Praktikum & Tugas 9: Machine Learning Naïve Bayes

Rizky Hilmiawan Anggoro[1] - 01102221401

[1] Teknik Informatika, STT Terpadu Nurul Fikri, Depok

*E-mail: rizk22140ti@student.nurulfikri.ac.id

Abstract.
Penelitian ini bertujuan untuk mengadaptasi dan menerapkan alur kerja klasifikasi pembelajaran mesin yang sebelumnya terdapat dalam Jupyter Notebook Praktikum09.ipynb ke dataset baru, yaitu data diagnostik kanker payudara (WDBC) yang disediakan dalam data.csv. Proses dimulai dengan pembersihan data, termasuk penghapusan kolom yang tidak relevan (id dan Unnamed: 32), diikuti dengan pra-pemrosesan di mana variabel target kategorikal 'diagnosis' (Benign/Malignant) dikonversi menjadi nilai numerik (0/1). Fitur-fitur kemudian dinormalisasi menggunakan StandardScaler sebelum data dibagi menjadi set pelatihan dan pengujian. Model Gaussian Naive Bayes dilatih pada data yang telah di-scaling dan dievaluasi secara komprehensif. Hasil evaluasi menunjukkan kinerja model yang kuat, dengan Akurasi Pengujian (Testing Accuracy) sebesar 0.921 dan Akurasi Cross-Validation rata-rata 5-lipatan sebesar 0.938, mengindikasikan bahwa model Naive Bayes efektif dalam mengklasifikasikan kasus kanker payudara pada dataset ini.

1. Inisialisasi dan Pemuatan Data

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
from sklearn.preprocessing import StandardScaler
from sklearn.naive_bayes import GaussianNB

# Memuat dataset baru
from google.colab import drive
drive.mount('/content/gdrive')

import os

path = "/content/gdrive/MyDrive/praktikum_ml/praktikum08"
try:
    print(os.listdir(path))
except FileNotFoundError:
    print(f"Directory not found: {path}")

data = pd.read_csv(path + "/data/data.csv")
data.head()
```

```
Mounted at /content/gdrive
['notebooks', 'data']
```

	id	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean
0	842302	M	17.99	10.38	122.80	1001.0	0.11840	0.27760
1	842517	M	20.57	17.77	132.90	1326.0	0.08474	0.07864
2	84300903	M	19.69	21.25	130.00	1203.0	0.10960	0.15990
3	84348301	M	11.42	20.38	77.58	386.1	0.14250	0.28390
4	84358402	M	20.29	14.34	135.10	1297.0	0.10030	0.13280

5 rows × 33 columns

Bagian ini berfungsi untuk mengimpor semua pustaka Python yang diperlukan untuk analisis data dan pembelajaran mesin, seperti pandas untuk manipulasi data, numpy untuk operasi numerik, matplotlib dan seaborn untuk visualisasi, serta berbagai modul dari scikit-learn untuk pemodelan. Setelah inisialisasi, data diagnostik kanker payudara dimuat dari file data.csv ke dalam sebuah DataFrame data menggunakan fungsi pd.read_csv(), dan lima baris pertama data ditampilkan untuk verifikasi awal.

2. Inspeksi dan Pembersihan Data Awal

```
# Inspeksi data
data.shape
```

```
(569, 33)
```

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 569 entries, 0 to 568
Data columns (total 33 columns):
 #   Column                   Non-Null Count  Dtype
---  ------                   --------------  -----
 0   id                       569 non-null    int64
 1   diagnosis                569 non-null    object
 2   radius_mean              569 non-null    float64
 3   texture_mean             569 non-null    float64
 4   perimeter_mean           569 non-null    float64
 5   area_mean                569 non-null    float64
 6   smoothness_mean          569 non-null    float64
 7   compactness_mean         569 non-null    float64
 8   concavity_mean           569 non-null    float64
 9   concave points_mean      569 non-null    float64
 10  symmetry_mean            569 non-null    float64
 11  fractal_dimension_mean   569 non-null    float64
 12  radius_se                569 non-null    float64
 13  texture_se               569 non-null    float64
 14  perimeter_se             569 non-null    float64
 15  area_se                  569 non-null    float64
 16  smoothness_se            569 non-null    float64
 17  compactness_se           569 non-null    float64
 18  concavity_se             569 non-null    float64
 19  concave points_se        569 non-null    float64
 20  symmetry_se              569 non-null    float64
 21  fractal_dimension_se     569 non-null    float64
 22  radius_worst             569 non-null    float64
 23  texture_worst            569 non-null    float64
 24  perimeter_worst          569 non-null    float64
 25  area_worst               569 non-null    float64
 26  smoothness_worst         569 non-null    float64
 27  compactness_worst        569 non-null    float64
 28  concavity_worst          569 non-null    float64
 29  concave points_worst     569 non-null    float64
 30  symmetry_worst           569 non-null    float64
 31  fractal_dimension_worst  569 non-null    float64
 32  Unnamed: 32              0 non-null      float64
dtypes: float64(31), int64(1), object(1)
```

```
# Cek nilai yang hilang (missing values)
data.isnull().sum()
```

	0
id	0
diagnosis	0
radius_mean	0
texture_mean	0
perimeter_mean	0
area_mean	0
smoothness_mean	0
compactness_mean	0
concavity_mean	0
concave points_mean	0
symmetry_mean	0
fractal_dimension_mean	0
radius_se	0
texture_se	0
perimeter_se	0
area_se	0
smoothness_se	0
compactness_se	0
concavity_se	0
concave points_se	0
symmetry_se	0

```
# Kolom 'id' tidak relevan untuk klasifikasi, dan kolom terakhir (Unnamed: 32) kosong. Kita hapus keduanya.
data = data.drop(columns=['id', 'Unnamed: 32'], axis=1)
data.head()
```

	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean	concavity_mean
0	M	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.3001
1	M	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.0869
2	M	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.1974
3	M	11.42	20.38	77.58	386.1	0.14250	0.28390	0.2414
4	M	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.1980

5 rows × 31 columns

Serangkaian sel kode berikutnya didedikasikan untuk memahami struktur data dan melakukan pembersihan awal. Perintah data.shape memberikan informasi tentang jumlah baris dan kolom dalam dataset, sementara data.info() menampilkan ringkasan detail, termasuk tipe data setiap kolom dan jumlah nilai non-null. Selanjutnya, data.isnull().sum() digunakan untuk

mengidentifikasi dan menghitung total nilai yang hilang (missing values) di setiap kolom. Berdasarkan hasil inspeksi, kolom 'id' yang tidak relevan untuk proses klasifikasi dan kolom 'Unnamed: 32' yang teridentifikasi kosong (seperti yang terlihat dari data.info()) dihapus dari dataset menggunakan fungsi data.drop().

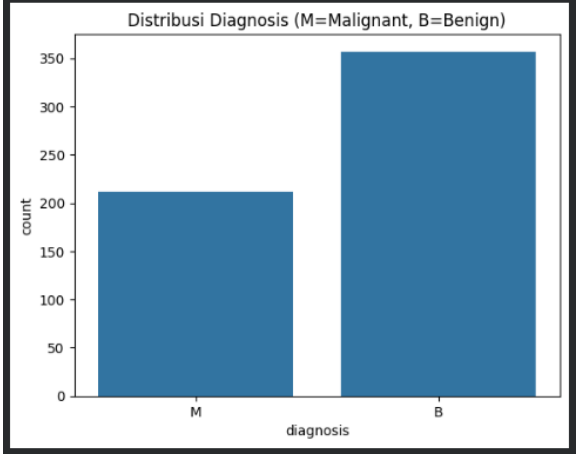
3. Analisis dan Pra-pemrosesan Variabel Target

```
# Cek distribusi kelas target 'diagnosis'
data['diagnosis'].value_counts()
```

diagnosis	count
B	357
M	212

dtype: int64

```
# Visualisasi distribusi kelas target
sns.countplot(x='diagnosis', data=data)
plt.title('Distribusi Diagnosis (M=Malignant, B=Benign)')
plt.show()
```




```
# Konversi fitur kategorikal 'diagnosis' menjadi numerik (B=0, M=1)
data['diagnosis'].replace({'B': 0, 'M': 1}, inplace=True)
data.head()
```

	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean
0	1	17.99	10.38	122.80	1001.0	0.11840
1	1	20.57	17.77	132.90	1326.0	0.08474
2	1	19.69	21.25	130.00	1203.0	0.10960
3	1	11.42	20.38	77.58	386.1	0.14250
4	1	20.29	14.34	135.10	1297.0	0.10030

5 rows × 31 columns

Bagian ini berfokus pada variabel target, yaitu 'diagnosis'. Pertama, data['diagnosis'].value_counts() digunakan untuk melihat distribusi jumlah kasus jinak (B - Benign) dan ganas (M - Malignant) dalam dataset. Distribusi ini kemudian divisualisasikan menggunakan countplot dari seaborn untuk memberikan representasi grafis yang jelas. Langkah pra-pemrosesan krusial dilakukan dengan mengkonversi nilai kategorikal 'B' dan 'M' menjadi nilai numerik 0 dan 1, yang merupakan format yang diperlukan oleh algoritma pembelajaran mesin.

4. Pemisahan Fitur dan Target

```
# Memisahkan fitur (X) dan target (Y)
X = data.drop(columns=['diagnosis'])
Y = data['diagnosis']

X.head()
```

	radius_mean	texture_mean	perimeter_mean	area_mean
0	17.99	10.38	122.80	1001.0
1	20.57	17.77	132.90	1326.0
2	19.69	21.25	130.00	1203.0
3	11.42	20.38	77.58	386.1
4	20.29	14.34	135.10	1297.0

5 rows × 30 columns

```
Y.head()
```

	diagnosis
0	1
1	1
2	1
3	1
4	1

dtype: int64

Setelah pembersihan dan konversi variabel target, data dipisahkan menjadi dua komponen utama: Fitur (X) dan Target (Y). Fitur X adalah semua kolom yang akan digunakan untuk memprediksi, yang diperoleh dengan menghapus kolom diagnosis dari DataFrame utama. Sementara itu, Target Y hanya berisi kolom diagnosis. Kedua DataFrame X dan Y kemudian ditampilkan lima baris teratasnya untuk memastikan pemisahan telah dilakukan dengan benar.

5. **Pembagian Data Latih dan Uji**

```
# Membagi data menjadi data latih (training) dan data uji (testing)
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2, stratify=Y, random_state=42)

print(X.shape, X_train.shape, X_test.shape)
```

```
(569, 30) (455, 30) (114, 30)
```

Data yang telah dipisahkan kemudian dibagi menjadi set data latih (training set) dan set data uji (testing set) menggunakan fungsi train_test_split dari scikit-learn. Pembagian ini menggunakan rasio 80% untuk pelatihan dan 20% untuk pengujian (test_size=0.2). Parameter stratify=Y memastikan bahwa proporsi kelas target (jinak dan ganas) dipertahankan secara merata di kedua

set data, mencegah bias. Hasil pembagian diverifikasi dengan mencetak dimensi (shape) dari set data asli, latih, dan uji.

6. Normalisasi Fitur (Scaling)

```
# Normalisasi/Scaling fitur menggunakan StandardScaler
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

Karena fitur-fitur dalam dataset memiliki rentang nilai yang berbeda-beda, langkah normalisasi atau scaling sangat penting untuk mencegah fitur dengan nilai besar mendominasi proses pelatihan model. StandardScaler digunakan untuk mengubah data sehingga memiliki rata-rata nol dan deviasi standar satu. Proses fit_transform diterapkan pada data latih (X_train) untuk mempelajari parameter scaling, dan kemudian transform diterapkan pada data uji (X_test) menggunakan parameter yang sama untuk menghindari kebocoran data (data leakage).

7. Pelatihan Model Naive Bayes

```
# Inisialisasi dan latih model Naive Bayes (GaussianNB)
nb_model = GaussianNB()
nb_model.fit(X_train_scaled, Y_train)
```

```
GaussianNB
GaussianNB()
```

Pada bagian ini, model klasifikasi Naive Bayes diimplementasikan. Secara spesifik, GaussianNB (Gaussian Naive Bayes) dipilih karena fitur-fitur dalam dataset ini diasumsikan mengikuti distribusi Gaussian (normal). Model diinisialisasi dan kemudian dilatih (fit) menggunakan data latih yang telah di-scaling (X_train_scaled) dan variabel target latih (Y_train).

8. Prediksi dan Evaluasi Akurasi

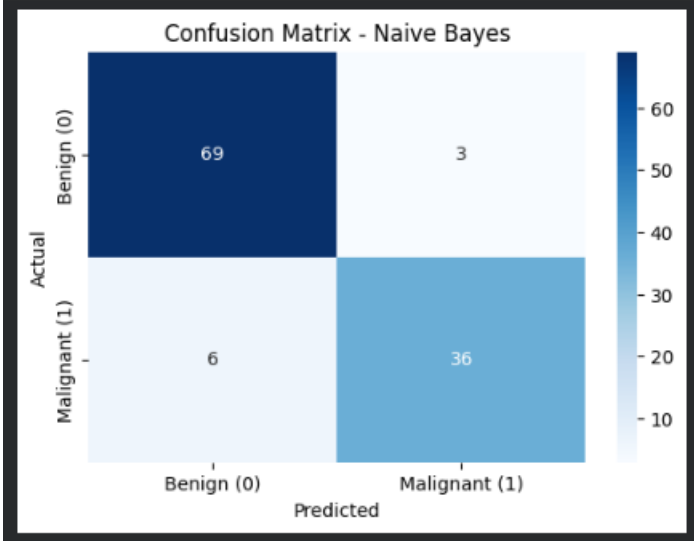
```
# Prediksi dan Evaluasi Akurasi
train_pred_nb = nb_model.predict(X_train_scaled)
test_pred_nb = nb_model.predict(X_test_scaled)

print("Training Accuracy (NB): ", accuracy_score(Y_train, train_pred_nb))
print("Testing Accuracy (NB): ", accuracy_score(Y_test, test_pred_nb))
```

```
Training Accuracy (NB):  0.945054945054945
Testing Accuracy (NB):  0.9210526315789473
```

Setelah model dilatih, model digunakan untuk membuat prediksi pada set data latih dan set data uji. Akurasi model dihitung menggunakan fungsi accuracy_score dengan membandingkan hasil prediksi dengan nilai target yang sebenarnya (Y_train dan Y_test). Hasil akurasi pelatihan dan pengujian dicetak untuk menilai seberapa baik model mempelajari data dan seberapa baik model dapat digeneralisasi ke data baru.

9. Visualisasi Confusion Matrix

```python
# Visualisasi Confusion Matrix
plt.figure(figsize=(6,4))
cm_nb = confusion_matrix(Y_test, test_pred_nb)
sns.heatmap(cm_nb, annot=True, fmt="d", cmap="Blues",
            xticklabels=['Benign (0)', 'Malignant (1)'],
            yticklabels=['Benign (0)', 'Malignant (1)'])
plt.title("Confusion Matrix - Naive Bayes")
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.show()
```



Untuk evaluasi yang lebih mendalam, Confusion Matrix dibuat dan divisualisasikan menggunakan heatmap dari seaborn. Confusion Matrix memberikan rincian kinerja model, menunjukkan jumlah prediksi benar (True Positive dan True Negative) dan prediksi salah (False Positive dan False Negative) untuk setiap kelas. Visualisasi ini membantu memahami jenis kesalahan yang dibuat oleh model.

10. Classification Report dan Cross Validation

```python
# Classification Report
print("\nClassification Report (NB):")
print(classification_report(Y_test, test_pred_nb))
```

```
Classification Report (NB):
              precision    recall  f1-score   support

           0       0.92      0.96      0.94        72
           1       0.92      0.86      0.89        42

    accuracy                           0.92       114
   macro avg       0.92      0.91      0.91       114
weighted avg       0.92      0.92      0.92       114
```

```python
# Cross Validation
from sklearn.model_selection import cross_val_score
cv_nb = cross_val_score(nb_model, X, Y, cv=5, scoring='accuracy')

print("\nNaive Bayes Cross Validation Accuracy (5-Fold):")
print("Scores:", cv_nb)
print("Mean Accuracy:", cv_nb.mean())
print("Std Deviation:", cv_nb.std())
```

```
Naive Bayes Cross Validation Accuracy (5-Fold):
Scores: [0.92105263 0.92105263 0.94736842 0.94736842 0.95575221]
Mean Accuracy: 0.9385188635305075
Std Deviation: 0.014585994424363306
```

ikan gambaran komprehensif tentang kinerja menyajikan metrik Precision, Recall, dan F1-

Score untuk setiap kelas, memberikan wawasan tentang kualitas prediksi model. Kedua, Cross Validation (5-Fold) dilakukan pada seluruh dataset untuk menguji stabilitas model. Hasil skor akurasi dari setiap fold dan rata-rata akurasi keseluruhan dicetak, yang menunjukkan seberapa konsisten kinerja model terlepas dari bagaimana data dibagi.