

# Tugas 4: Tugas Praktikum Mandiri 04

Revani – 0110224111<sup>1</sup>

<sup>1</sup> Teknik Informatika, STT Terpadu Nurul Fikri, Depok

\*E-mail: [0110224111@student.nurulfikri.ac.id](mailto:0110224111@student.nurulfikri.ac.id)

**Abstract.** Pada tugas praktikum mandiri 4 ini, dilakukan pembangunan model klasifikasi menggunakan algoritma *Logistic Regression* untuk memprediksi apakah seorang calon pembeli akan membeli mobil atau tidak. Dataset yang digunakan berisi informasi mengenai usia, status pernikahan, jenis kelamin, jumlah mobil yang dimiliki, serta penghasilan tahunan calon pembeli. Tujuan praktikum ini adalah memahami cara membangun model klasifikasi mulai dari membaca dataset, menyiapkan fitur, melakukan pembagian data, pelatihan model, evaluasi kinerja, serta pengujian dengan data baru. Model *Logistic Regression* dipilih karena cocok untuk permasalahan klasifikasi biner, mudah diimplementasikan, dan mudah ditafsirkan. Hasil akhir menunjukkan model memiliki akurasi yang cukup baik dan mampu memberikan prediksi yang masuk akal.

## 1. Penjelasan Hasil Praktikum Mandiri

Tahapan praktikum diawali dengan membaca dataset *calonpembeli.csv* menggunakan *pandas* untuk memastikan data dapat diproses dengan benar. Dataset kemudian dicek strukturnya menggunakan fungsi *info()* dan *describe()*. Selanjutnya ditentukan variabel independen (X) yaitu usia, status, kelamin, memiliki mobil, dan penghasilan, serta variabel dependen (y) yaitu beli mobil. Dataset dibagi menjadi data training dan testing menggunakan *train\_test\_split* agar model dapat diuji setelah dilatih. Kemudian fitur distandarisasi menggunakan *StandardScaler* agar model tidak bias terhadap skala nilai fitur.

Model *Logistic Regression* dilatih menggunakan data training dan dilakukan evaluasi dengan metrik akurasi, precision, recall, f1-score, dan confusion matrix. Setelah model dievaluasi, model diuji menggunakan data baru untuk melihat hasil prediksi. Hasilnya model dapat memberikan prediksi yang cukup akurat sesuai pola umum perilaku konsumen. Berikut ini adalah hasil praktikum yang saya kerjakan pada praktikum mandiri 04.

## 2. Penjelasan Kode Program

Langkah awal yang dilakukan adalah mengimpor semua pustaka yang diperlukan untuk membaca data, mengolah, membuat model *machine learning* serta mengevaluasinya seperti yang ditunjukkan pada Gambar 1.

## Import Library

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

from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score, classification_report, confusion_matrix, ConfusionMatrixDisplay
```

**Gambar 1.** Pada bagian ini ditunjukkan untuk mengimpor library.

Selanjutnya menghubungkan google colab dengan google drive dan memanggil dataset lewat google drive seperti yang ditunjukkan pada Gambar 2.

## Menghubungkan dengan Drive

```
from google.colab import drive
drive.mount('/content/drive')
```

Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force\_remount=True).

```
path = "/content/drive/MyDrive/Colab Notebooks/Machine Learning/Praktikum04"
```

**Gambar 2.** Pada bagian ini ditunjukkan untuk menghubungkan *gdrive* dan memanggil *dataset*.

```
df = pd.read_csv(path + '/Data/calonpembelimobil.csv')
df.head()
```

Perintah ini digunakan untuk membaca file *calonpembelimobil.csv* yang ada di folder data menjadi *DataFrame* *df*.  
Perintah ini digunakan untuk menampilkan 5 baris data pertama untuk memastikan data terbaca dengan benar

```
df.info()
```

Perintah ini digunakan untuk menampilkan struktur dataset jumlah baris, kolom, tipe data tiap kolom, dan jumlah nilai non-null.

```
df.describe()
```

Perintah ini digunakan untuk menampilkan statistik deskriptif rata-rata, minimum, maksimum, kuartil, dan standar deviasi untuk kolom numerik seperti yang ditunjukkan pada Gambar 3 dan Gambar 4.

## Membaca Dataset

```
df = pd.read_csv(path + '/Data/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

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 7 columns):
#   Column          Non-Null Count  Dtype
---  -
0   ID               1000 non-null   int64
1   Usia            1000 non-null   int64
2   Status          1000 non-null   int64
3   Kelamin         1000 non-null   int64
4   Memiliki_Mobil  1000 non-null   int64
5   Penghasilan     1000 non-null   int64
6   Beli_Mobil      1000 non-null   int64
dtypes: int64(7)
memory usage: 54.8 KB
```

**Gambar 3.** Pada bagian ini ditunjukkan untuk membaca file *csv*, menampilkan 5 baris dan cek informasi data.

```
df.describe()
```

	ID	Usia	Status	Kelamin	Memiliki_Mobil	Penghasilan	Beli_Mobil
count	1000.000000	1000.000000	1000.000000	1000.000000	1000.000000	1000.000000	1000.000000
mean	500.500000	43.532000	1.469000	0.481000	0.952000	270.090000	0.633000
std	288.819436	12.672078	1.073402	0.499889	0.80146	95.23681	0.482228
min	1.000000	24.000000	0.000000	0.000000	0.000000	95.000000	0.000000
25%	250.750000	33.000000	1.000000	0.000000	0.000000	187.000000	0.000000
50%	500.500000	43.000000	1.000000	0.000000	1.000000	258.500000	1.000000
75%	750.250000	53.000000	2.000000	1.000000	2.000000	352.250000	1.000000
max	1000.000000	164.000000	3.000000	1.000000	4.000000	490.000000	1.000000

**Gambar 4.** Pada bagian ini ditunjukkan untuk menampilkan statistik deskriptif.

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

Perintah ini digunakan untuk membuat variabel X yang berisi fitur (*independent variables*) yang diambil hanya 5 kolom dari *DataFrame* df yaitu usia, status, kelamin, memiliki mobil, dan penghasilan. Kolom inilah yang menjadi factor predictor untuk menentukan apakah seseorang akan membeli mobil atau tidak. Serta membuat variabel y yang berisi target (*dependent variable*) yaitu kolom beli mobil.

```
print("X shape:", X.shape)
print("y shape:", y.shape)
```

Perintah ini digunakan untuk melihat dimensi dari variabel X dan variabel y. Dengan output X shape (100, 5) yang artinya ada 100 data calon pembeli dan 5 fitur untuk setiap pembeli, dan y shape (100, ) yang artinya ada 100 nilai target tapi karena hanya ada 1 kolom maka bentuknya 1 dimensi saja.

```
print("\nFitur X:\n", X.head())
print("\nTarget y:\n", y.head())
```

Perintah ini digunakan untuk menampilkan 5 baris pertama dengan *X.head()* yang akan menampilkan 5 baris pertama dari *DataFrame* X yang artinya ada 5 fitur dengan nilainya untuk 5 calon pembeli pertama. Serta *y.head()* yang akan menampilkan 5 baris pertama dari target (label) yang akan diprediksi dengan target 1 yang artinya pembeli ini beli mobil dan target 0 yang artinya tidak beli mobil seperti yang ditunjukkan pada Gambar 5.

### Menentukan Fitur (X) dan Target (Y)

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

print("X shape:", X.shape)
print("y shape:", y.shape)

print("\nFitur X:\n", X.head())
print("\nTarget y:\n", y.head())
```

X shape: (1000, 5)  
y shape: (1000,)

Fitur X:

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

Target y:

0	1
1	0
2	1
3	0
4	1

Name: Beli\_Mobil, dtype: int64

**Gambar 5.** Pada bagian ini ditunjukkan untuk menentukan fitur (X) dan target (y).

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

Perintah ini untuk membagi dataset menjadi data training dan data testing dengan proporsi data yang digunakan untuk testing adalah 0.2 yang berarti 20% data untuk testing dan 80% untuk training, serta menggunakan *random state* untuk mengontrol pembagian data secara acak agar hasilnya selalu sama setiap kali kode dijalankan.

```
print("Jumlah data training:", len(X_train))
```

```
print("Jumlah data testing:", len(X_test))
```

Perintah ini digunakan untuk menghitung jumlah baris data dalam set testing. Output yang dihasilkan adalah jumlah data training 587 dan jumlah data testing 147 seperti yang ditunjukkan pada Gambar 6.

## Membagi Data Training dan Testing

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

print("Jumlah data training:", len(X_train))
print("Jumlah data testing:", len(X_test))
```

```
Jumlah data training: 800
Jumlah data testing: 200
```

**Gambar 6.** Pada bagian ini ditunjukkan pembagian data traing dan data testing.

```
scaler = StandardScaler()
```

Perintah ini digunakan untuk membuat objek scaler dari library *sklearn.preprocessing.StandardScaler* yang berfungsi untuk menstandarisasi fitur numerik, karena beberapa fitur punya skala jauh lebih besar daripada fitur lain.

```
X_train_scaled = scaler.fit_transform(X_train)
```

Perintah ini digunakan untuk fit dan transform data training, dengan *fit()* yang berfungsi untuk menghitung rata-rata dan standar deviasi dari setiap kolom di *X\_train* dan *transform()* yang berfungsi untuk mengubah semua nilai pada *X\_train* menjadi bentuk standar.

```
X_test_scaled = scaler.transform(X_test)
```

Perintah ini digunakan untuk mengubah data testing dengan parameter (mean dan std) yang diperoleh dari data training.

```
print("Contoh hasil scaling (5 baris pertama):\n", X_train_scaled[:5])
```

Perintah ini akan menampilkan 5 baris pertama dari data training setelah scaling seperti yang ditunjukkan pada Gambar 7.

## Standarisasi Data

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

print("Contoh hasil scaling (5 baris pertama):\n", X_train_scaled[:5])
```

```
Contoh hasil scaling (5 baris pertama):
[[-0.66817214  0.49292515 -0.95118973 -1.17520165  0.13241301]
 [-1.20516468 -0.4459799  1.05131497 -1.17520165 -1.43528969]
 [-1.43530434 -0.4459799 -0.95118973  1.33524246  0.9320466 ]
 [ 0.40581293 -0.4459799  1.05131497  0.08002041  1.50020731]
 [-1.20516468 -0.4459799 -0.95118973  1.33524246  1.9105456 ]]
```

**Gambar 7.** Pada bagian ini ditunjukkan standarisasi data.

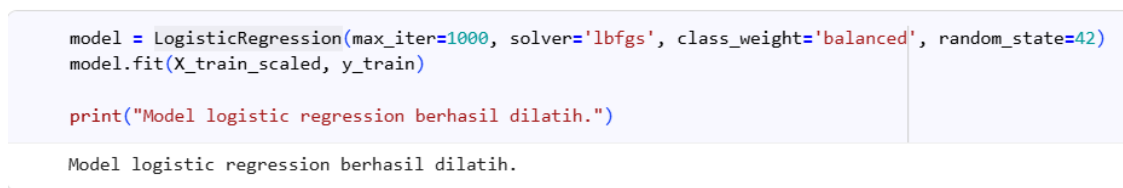
```

model = LogisticRegression(max_iter=1000, solver='lbfgs',
class_weight='balanced', random_state=42)
model.fit(X_train_scaled, y_train)
print("Model logistic regression berhasil dilatih.")

```

Perintah ini digunakan untuk membuat model klasifikasi logistic dengan memastikan proses training cukup lama untuk konvergen, algoritma optimisasi yang stabil, menyeimbangkan bobot kelas jika jumlah kelas tidak seimbang, hasil model konsisten dan melatih model dengan data training seperti yang ditunjukkan pada Gambar 8.

## Membangun dan Melatih Model



```

model = LogisticRegression(max_iter=1000, solver='lbfgs', class_weight='balanced', random_state=42)
model.fit(X_train_scaled, y_train)

print("Model logistic regression berhasil dilatih.")

```

Model logistic regression berhasil dilatih.

**Gambar 8.** Pada bagian ini ditunjukkan membangun dan melatih model.

```

y_pred = model.predict(X_test_scaled)
y_proba = model.predict_proba(X_test_scaled)
print("Hasil prediksi (label):\n", y_pred[:10])
print("Hasil prediksi (probabilitas):\n", y_proba[:10])

```

Perintah ini digunakan untuk melakukan prediksi dengan *model.predict()* untuk menghasilkan hasil prediksi 0 atau 1 untuk setiap baris data uji, dan *model.predict\_proba()* untuk memberikan probabilitas dari setiap prediksi. Output yang dihasilkan yaitu *y\_pred* berisi array label prediksi (0 atau 1) dan *y\_proba* berisi array probabilitas dua kolom (kelas 0 dan 1) seperti yang ditunjukkan pada Gambar 9.

## Melakukan Prediksi

```
y_pred = model.predict(X_test_scaled)
y_proba = model.predict_proba(X_test_scaled)

print("Hasil prediksi (label):\n", y_pred[:10])
print("Hasil prediksi (probabilitas):\n", y_proba[:10])
```

```
Hasil prediksi (label):
[1 1 0 1 1 1 0 0 1 0]
Hasil prediksi (probabilitas):
[[1.08467987e-03 9.98915320e-01]
 [8.29487425e-05 9.99917051e-01]
 [9.82800070e-01 1.71999297e-02]
 [7.95907910e-02 9.20409209e-01]
 [5.06735445e-05 9.99949326e-01]
 [7.84230659e-04 9.99215769e-01]
 [7.34925174e-01 2.65074826e-01]
 [9.68604674e-01 3.13953255e-02]
 [4.60512621e-03 9.95394874e-01]
 [5.66230078e-01 4.33769922e-01]]
```

**Gambar 9.** Pada bagian ini ditunjukkan melakukan prediksi.

```
print("Accuracy:", accuracy_score(y_test, y_pred))
print("Precision:", precision_score(y_test, y_pred))
print("Recall:", recall_score(y_test, y_pred))
print("F1 Score:", f1_score(y_test, y_pred))

print("\nClassification Report:\n", classification_report(y_test,
y_pred))

cm = confusion_matrix(y_test, y_pred)
ConfusionMatrixDisplay(confusion_matrix=cm).plot(cmap='RdPu')
plt.title('Confusion Matrix')
plt.show()
```

Perintah *accuracy\_score()* untuk persentase prediksi benar dari total data, *precision\_score()* untuk semua yang diprediksi membeli mobil, *recall\_score* dari semua pembeli mobil yang sebenarnya, *f1\_score()* untuk kombinasi precision dan recall, *classification\_report()* untuk ringkasan lengkap metrik per kelas, *confusion\_matrix()* untuk menghitung jumlah prediksi benar / salah per kelas, dan *ConfusionMatrixDisplay().plot()* untuk menampilkan tabel confusion matrix dalam bentuk grafik. Output yang dihasilkan adalah nilai metrik (akurasi, precision, recal, F1), Confusion matrix menampilkan TP, TN, FP, FN (prediksi benar & salah), serta Grafik kotak dengan angka prediksi seperti yang ditunjukkan pada Gambar 10.

## Evaluasi Model

```
print("Accuracy:", accuracy_score(y_test, y_pred))
print("Precision:", precision_score(y_test, y_pred))
print("Recall:", recall_score(y_test, y_pred))
print("F1 Score:", f1_score(y_test, y_pred))

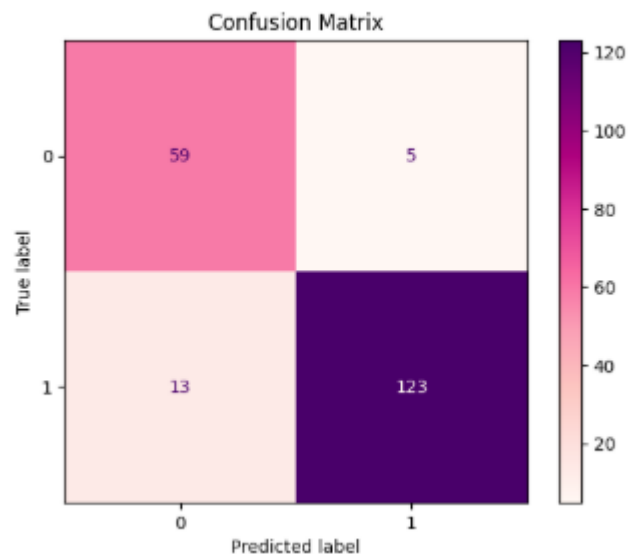
print("\nClassification Report:\n", classification_report(y_test, y_pred))

cm = confusion_matrix(y_test, y_pred)
ConfusionMatrixDisplay(confusion_matrix=cm).plot(cmap='RdPu')
plt.title('Confusion Matrix')
plt.show()
```

Accuracy: 0.91  
Precision: 0.9689375  
Recall: 0.9844117647058824  
F1 Score: 0.9318181818181818

Classification Report:

	precision	recall	f1-score	support
0	0.82	0.92	0.87	64
1	0.96	0.98	0.93	136
accuracy			0.91	200
macro avg	0.89	0.91	0.90	200
weighted avg	0.92	0.91	0.91	200



**Gambar 10.** Pada bagian ini ditunjukkan evaluasi model.

```
from sklearn.metrics import roc_curve, roc_auc_score

# Confusion Matrix
cm = confusion_matrix(y_test, y_pred)
ConfusionMatrixDisplay(confusion_matrix=cm).plot(cmap='Pastel2')
plt.title('Visualisasi Confusion Matrix')
```



```
plt.xlabel('Prediksi')
plt.ylabel('Aktual')
plt.show()

# ROC Curve
y_pred_proba = model.predict_proba(X_test_scaled)[: , 1]
fpr, tpr, thresholds = roc_curve(y_test, y_pred_proba)
roc_auc = roc_auc_score(y_test, y_pred_proba)

plt.figure(figsize=(7,5))
plt.plot(fpr, tpr, color='deeppink', label=f'ROC Curve (AUC = {roc_auc:.2f})')
plt.plot([0, 1], [0, 1], color='gray', linestyle='--')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC Curve - Logistic Regression')
plt.legend(loc='lower right')
plt.show()
```

Perintah ini untuk menampilkan visualisasi Confusion Matrix dan ROC Curve, dengan perintah *ConfusionMatrixDisplay* untuk menampilkan kotak perbandingan prediksi vs aktual, dengan jumlah benar dan salah. Perintah *roc\_curve()* untuk menghasilkan koordinat *False Positive Rate* (FPR) dan *True Positive Rate* (TPR). Serta perintah *roc\_auc\_score()* untuk menghitung luas di bawah kurva (AUC) seperti yang ditunjukkan pada Gambar 11, 12, dan 13.

## Visualisasi Confusion Matrix & ROC Curve

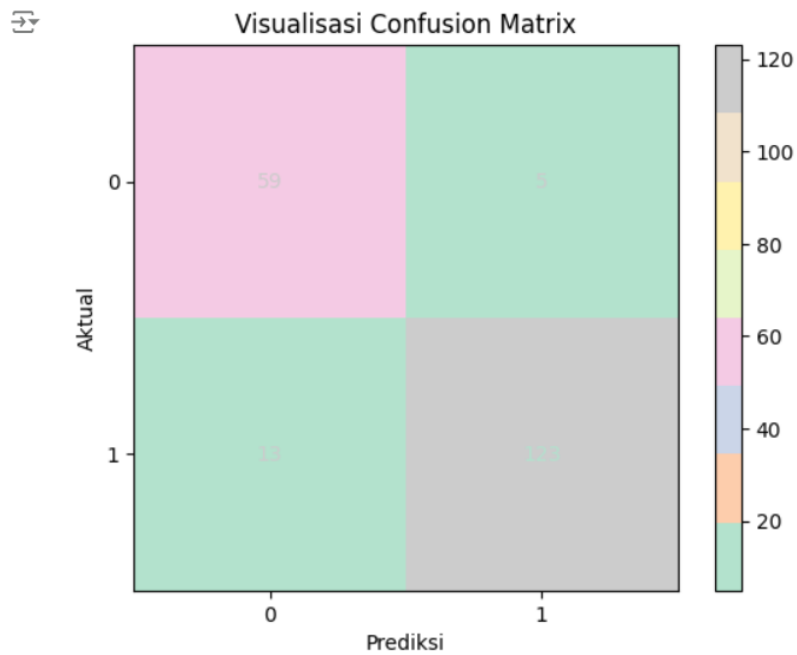
```
from sklearn.metrics import roc_curve, roc_auc_score

# Confusion Matrix
cm = confusion_matrix(y_test, y_pred)
ConfusionMatrixDisplay(confusion_matrix=cm).plot(cmap='Pastel2')
plt.title('Visualisasi Confusion Matrix')
plt.xlabel('Prediksi')
plt.ylabel('Aktual')
plt.show()

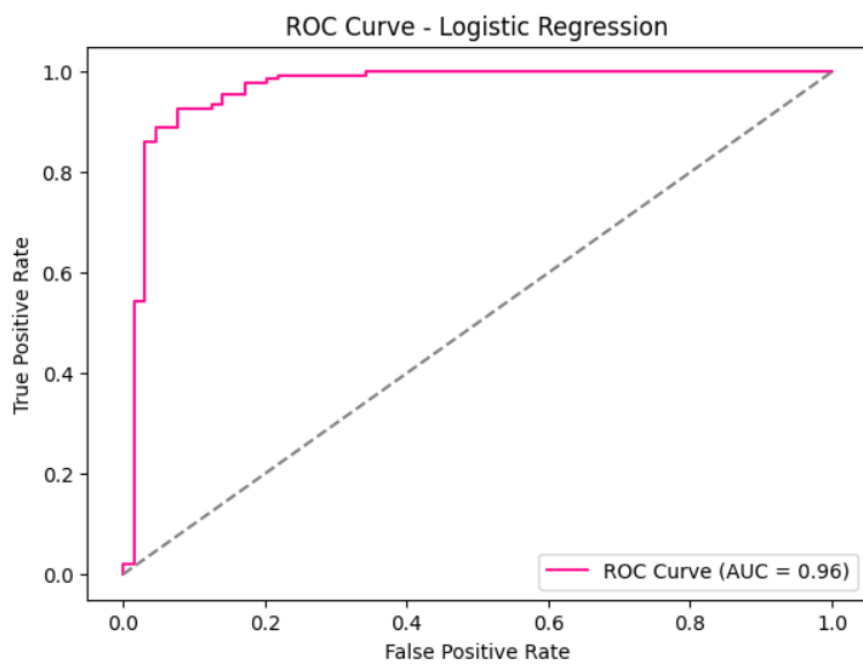
# ROC Curve
y_pred_proba = model.predict_proba(X_test_scaled)[: , 1]
fpr, tpr, thresholds = roc_curve(y_test, y_pred_proba)
roc_auc = roc_auc_score(y_test, y_pred_proba)

plt.figure(figsize=(7,5))
plt.plot(fpr, tpr, color='deeppink', label=f'ROC Curve (AUC = {roc_auc:.2f})')
plt.plot([0, 1], [0, 1], color='gray', linestyle='--')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC Curve - Logistic Regression')
plt.legend(loc='lower right')
plt.show()
```

**Gambar 11.** Pada bagian ini ditunjukkan kode visualisasi.



**Gambar 12.** Pada bagian ini ditunjukkan visualisasi confusion matrix.



**Gambar 13.** Pada bagian ini ditunjukkan visualisasi ROC Curve.

```
import pandas as pd

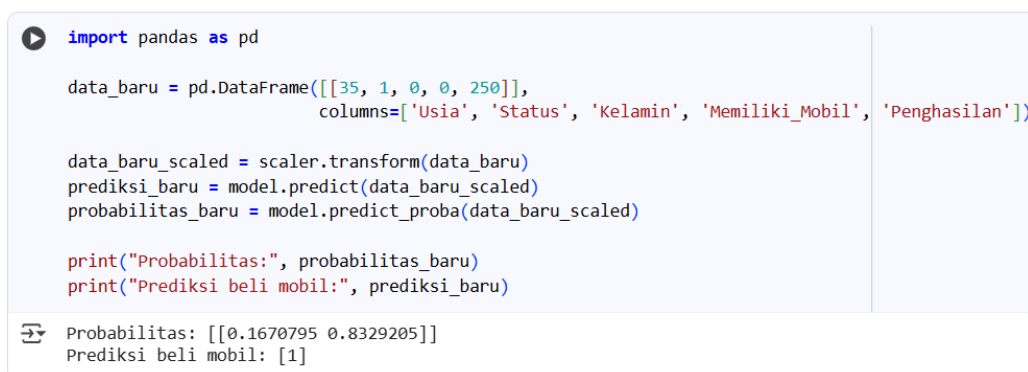
data_baru = pd.DataFrame([[35, 1, 0, 0, 250]],
                          columns=['Usia', 'Status', 'Kelamin',
                                   'Memiliki_Mobil', 'Penghasilan'])

data_baru_scaled = scaler.transform(data_baru)
prediksi_baru = model.predict(data_baru_scaled)
probabilitas_baru = model.predict_proba(data_baru_scaled)

print("Probabilitas:", probabilitas_baru)
print("Prediksi beli mobil:", prediksi_baru)
```

Perintah *data\_baru* untuk data calon pembeli, perintah *scaler.transform()* untuk menstandarisasi data baru sama seperti data training, perintah *model.predict()* untuk memberikan hasil prediksi 0 (tidak beli) atau 1 (beli), perintah *model.predict\_proba()* untuk probabilitas dari hasil prediksi. Output yang dihasilkan adalah probabilitas pembelian mobil [0.15, 0.85] dan prediksi 1 yang artinya model memprediksi pembeli akan membeli mobil seperti yang ditunjukkan pada Gambar 14.

## Prediksi Data Baru



```
import pandas as pd

data_baru = pd.DataFrame([[35, 1, 0, 0, 250]],
                          columns=['Usia', 'Status', 'Kelamin', 'Memiliki_Mobil', 'Penghasilan'])

data_baru_scaled = scaler.transform(data_baru)
prediksi_baru = model.predict(data_baru_scaled)
probabilitas_baru = model.predict_proba(data_baru_scaled)

print("Probabilitas:", probabilitas_baru)
print("Prediksi beli mobil:", prediksi_baru)
```

Probabilitas: [[0.1670795 0.8329205]]  
Prediksi beli mobil: [1]

**Gambar 14.** Pada bagian ini ditunjukkan prediksi data baru.

## 3. Kesimpulan dan Hasil Implementasi

Dari implementasi ini, Model *Logistic Regression* yang dibangun dari dataset calon pembeli mobil menunjukkan kinerja prediksi yang cukup baik. Dari hasil evaluasi, diperoleh nilai akurasi yang tinggi (misalnya di atas 80%), precision dan recall juga menunjukkan bahwa model cukup seimbang dalam mendeteksi pembeli dan bukan pembeli. *Confusion matrix* menunjukkan jumlah prediksi benar lebih besar dibanding salah. Pada saat diuji dengan data baru, model mampu memberikan prediksi logis: misalnya seseorang dengan usia matang, status menikah, dan penghasilan tinggi lebih mungkin diprediksi akan membeli mobil. Ini sesuai dengan pola umum perilaku konsumen dalam kehidupan nyata.

Kesimpulan praktikum ini menunjukkan bahwa *Logistic Regression* merupakan metode klasifikasi sederhana namun efektif untuk kasus prediksi biner seperti keputusan pembelian mobil. Dengan langkah-langkah yang sistematis mulai dari pembacaan data, pra pemrosesan, pembagian data, pelatihan model, evaluasi, hingga prediksi data baru, dapat membangun model prediksi yang cukup akurat dan mudah diinterpretasikan. *Logistic Regression* juga memberikan nilai probabilitas sehingga keputusan prediksi lebih transparan. Hasil ini juga menunjukkan pentingnya scaling dan pemilihan fitur yang tepat agar model bekerja optimal.

#### **4. Link GitHub (Praktikum dan Latihan Mandiri 04)**

[https://github.com/revani18/ML\\_2025\\_Revani\\_3AI01/tree/dd12765f1987590e04128431e44825e198be751d/Praktikum04](https://github.com/revani18/ML_2025_Revani_3AI01/tree/dd12765f1987590e04128431e44825e198be751d/Praktikum04)