Laporan Praktikum Machine Learning 4

Muhammad Riandana Pranatha

0110224076

Abstrak

Pada praktikum ini dilakukan penerapan algoritma Logistic Regression untuk memprediksi dua kasus berbeda, yaitu klasifikasi status stunting anak berdasarkan data antropometri serta prediksi keputusan pembelian mobil berdasarkan atribut sosial ekonomi. Praktikum ini bertujuan untuk memahami proses pra-pemrosesan data, pembuatan model, evaluasi kinerja model, serta interpretasi hasil menggunakan metrik evaluasi seperti akurasi, precision, recall, F1-score, dan ROC-AUC.
Hasil yang diperoleh menunjukkan bahwa model Logistic Regression mampu memberikan performa yang baik dan dapat digunakan sebagai model klasifikasi biner yang efektif.

Landasan Teori

Regresi logistik merupakan metode statistik yang digunakan untuk memodelkan hubungan antara variabel independen dengan variabel dependen kategorikal (biner).
Berbeda dengan regresi linear yang menghasilkan nilai kontinu, regresi logistik menghasilkan probabilitas dari suatu kejadian dengan fungsi logit.
Algoritma ini bekerja dengan menghitung bobot (koefisien) tiap fitur dan kemudian memetakannya ke rentang probabilitas 0–1 melalui fungsi sigmoid.
Model ini umum digunakan untuk klasifikasi biner seperti deteksi penyakit, prediksi pelanggan churn, maupun analisis sosial.

Praktikum

1. 1. Mengimpor Library yang Diperlukan

Library yang digunakan antara lain pandas, numpy, matplotlib, seaborn, dan sklearn untuk analisis data, pemodelan, serta visualisasi hasil.

```python
# 1. Import Library yang Dibutuhkan
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, cross_val_score, RepeatedStratifiedKFold
from sklearn.preprocessing import StandardScaler, OneHotEncoder
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import confusion_matrix, classification_report, roc_curve, roc_auc_score, accuracy_score, precision_score, recall_score, f1_score

# Matikan warning agar output lebih rapi
import warnings
warnings.filterwarnings('ignore')

# 2. Muat Dataset (Pastikan file berada di path yang benar)
# Ganti path jika perlu
df = pd.read_csv(path + 'stunting_wasting_dataset.csv')

# Tampilkan informasi dasar data
print("Info Dataset:")
df.info()
print("\n5 Baris Pertama:")
print(df.head())
```

```
Info Dataset:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 100000 entries, 0 to 99999
Data columns (total 6 columns):
 #   Column             Non-Null Count   Dtype
---  ------             --------------   -----
 0   Jenis Kelamin      100000 non-null  object
 1   Umur (bulan)       100000 non-null  int64
 2   Tinggi Badan (cm)  100000 non-null  float64
 3   Berat Badan (kg)   100000 non-null  float64
 4   Stunting           100000 non-null  object
 5   Wasting            100000 non-null  object
dtypes: float64(2), int64(1), object(3)
memory usage: 4.6+ MB

5 Baris Pertama:
  Jenis Kelamin  Umur (bulan)  Tinggi Badan (cm)  Berat Badan (kg)  \
0     Laki-laki            19               91.6              13.3
1     Laki-laki            20               77.7               8.5
2     Laki-laki            10               79.0              10.3
3     Perempuan             2               50.3               8.3
4     Perempuan             5               56.4              10.9

           Stunting            Wasting
0              Tall  Risk of Overweight
1           Stunted         Underweight
2            Normal  Risk of Overweight
3  Severely Stunted  Risk of Overweight
4  Severely Stunted  Risk of Overweight
```

2. Membaca Dataset

Dataset utama yang digunakan adalah **stunting_wasting_dataset.csv**.
Langkah ini dilakukan dengan pd.read_csv() untuk membaca data dan memastikan data terbaca dengan benar.

```
# 2. Muat Dataset (Pastikan file berada di path yang benar)
# Ganti path jika perlu
df = pd.read_csv(path + 'stunting_wasting_dataset.csv')

# Tampilkan informasi dasar data
print("Info Dataset:")
df.info()
print("\n5 Baris Pertama:")
print(df.head())
```

```
Info Dataset:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 100000 entries, 0 to 99999
Data columns (total 6 columns):
 #   Column             Non-Null Count   Dtype
---  ------             --------------   -----
 0   Jenis Kelamin      100000 non-null  object
 1   Umur (bulan)       100000 non-null  int64
 2   Tinggi Badan (cm)  100000 non-null  float64
 3   Berat Badan (kg)   100000 non-null  float64
 4   Stunting           100000 non-null  object
 5   Wasting            100000 non-null  object
dtypes: float64(2), int64(1), object(3)
memory usage: 4.6+ MB

5 Baris Pertama:
  Jenis Kelamin  Umur (bulan)  Tinggi Badan (cm)  Berat Badan (kg)  \
0     Laki-laki            19               91.6              13.3
1     Laki-laki            20               77.7               8.5
2     Laki-laki            10               79.0              10.3
3     Perempuan             2               50.3               8.3
4     Perempuan             5               56.4              10.9

           Stunting             Wasting
0              Tall  Risk of Overweight
1           Stunted         Underweight
2            Normal  Risk of Overweight
3  Severely Stunted  Risk of Overweight
4  Severely Stunted  Risk of Overweight
```

3. Mengecek dan Membersihkan Data

Dilakukan pemeriksaan struktur data (df.info()) dan pengecekan missing value untuk memastikan semua kolom terisi lengkap.
Jika ada nilai kosong, dilakukan penanganan sesuai konteks dataset.

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 100000 entries, 0 to 99999
Data columns (total 6 columns):
 #   Column             Non-Null Count   Dtype
---  ------             --------------   -----
 0   Jenis Kelamin      100000 non-null  object
 1   Umur (bulan)       100000 non-null  int64
 2   Tinggi Badan (cm)  100000 non-null  float64
 3   Berat Badan (kg)   100000 non-null  float64
 4   Stunting           100000 non-null  object
 5   Wasting            100000 non-null  object
dtypes: float64(2), int64(1), object(3)
memory usage: 4.6+ MB
```

4. Encoding Variabel Kategorikal

Variabel seperti jenis kelamin (`JK`) dan status stunting diubah ke bentuk numerik (0 dan 1) agar dapat diproses oleh model.

```
[ ]
df.isnull().sum()
```

	0
Jenis Kelamin	0
Umur (bulan)	0
Tinggi Badan (cm)	0
Berat Badan (kg)	0
Stunting	0
Wasting	0

dtype: int64

```
df['Stunting'].unique()
df['Jenis Kelamin'].unique()
```

```
array(['Laki-laki', 'Perempuan'], dtype=object)
```

```
# 1. Mapping kolom Stunting -> biner
map_stunt = {'Stunted': 1, 'Severely Stunted': 1, 'Normal': 0, 'Tall': 0}
df['Stunting_bin'] = df['Stunting'].map(map_stunt).astype('Int64')

# 2. Mapping kolom Jenis Kelamin -> biner
#    Laki-Laki = 1, Perempuan = 0
df['JK_bin'] = (df['Jenis Kelamin'] == 'Laki-laki').astype(int)

print("Distribusi Stunting_bin:\n", df['Stunting_bin'].value_counts())
print("\nDistribusi JK_bin:\n", df['JK_bin'].value_counts())
```

```
Distribusi Stunting_bin:
 Stunting_bin
0    78021
1    21979
Name: count, dtype: Int64

Distribusi JK_bin:
 JK_bin
1    50179
0    49821
Name: count, dtype: int64
```

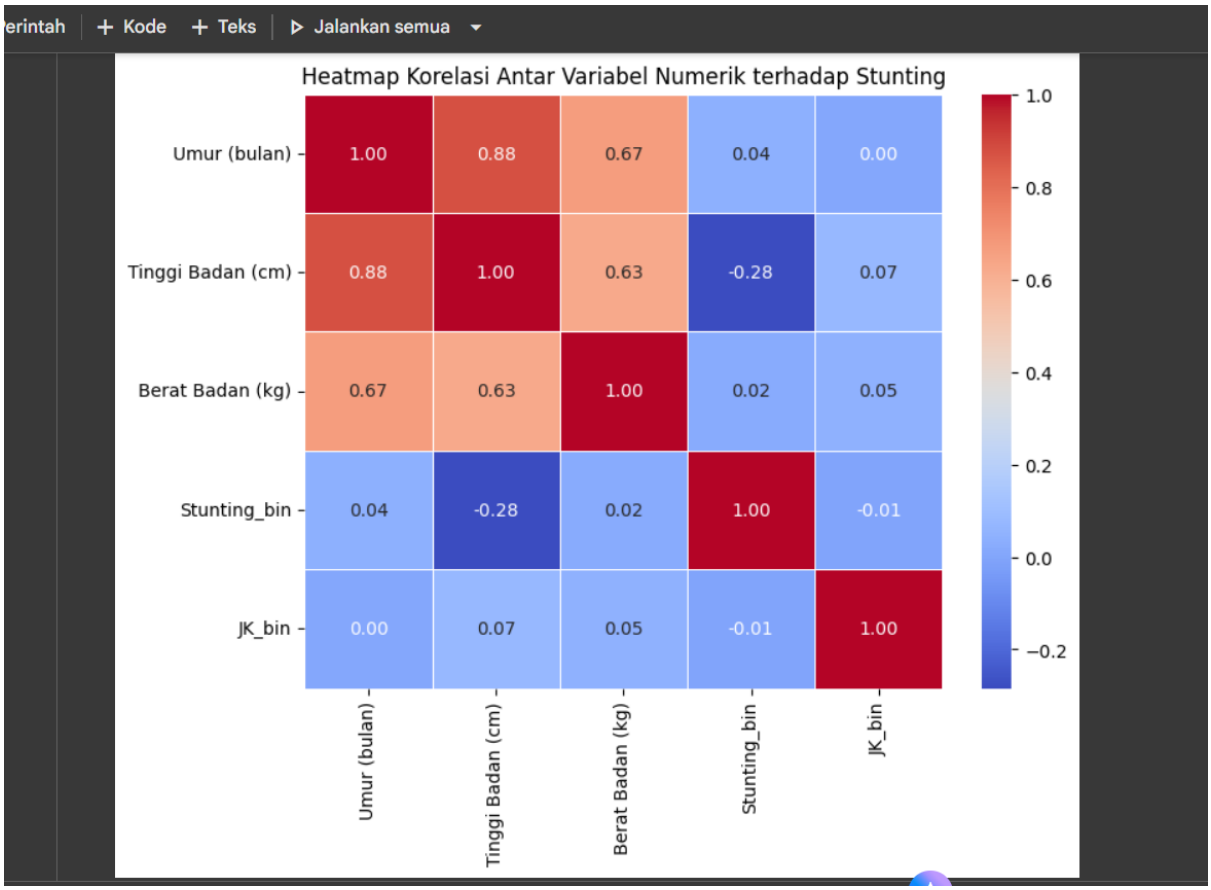
```
corr_matrix = df.corr(numeric_only=True)
corr_matrix
```

	Umur (bulan)	Tinggi Badan (cm)	Berat Badan (kg)	Stunting_bin	JK_bin
Umur (bulan)	1.000000	0.875869	0.665389	0.038630	0.004046
Tinggi Badan (cm)	0.875869	1.000000	0.626005	-0.283855	0.073505
Berat Badan (kg)	0.665389	0.626005	1.000000	0.021090	0.045797
Stunting_bin	0.038630	-0.283855	0.021090	1.000000	-0.005981
JK_bin	0.004046	0.073505	0.045797	-0.005981	1.000000

5. Analisis Korelasi dan Visual

Dilakukan analisis korelasi antar variabel numerik untuk mengetahui hubungan antar fitur menggunakan heatmap.

```
# Visualisasi heatmap
import seaborn as sns
import matplotlib.pyplot as plt

plt.figure(figsize=(8,6))
sns.heatmap(corr_matrix, annot=True, cmap='coolwarm', fmt=".2f", linewidths=0.5)
plt.title("Heatmap Korelasi Antar Variabel Numerik terhadap Stunting", fontsize=12)
plt.show()
```



6. Pembagian Data

Dataset dibagi menjadi 80% untuk training set dan 20% untuk testing set, Langkah ini menggunakan fungsi train_test_split() dari scikit-Learn

```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression

preprocess = ColumnTransformer(
    transformers=[
        ('num', StandardScaler(), feature_num),
        ('bin', 'passthrough', feature_bin)
    ],
    remainder='drop'
)

model = LogisticRegression(
    max_iter=1000,
    solver='lbfgs',
    class_weight='balanced',
    random_state=42
)

# Split data menjadi train dan test
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42
)

# Inisialisasi model
clf = LogisticRegression()

clf = Pipeline([
    ('preprocess', preprocess),
    ('model', model)
])

clf.fit(X_train, y_train)
print("✅ Model Logistic Regression berhasil dilatih.")
```

7. Pembuatan dan Pelatihan Model Logistic Regression

Model Logistic Regression diinisialisasi dengan parameter class_weight='balanced' dan max_iter=1000, kemudian dilatih menggunakan data training.

```
# Inisialisasi model
clf = LogisticRegression()

clf = Pipeline([
    ('preprocess', preprocess),
    ('model', model)
])

clf.fit(X_train, y_train)
print("✅ Model Logistic Regression berhasil dilatih.")
```

```
✅ Model Logistic Regression berhasil dilatih.
```

8. Evaluasi Model

Model diuji menggunakan data testing, lalu dihitung metrik:

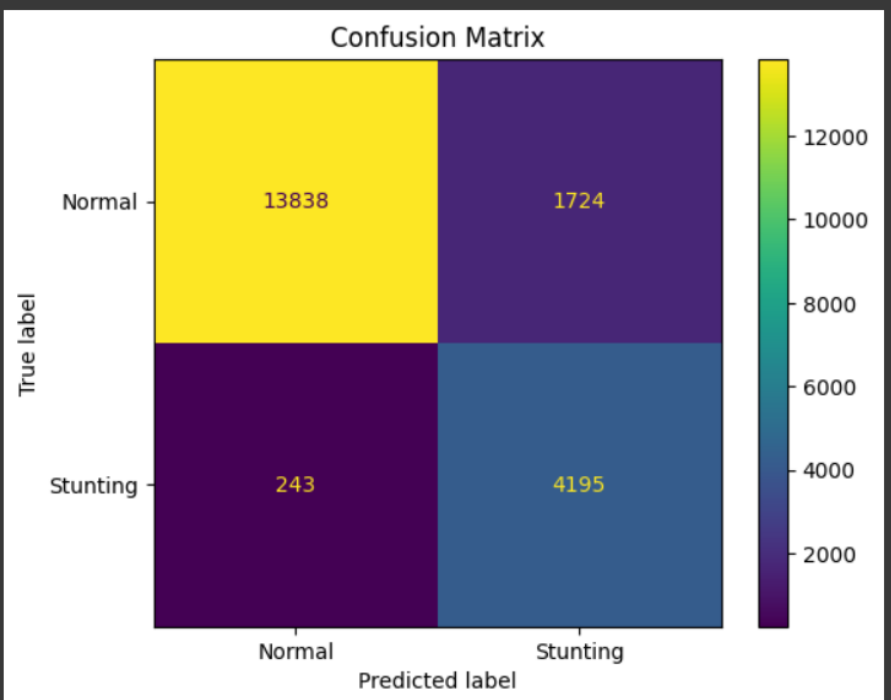
- Akurasi
- Precision
- Recall
- F1-score
- ROC-AUC

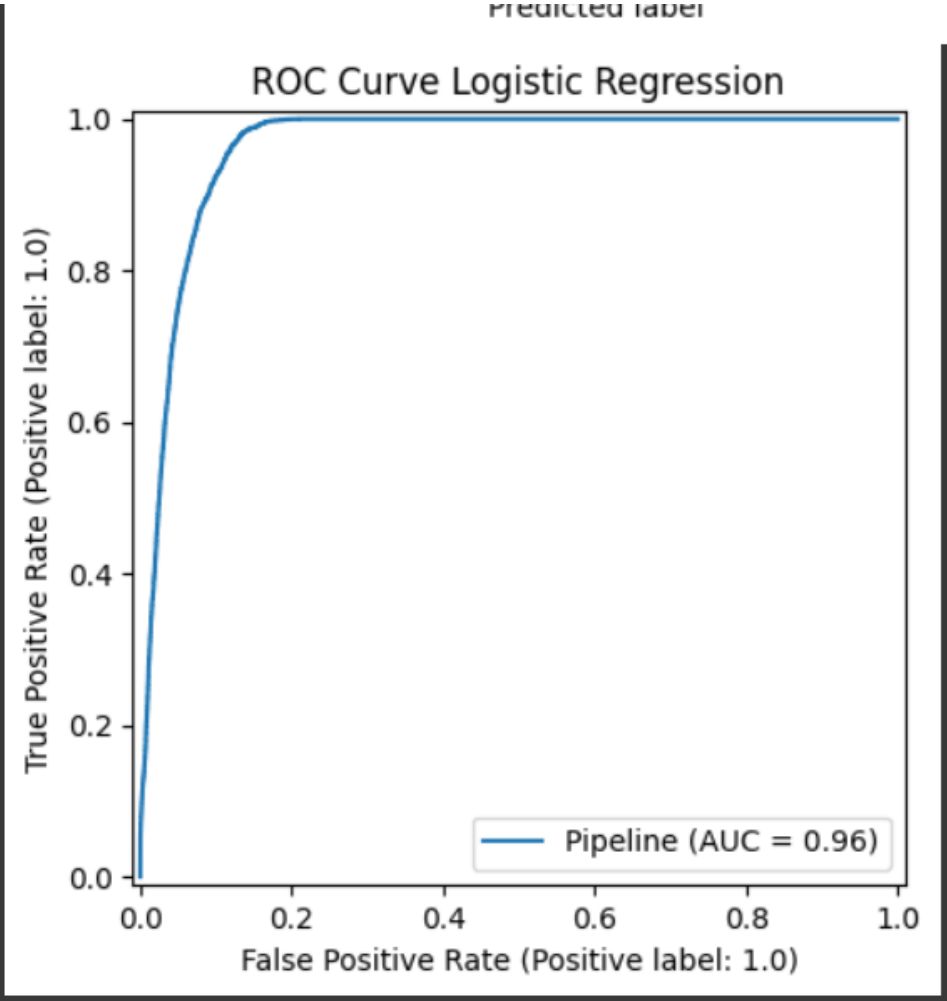
Hasil evaluasi divisualisasikan menggunakan Confusion Matrix dan ROC Curve.

```
y_pred = clf.predict(X_test)
y_prob = clf.predict_proba(X_test)[:, 1]

print(f"Akurasi   : {accuracy_score(y_test, y_pred):.4f}")
print(f"Precision : {precision_score(y_test, y_pred, zero_division=0):.4f}")
print(f"Recall    : {recall_score(y_test, y_pred, zero_division=0):.4f}")
print(f"F1-Score  : {f1_score(y_test, y_pred, zero_division=0):.4f}")
print(f"ROC-AUC   : {roc_auc_score(y_test, y_prob):.4f}")
```

```
Akurasi   : 0.9016
Precision : 0.7087
Recall    : 0.9452
F1-Score  : 0.8101
ROC-AUC   : 0.9639
```

```
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay
from sklearn.metrics import RocCurveDisplay

ConfusionMatrixDisplay(confusion_matrix(y_test, y_pred),
                       display_labels=['Normal', 'Stunting']
                       ).plot(values_format='d')
plt.title("Confusion Matrix")
plt.show()

RocCurveDisplay.from_estimator(clf, X_test, y_test)
plt.title("ROC Curve Logistic Regression")
plt.show()
```


Confusion Matrix
Normal
Stunting
True label
13838
1724
243
4195
Normal
Stunting
Predicted label
12000
10000
8000
6000
4000
2000


ROC Curve Logistic Regression
True Positive Rate (Positive label: 1.0)
False Positive Rate (Positive label: 1.0)
Pipeline (AUC = 0.96)
0.0
0.2
0.4
0.6
0.8
1.0

```
from sklearn.metrics import classification_report
print(classification_report(y_test, y_pred, target_names=['Tidak Stunting (0)', 'Stunting (1)']))
```

```
                    precision    recall  f1-score   support

Tidak Stunting (0)       0.98      0.89      0.93     15562
     Stunting (1)        0.71      0.95      0.81      4438

          accuracy                           0.90     20000
         macro avg       0.85      0.92      0.87     20000
      weighted avg       0.92      0.90      0.91     20000
```

9. Cross-Validation

Untuk memastikan model tidak hanya bagus di satu subset data, dilakukan cross-validation sebanyak 5 fold menggunakan cross_val_score().

```
]
    from sklearn.model_selection import cross_val_score

    scores = cross_val_score(clf, X, y, cv=5)

    print("Skor tiap fold:", scores)
    print("Rata-rata akurasi:", np.mean(scores))
    print("Standar deviasi:", np.std(scores))
```

```
Skor tiap fold: [0.9062  0.9013  0.9052  0.89905 0.9002 ]
Rata-rata akurasi: 0.9023899999999999
Standar deviasi: 0.0028125433329995106
```

10. Interpretasi Model (Koefisien dan Odds Ratio)

Dihitung nilai koefisien dan odds ratio untuk memahami pengaruh setiap fitur terhadap peluang terjadinya kasus stunting.

```
]
    feat_names = feature_num + feature_bin
    coefs = clf.named_steps['model'].coef_[0]
    odds = np.exp(coefs)

    coef_df = pd.DataFrame({
        'Fitur': feat_names,
        'Koefisien (log-odds)': coefs,
        'Odds Ratio (e^coef)': odds
    }).sort_values('Odds Ratio (e^coef)', ascending=False)

    display(coef_df)
```

	Fitur	Koefisien (log-odds)	Odds Ratio (e^coef)
0	Umur (bulan)	8.512751	4977.840187
3	JK_bin	1.671345	5.319315
2	Berat Badan (kg)	0.678403	1.970728
1	Tinggi Badan (cm)	-10.547296	0.000026

11. Prediksi Data Baru

Model diuji menggunakan data baru (contoh dua anak dengan atribut umur, tinggi, berat, dan jenis kelamin).
Hasil prediksi berupa probabilitas dan label (0 = Tidak Stunting, 1 = Stunting).

```
data_baru = pd.DataFrame({
    'Umur (bulan)': [24, 10],
    'Tinggi Badan (cm)': [79.0, 72.5],
    'Berat Badan (kg)': [9.2, 7.8],
    'JK_bin': [1, 0]
})

pred = clf.predict(data_baru)
prob = clf.predict_proba(data_baru)[:, 1]

hasil = data_baru.copy()
hasil['Prob_Stunting'] = prob
hasil['Pred (0=Tidak,1=Ya)'] = pred
display(hasil)
```

	Umur (bulan)	Tinggi Badan (cm)	Berat Badan (kg)	JK_bin	Prob_Stunting	Pred (0=Tidak,1=Ya)
0	24	79.0	9.2	1	0.998179	1.0
1	10	72.5	7.8	0	0.002075	0.0

12. Eksperimen Tambahan – Dataset Pembeli Mobil

Selain dataset stunting, dilakukan eksperimen kedua dengan dataset calonpembelimobil.csv.
Langkah-langkahnya meliputi:

- Membaca dataset

```
import pandas as pd

df = pd.read_csv(path + 'calonpembelimobil.csv', sep=',')

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

- Membagi data menjadi train dan test

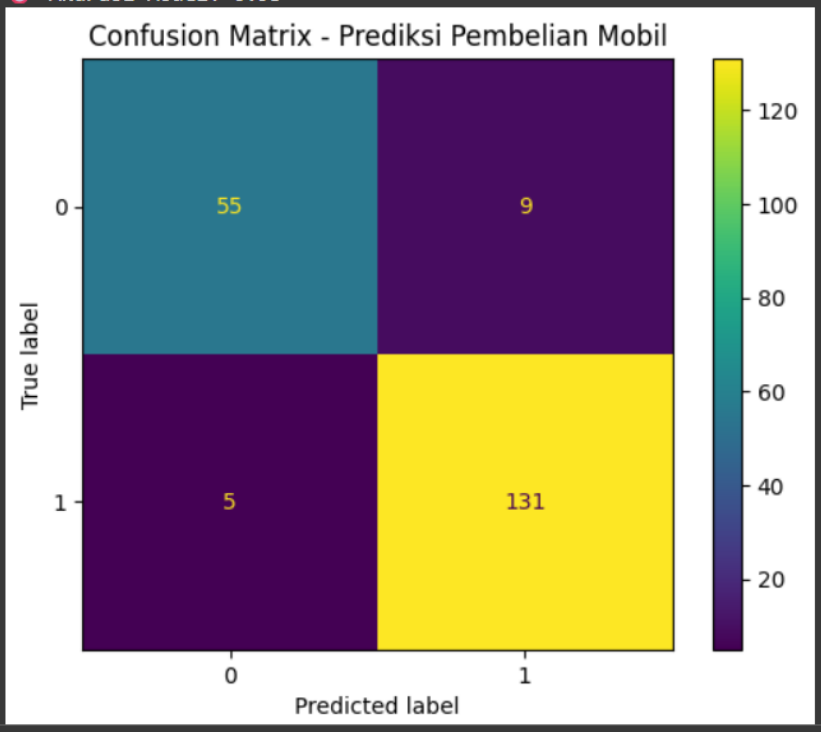
```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, confusion_matrix, ConfusionMatrixDisplay, RocCurveDisplay
import matplotlib.pyplot as plt

df = pd.read_csv(path + 'calonpembelimobil.csv', sep=',')
print("✅ Data berhasil dibaca!")
print(df.head())

X = df[['Usia', 'Status', 'Kelamin', 'Memiliki_Mobil', 'Penghasilan']]
y = df['Beli_Mobil']

X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42
)

clf = LogisticRegression(max_iter=1000)
clf.fit(X_train, y_train)
print("\n✅ Model Logistic Regression berhasil dilatih!")

y_pred = clf.predict(X_test)

akurasi = accuracy_score(y_test, y_pred)
print(f"\n🎯 Akurasi Model: {akurasi:.2f}")

ConfusionMatrixDisplay(confusion_matrix(y_test, y_pred)).plot(values_format='d')
plt.title("Confusion Matrix - Prediksi Pembelian Mobil")
plt.show()

RocCurveDisplay.from_estimator(clf, X_test, y_test)
plt.title("ROC Curve - Logistic Regression")
plt.show()
```

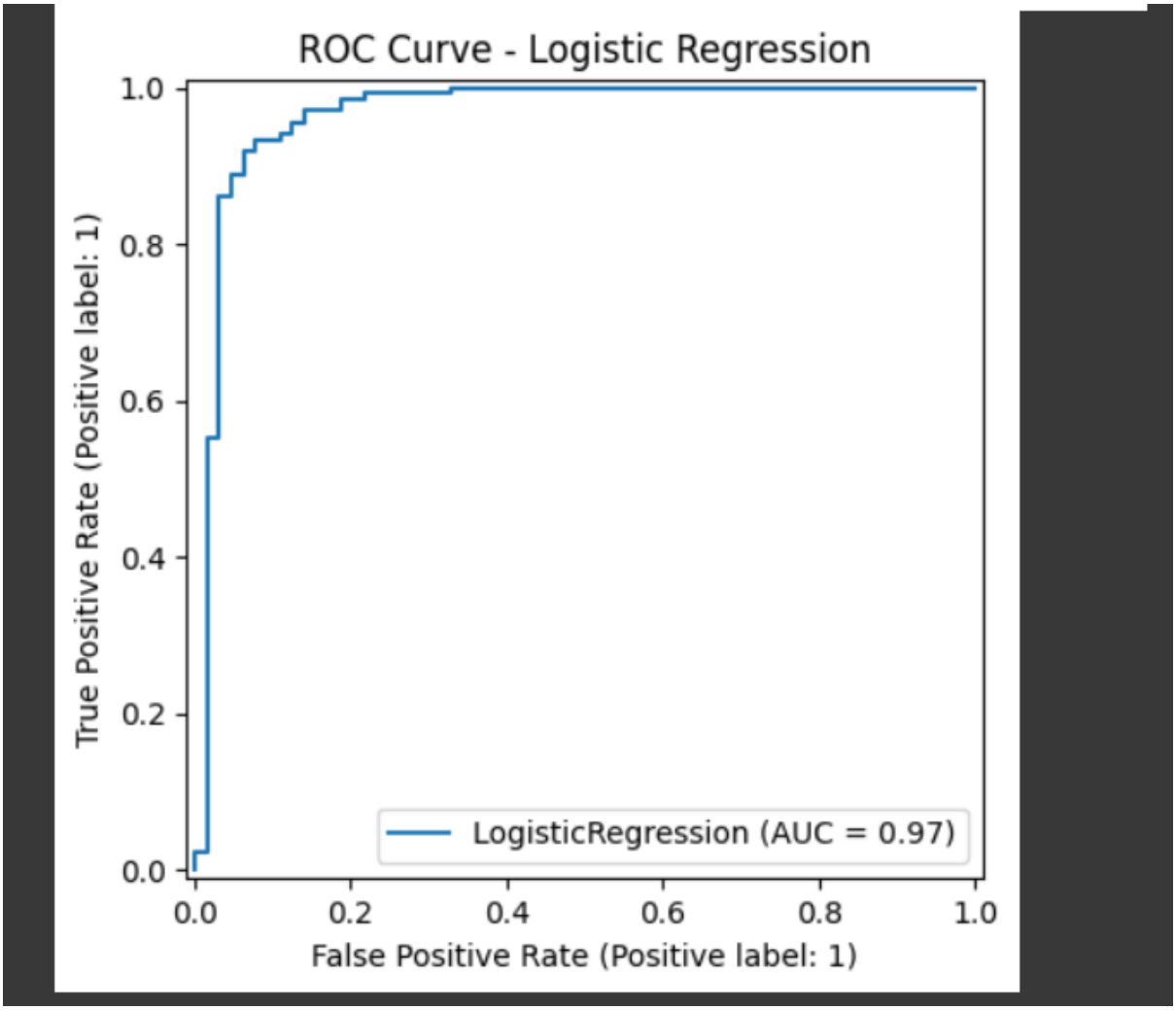
✅ Data berhasil dibaca!

	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

✅ Model Logistic Regression berhasil dilatih!

🎯 Akurasi Model: 0.93





- Melatih model Logistic Regression

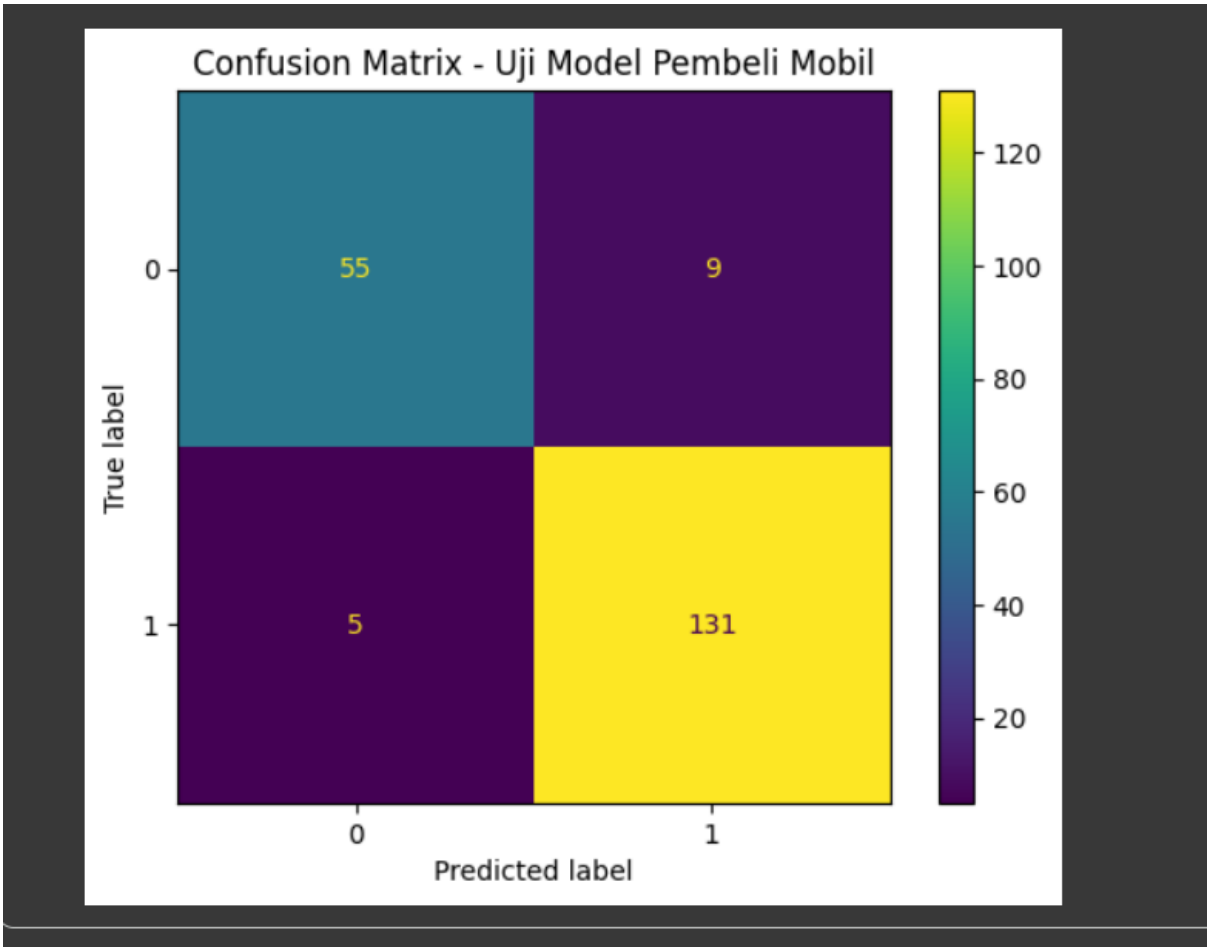
```
]
    y_pred = clf.predict(X_test)

    hasil_uji = pd.DataFrame({
        'Prediksi': y_pred,
        'Aktual': y_test.values
    })
    print(hasil_uji.head(10))
```

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

- Menghitung akurasi dan menampilkan confusion matrix serta ROC curve.

```
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix, ConfusionMatrixDisplay

print("Akurasi:", accuracy_score(y_test, y_pred))

print("\nLaporan Kinerja Model:\n")
print(classification_report(y_test, y_pred))

cm = confusion_matrix(y_test, y_pred)
ConfusionMatrixDisplay(cm).plot(values_format='d')
plt.title("Confusion Matrix - Uji Model Pembeli Mobil")
plt.show()
```

```
Akurasi: 0.93

Laporan Kinerja Model:

              precision    recall  f1-score   support

           0       0.92      0.86      0.89        64
           1       0.94      0.96      0.95       136

    accuracy                           0.93       200
   macro avg       0.93      0.91      0.92       200
weighted avg       0.93      0.93      0.93       200
```



- # Analisis Hasil
- Model Logistic Regression pada dataset stunting memberikan hasil yang sangat baik. Nilai akurasi yang tinggi (sekitar 0.90) menunjukkan kemampuan model dalam memprediksi dengan benar.
  Nilai recall yang tinggi mengindikasikan bahwa model efektif dalam mengenali kasus stunting dengan akurat, sementara precision yang sedikit lebih rendah menunjukkan masih ada beberapa prediksi salah positif.

- Kurva ROC yang mendekati nilai 1 menandakan model memiliki kemampuan klasifikasi yang sangat baik.
  Sementara pada kasus prediksi pembelian mobil, model juga menunjukkan hasil yang stabil dengan akurasi yang cukup tinggi, menandakan bahwa fitur sosial ekonomi cukup relevan terhadap keputusan pembelian mobil.

Kesimpulan

- Dari praktikum ini dapat disimpulkan bahwa Logistic Regression merupakan algoritma yang efektif untuk menyelesaikan masalah klasifikasi biner.
  Model ini memberikan hasil yang akurat, mudah diinterpretasi, dan efisien dalam komputasi.
  Dengan penerapan preprocessing dan cross-validation, model menjadi lebih stabil dan dapat diandalkan.
- Selain itu, eksperimen tambahan pada dataset pembeli mobil memperkuat pemahaman bahwa Logistic Regression bisa diterapkan di berbagai bidang, baik medis maupun ekonomi.

Link GDrive:
https://drive.google.com/drive/folders/1z4s6VxDRpkNs1RD_qpknpSDESOR8IMQ2?usp=drive_link
Link GitHub: https://github.com/Sakiaihara12/Machine-Learning.git