Tugas 4: Praktikum Mandiri Logistic Regression

Rika Rahma - 0110222134 [1]

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

*E-mail: rika22134ti@student.nurulfikri.ac.id

Abstract. Praktikum Mandiri ini bertujuan untuk memprediksi kemungkinan seseorang membeli mobil berdasarkan beberapa faktor seperti usia, status, jenis kelamin, kepemilikan mobil, dan penghasilan menggunakan algoritma Logistic Regression. Data dibagi menjadi data latih dan data uji dengan perbandingan 80:20, serta dilakukan normalisasi fitur menggunakan *StandardScaler* agar skala antar variabel seragam. Model Logistic Regression kemudian dilatih dan dievaluasi menggunakan metrik akurasi, confusion matrix, dan classification report. Hasil evaluasi menunjukkan akurasi model sebesar 93%, yang menandakan bahwa model mampu mengklasifikasikan calon pembeli dengan cukup baik. Selain itu, model juga diuji pada dataset baru untuk melihat kemampuan generalisasi, di mana diperoleh prediksi bahwa 40% dari data baru diperkirakan akan membeli mobil.

1. Penjelasan Kode dan Output

1.1 Import Library

Kode:

```
# Import Library
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
from google.colab import drive
```

Gambar 1.1 Import Library

Penjelasan:

Bagian ini digunakan untuk memanggil seluruh library yang dibutuhkan dalam proses analisis dan pemodelan:

- pandas (pd) → untuk membaca dan mengolah dataset dalam bentuk tabel.
- train_test_split → untuk membagi data menjadi data latih (training) dan data uji (testing).
- LogisticRegression → algoritma klasifikasi untuk memprediksi dua kemungkinan (0 = tidak beli, 1 = beli).
- StandardScaler → melakukan standarisasi pada fitur numerik agar semua fitur memiliki skala serupa.
- accuracy_score, classification_report, confusion_matrix → untuk mengevaluasi performa model.
- drive → untuk menghubungkan Google Colab ke Google Drive agar dataset bisa diakses.

1.2 Hubungkan ke Google Drive

Kode:

```
# Hubungkan ke Google Drive
drive.mount('/content/gdrive')
path = '/content/gdrive/MyDrive/Praktikum_ML/praktikum04/data/'
```

Gambar 1.2 Menghubungkan ke Google Drive

Penjelasan:

- drive.mount('/content/gdrive') menghubungkan Google Colab ke akun Google Drive.
- Variabel path menyimpan lokasi folder tempat file dataset disimpan. Dengan ini, Colab dapat membaca file CSV secara langsung dari Drive.

1.3 Load Dataset

Kode:

```
# Load Dataset
df = pd.read_csv(path + 'calonpembelimobil.csv')
df.head()
```

	ID	Usia	Status	Kelamin	Memiliki_Mobil	Penghasilan	Beli_Mobil
0	1	32	1	0	0	240	1
1	2	49	2	1	1	100	0
2	3	52	1	0	2	250	1
3	4	26	2	1	1	130	0
4	5	45	3	0	2	237	1

Gambar 1.3 Load Dataset

Penjelasan:

- pd.read_csv() digunakan untuk membaca file CSV berisi data calon pembeli mobil.
- df.head() menampilkan 5 baris pertama dari dataset untuk memastikan data terbaca dengan benar.

Interpretasi:

Dataset berisi 7 kolom:

- ID → identitas calon pembeli.
- Usia, Status, Kelamin, Memiliki_Mobil, Penghasilan → fitur yang digunakan untuk prediksi.
- Beli_Mobil → target atau label (0 = tidak beli, 1 = beli).

1.4 Cek Distribusi Target

Kode:

```
# Cek Distribusi Target (Beli_Mobil)
print("\nDistribusi kelas (Beli_Mobil):")
print(df['Beli_Mobil'].value_counts())
print(df['Beli_Mobil'].value_counts(normalize=True))
```

```
Distribusi kelas (Beli_Mobil):
Beli_Mobil
1    633
0    367
Name: count, dtype: int64
Beli_Mobil
1    0.633
0    0.367
Name: proportion, dtype: float64
```

Gambar 1.4 Cek Distribusi Target

Penjelasan:

- value_counts() menghitung jumlah data tiap kelas target.
- normalize=True menampilkan proporsi dalam bentuk persentase. Hasilnya menunjukkan bahwa:
- 63,3% data berlabel “Beli” (1).
- 36,7% data berlabel “Tidak Beli” (0).

1.5 Preprocessing Data

Kode:

```
# Preprocessing Data
# Pisahkan fitur (X) dan target (y)
X = df.drop(['ID', 'Beli_Mobil'], axis=1)
y = df['Beli_Mobil']

# Split data latih & uji (80:20)
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42, stratify=y
)

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

Gambar 1.5 Preprocessing Data

Penjelasan:

- X = df.drop(['ID', 'Beli_Mobil'], axis=1) berfungsi untuk memisahkan fitur independen (variabel input) dari dataset dengan menghapus kolom ID dan Beli_Mobil, karena Beli_Mobil merupakan variabel target yang ingin diprediksi, sedangkan ID hanya penanda data.
- y = df['Beli_Mobil'] menyimpan variabel dependen (target), yaitu keputusan apakah seseorang membeli mobil atau tidak.

- train_test_split() digunakan untuk membagi dataset menjadi data latih (80%) dan data uji (20%) agar model bisa diuji pada data yang belum pernah dilihat sebelumnya. Parameter random_state=42 memastikan hasil pembagian selalu sama setiap kali dijalankan, sedangkan stratify=y menjaga proporsi kelas target tetap seimbang di data latih dan uji.
- StandardScaler() digunakan untuk menstandarkan skala fitur numerik agar setiap variabel memiliki skala yang sebanding.
- scaler.fit_transform(X_train) menyesuaikan skala data berdasarkan rata-rata dan standar deviasi dari data latih, kemudian mentransformasikannya.
- scaler.transform(X_test) menerapkan skala yang sama pada data uji tanpa menghitung ulang, agar hasil tetap konsisten.

1.6 Latih Model Logistic Regression

Kode:

```
# Latih Model Logistic Regression
model = LogisticRegression(
    class_weight='balanced',   # untuk menangani imbalance data
    random_state=42,
    max_iter=1000
)
model.fit(X_train_scaled, y_train)
print("\n✅ Model Logistic Regression berhasil dilatih!")


✅ Model Logistic Regression berhasil dilatih!
```

Gambar 1.6 Melatih Model Logistic Regeression

Penjelasan:

- Model Logistic Regression dilatih dengan parameter:
  - class_weight='balanced' → menyesuaikan bobot tiap kelas agar model tidak bias pada kelas mayoritas.
  - random_state=42 → untuk hasil yang konsisten.
  - max_iter=1000 → memperbesar jumlah iterasi agar model konvergen.
- fit() berarti model mempelajari pola hubungan antara fitur dan label (0/1).
- Outputnya menyatakan bahwa model logistic regression berhasil dilatih, artinya modelnya sudah berhasil dilatih dan tidak ada error

1.7 Evaluasi Model

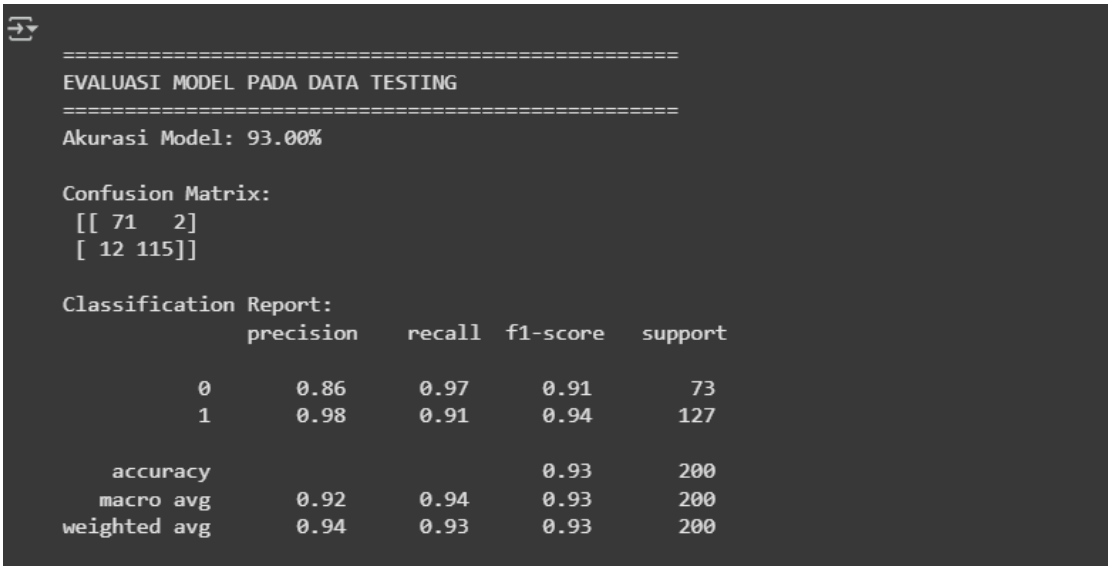
Kode:

```
# Evaluasi Model Pada Data Testing
y_pred = model.predict(X_test_scaled)

print("\n" + "="*50)
print("EVALUASI MODEL PADA DATA TESTING")
print("="*50)
print(f"Akurasi Model: {accuracy_score(y_test, y_pred):.2%}")
print("\nConfusion Matrix:\n", confusion_matrix(y_test, y_pred))
print("\nClassification Report:\n", classification_report(y_test, y_pred, zero_division=1))
```

Gambar 1.7 Evaluasi Model Pada data Testing

Output:



```
==================================================
EVALUASI MODEL PADA DATA TESTING
==================================================
Akurasi Model: 93.00%

Confusion Matrix:
 [[ 71   2]
 [ 12 115]]

Classification Report:
              precision    recall  f1-score   support

           0       0.86      0.97      0.91        73
           1       0.98      0.91      0.94       127

    accuracy                           0.93       200
   macro avg       0.92      0.94      0.93       200
weighted avg       0.94      0.93      0.93       200
```

Gambar 1.7.1 Output Evaluasi Model Pada data Testing

Penjelasan:

- model.predict(X_test_scaled) digunakan untuk memprediksi label target pada data uji yang telah dinormalisasi.
- accuracy_score(y_test, y_pred) menghitung tingkat akurasi model, yaitu persentase prediksi yang benar dibandingkan seluruh data uji.
- confusion_matrix(y_test, y_pred) menampilkan matriks kebingungan yang memperlihatkan jumlah prediksi benar dan salah untuk tiap kelas (BELI dan TIDAK BELI).
- classification_report(y_test, y_pred, zero_division=1) memberikan laporan evaluasi yang mencakup precision, recall, f1-score, dan accuracy untuk tiap kelas target.
- Akurasi 93% → model mampu memprediksi dengan benar 93 dari 100 data uji.
- Confusion Matrix:
  - True Negatives (71): diprediksi “tidak beli” dan benar.
  - False Positives (2): diprediksi “beli” padahal sebenarnya tidak.
  - False Negatives (12): diprediksi “tidak beli” padahal sebenarnya beli.
  - True Positives (115): diprediksi “beli” dan benar.
- Classification Report:
  - *Precision (0.86 untuk kelas 0)* berarti dari semua prediksi “tidak beli”, 86% benar.
  - *Recall (0.97 untuk kelas 0)* berarti model berhasil menemukan 97% data “tidak beli” yang benar.
  - *F1-score* adalah kombinasi keduanya — semakin tinggi, semakin baik.
- Kesimpulan evaluasi: model sangat baik, dengan keseimbangan antara *precision* dan *recall* di atas 0.9.

1.8 Prediksi Dataset Baru

Kode:

```python
# Menggunakan dataset baru untuk menguji model
print("\n" + "="*50)
print("MENGGUNAKAN DATASET BARU UNTUK MENGUJI MODEL")
print("="*50)

dataset_baru = pd.DataFrame({
    'Usia': [25, 47, 33, 52, 29, 41, 36, 58, 31, 44],
    'Status': [1, 2, 1, 3, 2, 2, 1, 3, 1, 2],
    'Kelamin': [0, 1, 0, 1, 0, 1, 0, 1, 0, 1],
    'Memiliki_Mobil': [0, 1, 0, 2, 1, 1, 0, 2, 0, 1],
    'Penghasilan': [180, 220, 280, 150, 190, 260, 230, 120, 270, 200]
})
print(dataset_baru)

# Prediksi pada dataset baru
dataset_baru_scaled = scaler.transform(dataset_baru)
prediksi_baru = model.predict(dataset_baru_scaled)

# Tambahkan hasil ke dataframe
dataset_baru['Prediksi_Beli'] = prediksi_baru
dataset_baru['Keputusan'] = dataset_baru['Prediksi_Beli'].map({0: 'TIDAK BELI', 1: 'BELI'})

print("\nHasil Prediksi Dataset Baru:")
print(dataset_baru)
```

Gambar 1.8 Menggunakan Dataset Baru untuk Menguji Model

Output:

```
==================================================
MENGGUNAKAN DATASET BARU UNTUK MENGUJI MODEL
==================================================
   Usia  Status  Kelamin  Memiliki_Mobil  Penghasilan
0    25       1        0               0          180
1    47       2        1               1          220
2    33       1        0               0          280
3    52       3        1               2          150
4    29       2        0               1          190
5    41       2        1               1          260
6    36       1        0               0          230
7    58       3        1               2          120
8    31       1        0               0          270
9    44       2        1               1          200

Hasil Prediksi Dataset Baru:
   Usia  Status  Kelamin  Memiliki_Mobil  Penghasilan  Prediksi_Beli  \
0    25       1        0               0          180              0
1    47       2        1               1          220              0
2    33       1        0               0          280              1
3    52       3        1               2          150              0
4    29       2        0               1          190              0
5    41       2        1               1          260              1
6    36       1        0               0          230              1
7    58       3        1               2          120              0
8    31       1        0               0          270              1
9    44       2        1               1          200              0

    Keputusan
0  TIDAK BELI
1  TIDAK BELI
2        BELI
3  TIDAK BELI
4  TIDAK BELI
5        BELI
6        BELI
7  TIDAK BELI
8        BELI
9  TIDAK BELI
```

Gambar 1.8.1 Menggunakan Dataset Baru untuk Menguji Model

Penjelasan:

- dataset_baru = pd.DataFrame({...})
  - Membuat *data baru* berisi 10 calon pembeli mobil.
  - Setiap baris merepresentasikan 1 orang calon pembeli.
  - Kolom yang digunakan sama seperti dataset awal: Usia, Status, Kelamin, Memiliki_Mobil, dan Penghasilan.
- dataset_baru_scaled = scaler.transform(dataset_baru)
  - Menstandarkan data baru agar memiliki skala yang sama dengan data training.
  - Ini penting karena model Logistic Regression sensitif terhadap perbedaan skala antar fitur.
- prediksi_baru = model.predict(dataset_baru_scaled)
  - Model melakukan *prediksi* terhadap data baru.
  - Hasil prediksi berupa angka 0 (tidak beli) dan 1 (beli).
- dataset_baru['Prediksi_Beli'] = prediksi_baru
  - Menambahkan kolom baru Prediksi_Beli yang berisi hasil prediksi numerik dari model.
- dataset_baru['Keputusan'] = dataset_baru['Prediksi_Beli'].map({0: 'TIDAK BELI', 1: 'BELI'})
  - Mengubah nilai 0 dan 1 menjadi teks agar mudah dipahami.
  - Nilai 0 → TIDAK BELI, 1 → BELI.
- print(dataset_baru)
  - Menampilkan seluruh data baru beserta hasil prediksi.
  - Dari output terlihat, model memprediksi bahwa 4 dari 10 calon pembeli akan membeli mobil, sedangkan 6 lainnya tidak.

1.9 Statistik Prediksi Dataset Baru

Kode:

```
# Statistik Prediksi Dataset Baru
print("\n" + "="*50)
print("STATISTIK PREDIKSI DATASET BARU")
print("="*50)
print(f"Total data baru: {len(dataset_baru)}")
print(f"Diprediksi BELI (1): {sum(prediksi_baru)} data")
print(f"Diprediksi TIDAK BELI (0): {len(prediksi_baru) - sum(prediksi_baru)} data")
print(f"Persentase prediksi BELI: {sum(prediksi_baru)/len(prediksi_baru):.1%}")
```

```
==================================================
STATISTIK PREDIKSI DATASET BARU
==================================================
Total data baru: 10
Diprediksi BELI (1): 4 data
Diprediksi TIDAK BELI (0): 6 data
Persentase prediksi BELI: 40.0%
```

Gambar 1.9 Statistik Prediksi Dataset baru

Penjelasan:

- len(dataset_baru) menghitung total jumlah data baru yang akan diprediksi.
- sum(prediksi_baru) menghitung jumlah data dengan hasil prediksi 1 (BELI).
- len(prediksi_baru) - sum(prediksi_baru) menghitung jumlah data dengan hasil prediksi 0 (TIDAK BELI).

- sum(prediksi_baru)/len(prediksi_baru) menghitung persentase data yang diprediksi akan membeli mobil, lalu diformat menjadi bentuk persentase dengan dua angka di belakang koma.
- Hasil output menunjukkan bahwa:
    - Total terdapat 10 data baru yang diuji.
    - Sebanyak 4 data diprediksi akan membeli mobil (BELI).
    - Sebanyak 6 data diprediksi tidak akan membeli mobil (TIDAK BELI).
    - Persentase kemungkinan pembeli baru adalah 40%, artinya 4 dari 10 orang dalam dataset baru berpotensi menjadi pembeli berdasarkan hasil prediksi model.

2. Kesimpulan Implementasi Algoritma

Dari hasil implementasi algoritma Logistic Regression pada dataset calon pembeli mobil:

1) Model berhasil mencapai akurasi 93%, menunjukkan kemampuan klasifikasi yang tinggi.
2) Berdasarkan *confusion matrix*, model relatif seimbang dalam mengenali kelas "beli" dan "tidak beli".
3) Uji pada dataset baru menunjukkan bahwa sekitar 40% calon pembeli berpotensi membeli mobil.
4) Faktor-faktor seperti penghasilan, usia, dan kepemilikan mobil diduga memiliki pengaruh besar terhadap keputusan pembelian.
5) Logistic Regression terbukti efektif untuk permasalahan klasifikasi biner, terutama ketika target hanya memiliki dua nilai (0 dan 1).

3. Link Github

Praktikum:
https://github.com/rikaaarahma/Machine-Learning/blob/main/praktikum04/notebook/praktikum04.ipynb
Praktikum Mandiri:
https://github.com/rikaaarahma/Machine-Learning/blob/main/praktikum04/notebook/praktikum_mandiri04_Rika%20Rahma.ipynb