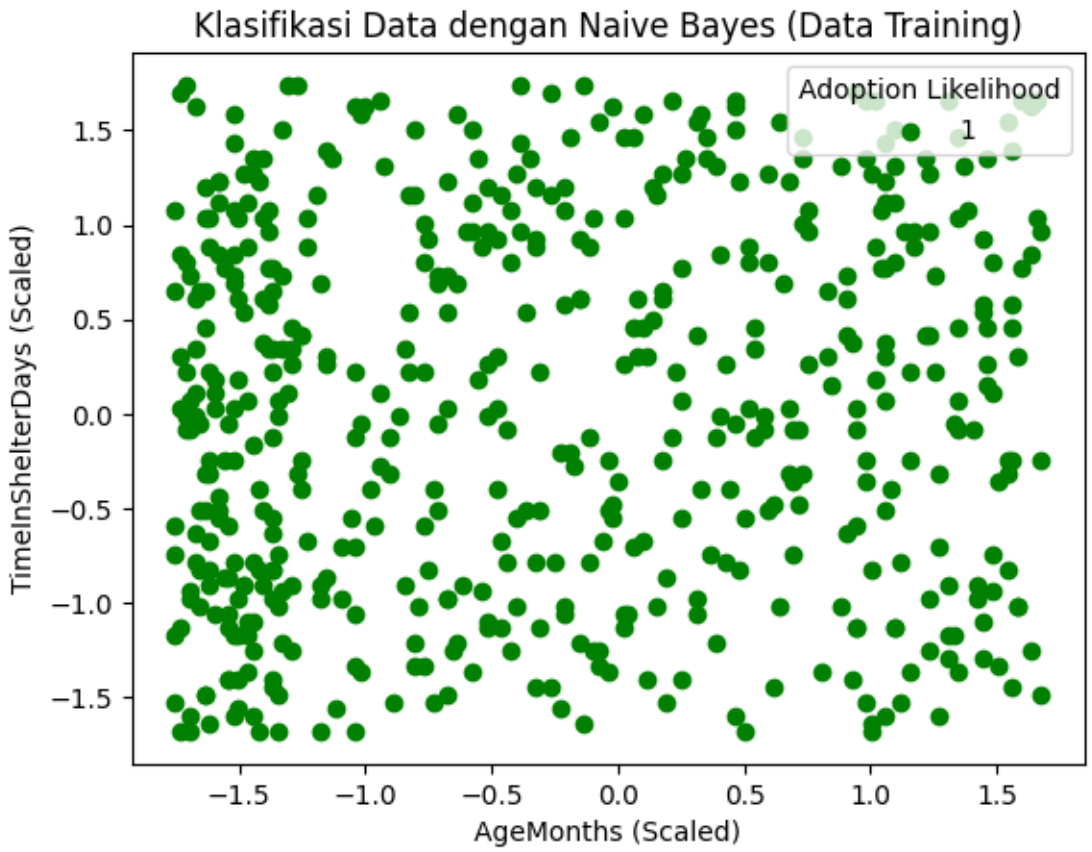
```
Confusion Matrix:
[[260  10]
 [106  26]]
Akurasi Model Naive Bayes: 71.14%
```

0.9 Visualisasi hasil pelatihan

[22]:
```
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('AgeMonths (Scaled)')
    plt.ylabel('TimeInShelterDays (Scaled)')
    plt.legend(title='Adoption Likelihood', loc='upper right')
    plt.show()
```

```
C:\Users\RESTU\AppData\Local\Temp\ipykernel_6536\1858157641.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)
Adoption Likelihood
0
TimeInShelterDays (Scaled)
AgeMonths (Scaled)
2
1
0
−1
−2
−2
−1
0
1
2

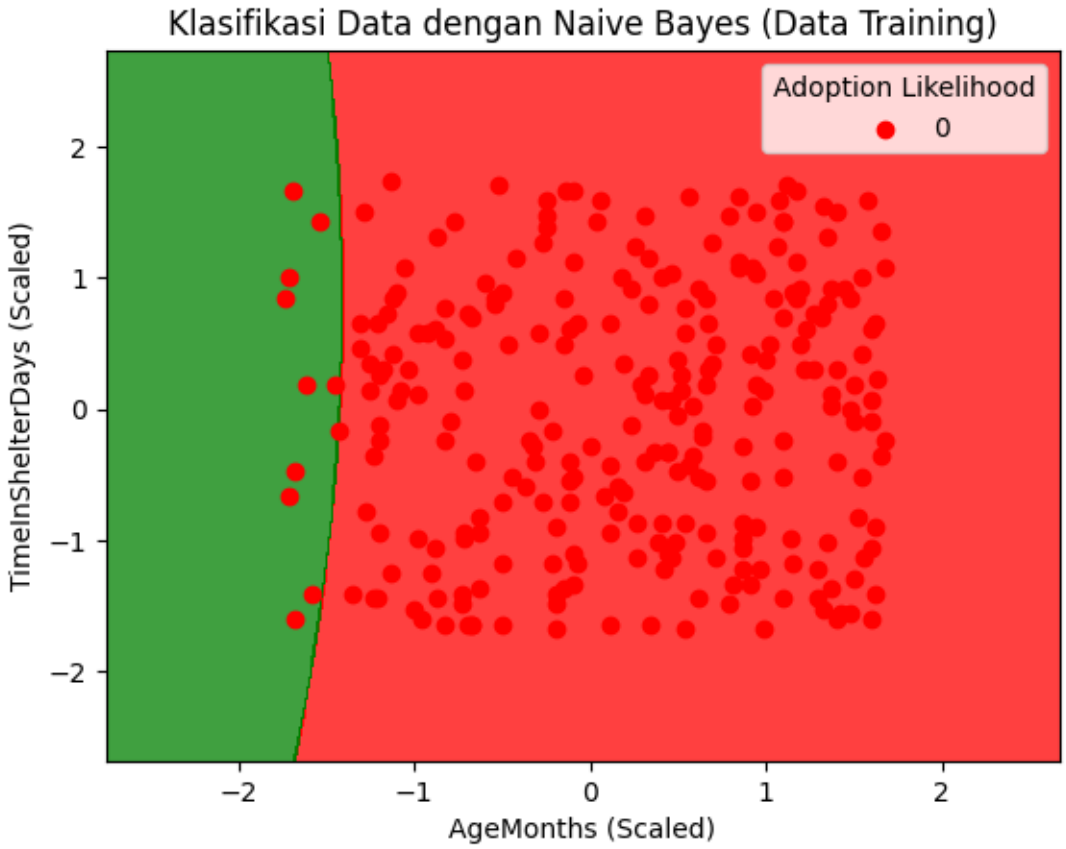


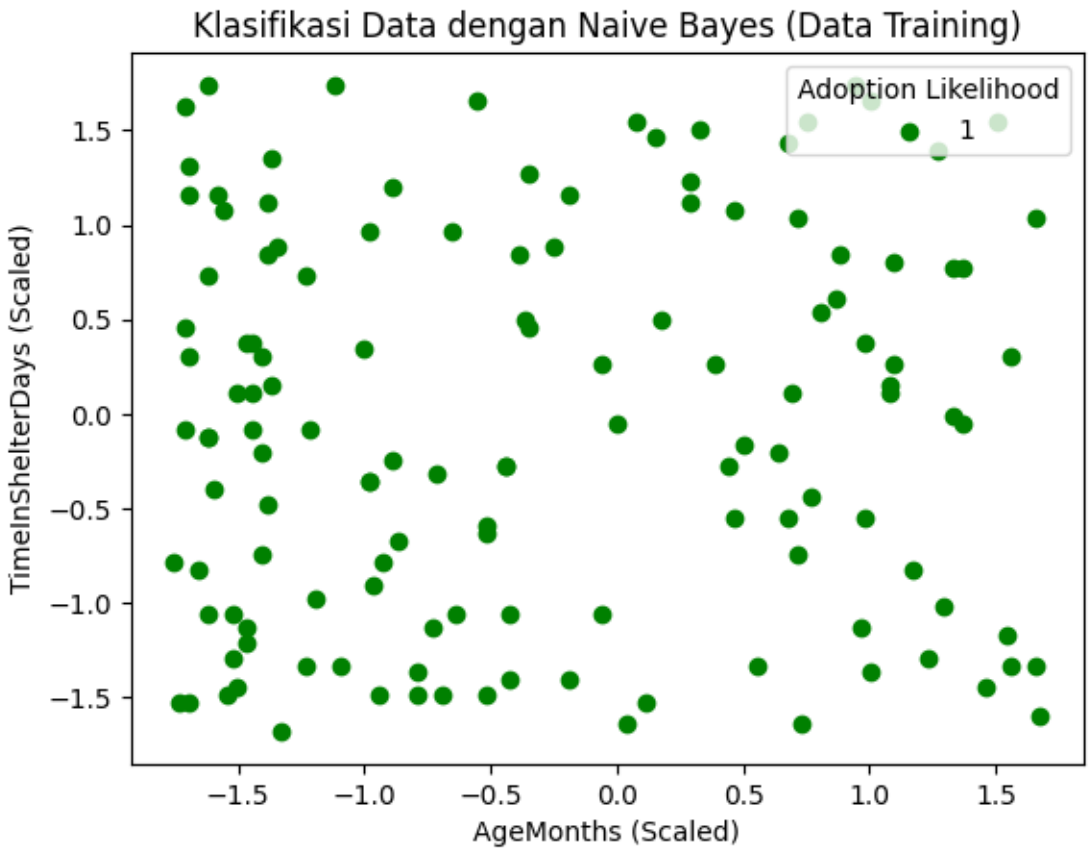
0.10 Visualisasi hasil pengujian

[23]:
```
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('AgeMonths (Scaled)')
    plt.ylabel('TimeInShelterDays (Scaled)')
```

```
    plt.legend(title='Adoption Likelihood', loc='upper right')
    plt.show()
```

```
C:\Users\RESTU\AppData\Local\Temp\ipykernel_6536\2972547322.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)
Adoption Likelihood
1
TimeInShelterDays (Scaled)
AgeMonths (Scaled)
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5