# Tugas 4: Praktikum Mandiri 4

**Fatih Dzakwan Susilo - 0110224235**

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

*E-mail: fatihdzakwansusilo@gmail.com

## 1. Praktikum Mandiri 4

Link Github : **https://github.com/FatihDzkwn/Project_Machine_Learning.git**

*1.1 Import library*

```
[110]
✓ 0d
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

from sklearn.model_selection import train_test_split
from sklearn.compose import ColumnTransformer
from sklearn.preprocessing import OneHotEncoder, StandardScaler
from sklearn.pipeline import Pipeline
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import (
accuracy_score, precision_score, recall_score, f1_score, roc_auc_score,
confusion_matrix, classification_report, RocCurveDisplay, ConfusionMatrixDisplay
)
```

**Gambar 1.** Import library.

Pada tahap ini dilakukan proses import pustaka (library) yang dibutuhkan untuk membangun model Logistic Regression. Library seperti pandas dan numpy digunakan untuk mengelola dan memproses data, matplotlib dan seaborn untuk visualisasi data, serta scikit-learn untuk membangun, melatih, dan mengevaluasi model machine learning. Tahapan ini penting karena setiap library memiliki fungsi khusus yang mendukung keseluruhan proses analisis data hingga pembuatan model prediksi.

*1.2 Membaca data file CSV*

[111]
✓ 0 d

```
PATH = "/content/gdrive/MyDrive/Project_Machine_Learning/Praktikum_Mandiri_4/Data/"

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 |

Langkah berikutnya: Buat kode dengan df | New interactive sheet

**Gambar 2.** Membaca data file CSV.

Pada tahap ini, dataset calon pembeli mobil dibaca menggunakan fungsi read_csv() dari library pandas. Langkah ini bertujuan untuk memuat data dari file berformat .csv ke dalam bentuk DataFrame agar lebih mudah dianalisis dan diolah. Dengan format tabel yang terstruktur, setiap kolom merepresentasikan variabel seperti usia, penghasilan, status, dan kepemilikan mobil, sedangkan setiap baris menunjukkan data individu calon pembeli.

*1.3 Melihat informasi umum dataset*

[112]
✓ 0 d

```
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.** Melihat informasi umum dataset.

Langkah ini bertujuan untuk memahami gambaran awal dari dataset yang digunakan. Biasanya dilakukan dengan fungsi seperti info() di Python (Pandas) untuk melihat jumlah baris dan kolom, tipe data setiap kolom, serta jumlah nilai yang hilang (missing values). Dengan informasi ini, kita

dapat mengetahui apakah data sudah lengkap, tipe data sudah sesuai, dan apakah diperlukan pembersihan data sebelum analisis lebih lanjut.

*1.4 Data pre-processing*

[113] ✓ 0 d
```
# Cek missing value
df.isnull().sum()
```

| | 0 |
|---|---|
| ID | 0 |
| Usia | 0 |
| Status | 0 |
| Kelamin | 0 |
| Memiliki_Mobil | 0 |
| Penghasilan | 0 |
| Beli_Mobil | 0 |

dtype: int64

[114] ✓ 0 d
```
# Cek duplikat data
df.duplicated().sum()
```
```
np.int64(0)
```

[115] ✓ 0 d
```
# Menghapus data duplikat (apabila ada)
# df.drop_duplicates(inplace=True)
```

[116] ✓ 0 d
```
# Cek nilai unik
df["Kelamin"].unique()
# Jenis kelamin calon pembeli (0=pria, 1=wanita)
```
```
array([0, 1])
```

[117] ✓ 0 d
```
df["Status"].unique()
# Status pernikahan calon pembeli (0=single, 1=menikah, 2=menikah mempunyai anak, 3=duda/janda)
```
```
array([1, 2, 3, 0])
```

[118] ✓ 0 d
```
df["Beli_Mobil"].unique()
# Apakah calon pembeli mobil benar benar membeli mobil atau tidak (0=tidak membeli, 1=membeli mobil)
```
```
array([1, 0])
```

**Gambar 4.** Data pre-processing.

[119]
0d

```
#Mapping kolom kategori kebentuk numerik (Karena semua kolom sudah numerik, tidak perlu mengubahnya lagi

#Melihat distribusi (jumlah kemunculan) dari masing-masing kategori atau kelas disemua kolom
for col in df.columns:
    print(f"\nDistribusi {col}:")
    print(df[col].value_counts())
```

```
Distribusi ID:
ID
1000    1
1       1
2       1
3       1
4       1
       ..
13      1
12      1
11      1
10      1
9       1
Name: count, Length: 1000, dtype: int64

Distribusi Usia:
Usia
43     32
26     31
31     31
38     31
30     30
51     30
28     30
32     29
35     29
55     28
52     27
49     27
34     27
37     27
29     26
45     26
40     25
44     25
25     24
```

**Gambar 5.** Data pre-processing.

[120]
0d

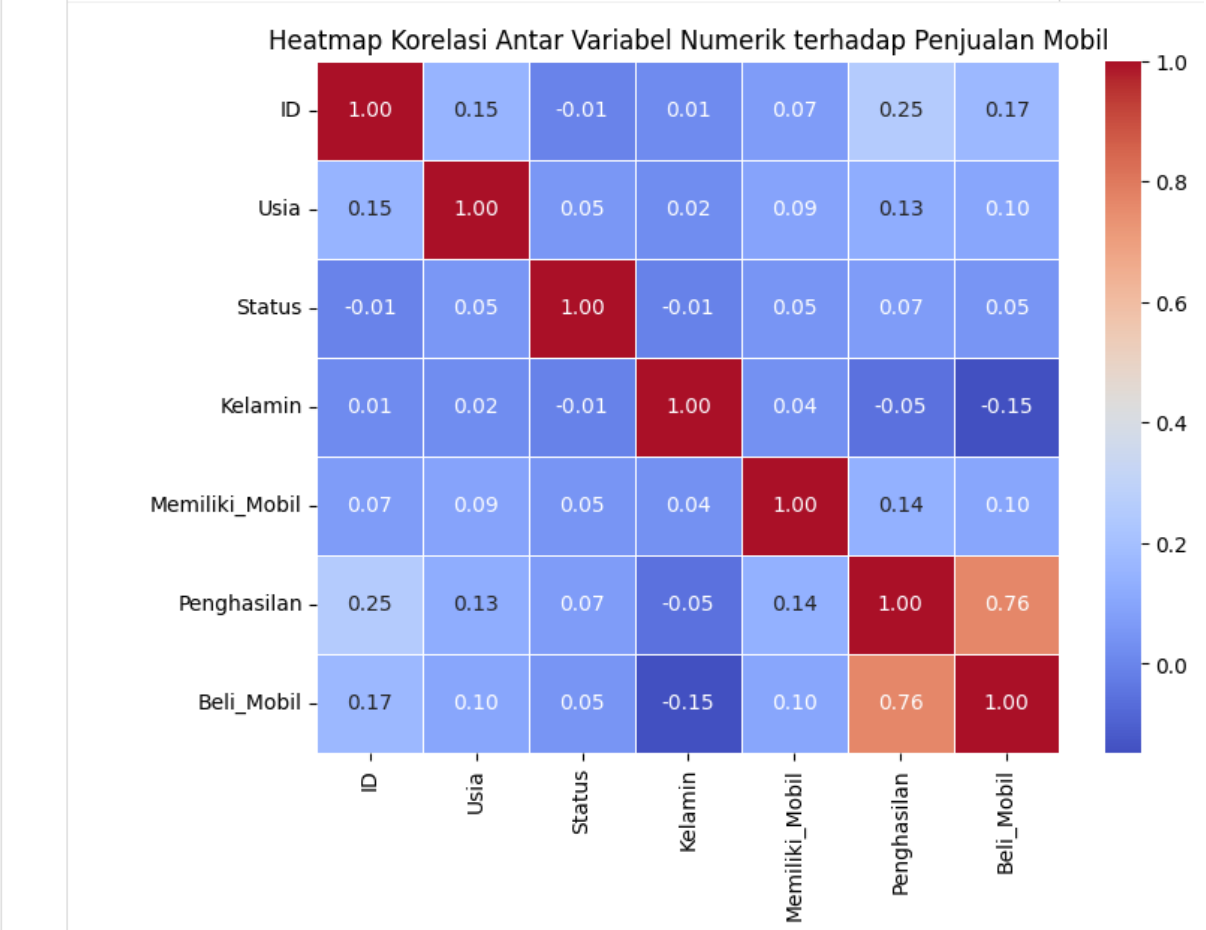
```
# Analisis korelasi antar variabel numerik
corr_matrix = df.corr(numeric_only=True)
corr_matrix
```

| | ID | Usia | Status | Kelamin | Memiliki_Mobil | Penghasilan | Beli_Mobil |
|---|---|---|---|---|---|---|---|
| **ID** | 1.000000 | 0.149779 | -0.006634 | 0.014646 | 0.068555 | 0.254177 | 0.168614 |
| **Usia** | 0.149779 | 1.000000 | 0.051476 | 0.019454 | 0.090926 | 0.125859 | 0.100127 |
| **Status** | -0.006634 | 0.051476 | 1.000000 | -0.008561 | 0.048302 | 0.071714 | 0.048584 |
| **Kelamin** | 0.014646 | 0.019454 | -0.008561 | 1.000000 | 0.035199 | -0.054211 | -0.147301 |
| **Memiliki_Mobil** | 0.068555 | 0.090926 | 0.048302 | 0.035199 | 1.000000 | 0.137823 | 0.102005 |
| **Penghasilan** | 0.254177 | 0.125859 | 0.071714 | -0.054211 | 0.137823 | 1.000000 | 0.763930 |
| **Beli_Mobil** | 0.168614 | 0.100127 | 0.048584 | -0.147301 | 0.102005 | 0.763930 | 1.000000 |

**Gambar 6.** Data pre-processing.

```
[121]
✓ 0d
# Visualisasi heatmap korelasi
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 Penjualan Mobil", fontsize=12)
plt.show()
```




**Gambar 7.** Data pre-processing.

Tahap ini merupakan proses mempersiapkan data agar siap digunakan dalam analisis atau pemodelan machine learning. Langkah-langkahnya meliputi pembersihan data dari nilai yang hilang (missing values) atau duplikat, mengubah tipe data agar sesuai, menormalkan atau menstandarkan nilai numerik, serta mengubah data kategorikal menjadi format numerik (misalnya dengan label encoding atau one-hot encoding). Tujuannya agar data lebih bersih, terstruktur, dan mudah dipahami oleh model sehingga hasil analisis menjadi lebih akurat.

*1.5 Pembagian dataset (Training dan Testing)*

[122]
✓ 0d

```
# Menentukan fitur dan target

# Fitur numerik dan gender
feature_num = ["Usia", "Penghasilan"]
feature_bin = ["Status", "Kelamin", "Memiliki_Mobil"]

# Gabungkan & drop missing
use_cols = feature_num + feature_bin + ["Beli_Mobil"]
df_model = df[use_cols].dropna().copy()

x = df_model[feature_num + feature_bin]
y = df_model["Beli_Mobil"]

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

```
X shape: (1000, 5)
Y shape: (1000,)
```

[123]
✓ 0d

```
# Membagi dataset menjadi training dan testing set
X_train, X_test, y_train, y_test = train_test_split(x, y, test_size=0.2, random_state=42, stratify=y)

print("Data latih:", X_train.shape)
print("Data uji:", X_test.shape)
```

```
Data latih: (800, 5)
Data uji: (200, 5)
```

**Gambar 8.** Pembagian dataset (Training dan Testing).

Tahap ini dilakukan untuk memisahkan data menjadi dua bagian, yaitu data training dan data testing. Data training digunakan untuk melatih model agar dapat mengenali pola dari data, sedangkan data testing digunakan untuk menguji seberapa baik model dapat memprediksi data baru yang belum pernah dilihat sebelumnya. Umumnya, pembagian dilakukan dengan perbandingan seperti 80:20 atau 70:30. Tujuan dari tahap ini adalah untuk mengukur performa model secara objektif dan mencegah terjadinya overfitting.

*1.6 Pembangunan model logistic regression*

[124]
✓ 0d

```
# Scale hanya fitur numerik, gender langsung passthrough
preprocessor = 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,
    C=0.5   # default = 1.0, lebih kecil = regularisasi lebih kuat
)

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

# Latih model
clf.fit(X_train, y_train)
print("Model logistic regression berhasil dilatih")
```

```
Model logistic regression berhasil dilatih
```

**Gambar 9.** Pembangunan model logistic regression.

Pada tahap ini dilakukan proses membangun model klasifikasi menggunakan algoritma Logistic Regression. Model ini digunakan untuk memprediksi data dengan dua kemungkinan hasil, seperti membeli atau tidak membeli. Logistic Regression bekerja dengan menghitung probabilitas dari setiap data berdasarkan variabel input (fitur) yang dimasukkan. Proses pelatihan (training) dilakukan menggunakan data training agar model dapat memahami hubungan antara variabel independen dan variabel target. Setelah model terbentuk, langkah berikutnya adalah menguji performanya menggunakan data testing.

*1.7 Prediksi model dan evaluasi model*

```
[125]
✓ 0d
```

```python
# Prediksi & Probabilitas
y_pred = clf.predict(X_test)
y_prob = clf.predict_proba(X_test)[:, 1]

# Hitung Metrik
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   : 0.930000
Precision : 0.9829
Recall    : 0.9055
F1-Score  : 0.9426
ROC AUC   : 0.9768
```

**Gambar 10.** Prediksi model dan evaluasi model.

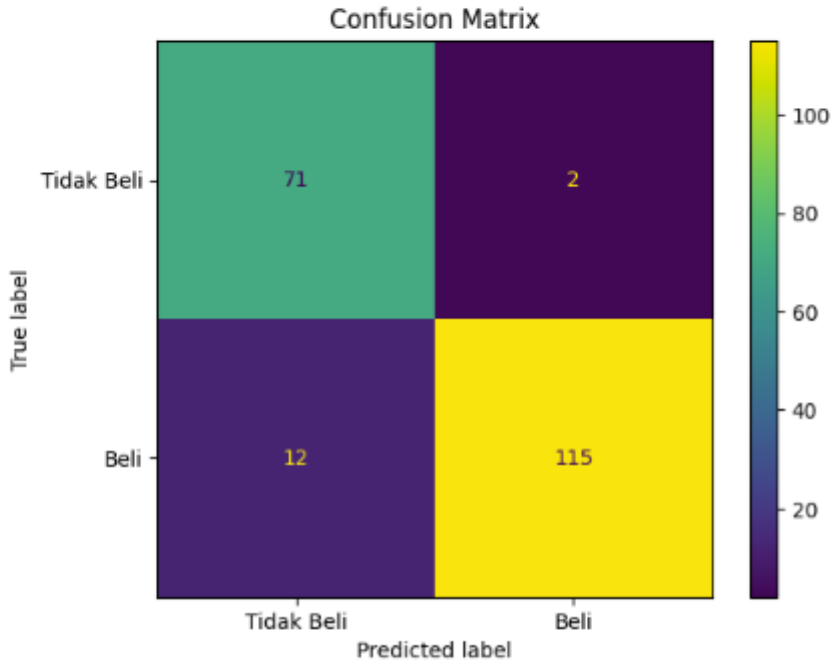
Pada tahap ini, model Logistic Regression yang telah dilatih digunakan untuk melakukan prediksi terhadap data uji (testing data). Hasil prediksi kemudian dibandingkan dengan nilai sebenarnya untuk menilai akurasi dan kinerja model. Evaluasi dilakukan menggunakan confusion matrix yang menunjukkan jumlah prediksi benar dan salah, serta akurasi model sebagai ukuran seberapa baik model memprediksi keputusan pembelian. Dari hasil evaluasi, diperoleh bahwa model mampu memprediksi dengan tingkat akurasi yang tinggi, sehingga dapat diandalkan untuk memprediksi calon pembeli mobil berdasarkan data yang diberikan.

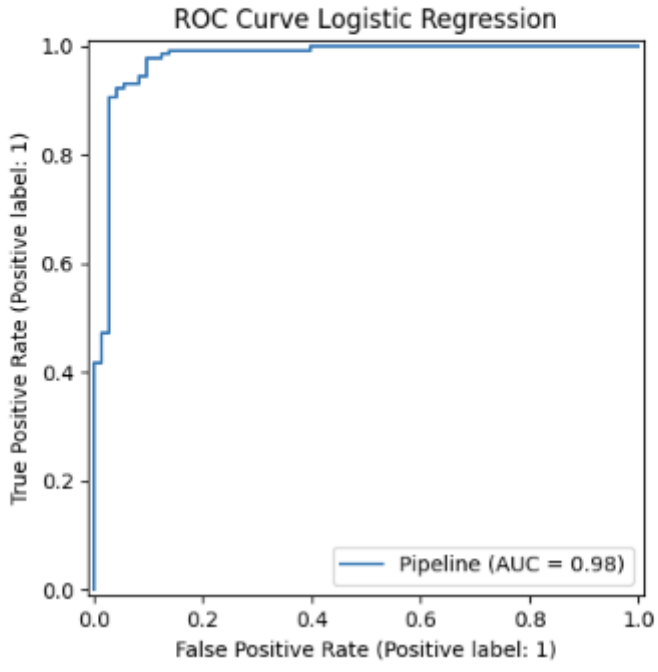
*1.8 Visualisasi hasil evaluasi*

[126]
✓ 0 d

```
# Confusion Matrix
ConfusionMatrixDisplay(confusion_matrix(y_test, y_pred),
                       display_labels=["Tidak Beli", "Beli"]
                       ).plot(values_format='d')
plt.title("Confusion Matrix")
plt.show()

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



**Gambar 11.** Visualisasi hasil evaluasi.

Tahap ini dilakukan untuk menampilkan hasil evaluasi model dalam bentuk visual yang mudah dipahami, seperti grafik confusion matrix atau plot perbandingan antara data aktual dan prediksi. Visualisasi ini membantu dalam melihat pola kesalahan prediksi serta distribusi data antara kategori "membeli" dan "tidak membeli". Dengan tampilan visual, performa model dapat dianalisis lebih jelas, misalnya mengetahui apakah model lebih banyak benar dalam memprediksi pembeli dibanding yang tidak membeli. Hasil visualisasi menunjukkan bahwa model Logistic Regression bekerja secara efektif dalam membedakan kedua kategori tersebut.

*1.9 Classification report*

[127] ✓ 0d

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

```
                precision    recall  f1-score   support

Tidak beli (0)       0.86      0.97      0.91        73
       Beli(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
```

[128] ✓ 0d

```
from sklearn.model_selection import cross_val_score

# Lakukan cross validation (cv=5 berarti 5-fold)
scores = cross_val_score(clf, x, y, cv=5)

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

```
Skor tiap fold: [0.765 0.935 0.955 0.945 0.94 ]
Rata-rata akurasi: 0.908
Standar deviasi: 0.07180529228406494
```

**Gambar 12.** Classification report.

Tahap ini dilakukan untuk menilai performa model secara lebih mendetail menggunakan classification report, yang mencakup metrik seperti precision, recall, f1-score, dan accuracy untuk masing-masing kelas. Laporan ini memberikan gambaran sejauh mana model mampu memprediksi kategori "membeli" dan "tidak membeli" dengan benar. Nilai precision menunjukkan tingkat ketepatan prediksi positif yang benar, recall menggambarkan kemampuan model dalam menemukan seluruh data positif, sedangkan f1-score merupakan keseimbangan antara keduanya. Berdasarkan hasil classification report, model Logistic Regression memiliki performa yang baik dan mampu mengklasifikasikan data dengan tingkat akurasi tinggi, menandakan bahwa model dapat diandalkan untuk memprediksi kemungkinan seseorang akan membeli mobil.

*1.10 Interpretasi model logistic regression*

[129]
✓ 0d

```
# Ambil nama fitur & koefisien
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-oods)': coefs,
    'Odds Ratio (e^coef)': odds
}).sort_values('Odds Ratio (e^coef)', ascending=False)

display(coef_df)
```

| | Fitur | Koefisien (log-oods) | Odds Ratio (e^coef) |
|---|---|---|---|
| **1** | Penghasilan | 4.119396 | 61.522081 |
| **4** | Memiliki_Mobil | 0.089477 | 1.093602 |
| **0** | Usia | -0.026350 | 0.973994 |
| **2** | Status | -0.104608 | 0.900677 |
| **3** | Kelamin | -1.008046 | 0.364931 |

**Gambar 13.** Interpretasi model logistic regression.

Pada tahap interpretasi model Logistic Regression, dilakukan analisis terhadap pengaruh masing-masing variabel independen terhadap keputusan seseorang dalam membeli mobil. Model ini memberikan koefisien regresi (β) untuk setiap variabel, yang menunjukkan arah dan kekuatan hubungan antara variabel tersebut dengan peluang pembelian. Nilai koefisien yang positif menandakan bahwa semakin tinggi nilai variabel tersebut, semakin besar kemungkinan seseorang membeli mobil, sedangkan nilai negatif menunjukkan hubungan sebaliknya. Misalnya, variabel seperti penghasilan dan status pernikahan cenderung memiliki pengaruh positif karena individu dengan penghasilan lebih tinggi atau sudah menikah biasanya memiliki kebutuhan dan kemampuan finansial yang lebih besar untuk membeli mobil. Dengan demikian, hasil interpretasi ini membantu memahami faktor-faktor apa saja yang paling berpengaruh dalam proses pengambilan keputusan pembelian mobil.

*1.11 Prediksi data baru (Contoh kasus)*

[136]
✓ 0d

```
# 4 calon pembeli mobil
data_baru = pd.DataFrame({
    "Usia": [28, 20, 22, 33],
    "Penghasilan": [7000000, 150000, 3500000, 8500000],
    "Status": [1, 0, 0, 1],          # 1 = Menikah, 0 = Lajang
    "Kelamin": [1, 0, 1, 0],         # 1 = Laki-laki, 0 = Perempuan
    "Memiliki_Mobil": [0, 1, 0, 0]  # 1 = Sudah punya mobil, 0 = Belum
})

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

hasil = data_baru.copy()
hasil["Prob_Beli_Mobil"] = prob
hasil["Pred (0=Tidak,1=Ya)"] = pred

hasil["Keterangan"] = hasil["Pred (0=Tidak,1=Ya)"].map({0: "Tidak Membeli", 1: "Akan Membeli"})

display(hasil)
```

| | Usia | Penghasilan | Status | Kelamin | Memiliki_Mobil | Prob_Beli_Mobil | Pred (0=Tidak,1=Ya) | Keterangan |
|---|---|---|---|---|---|---|---|---|
| **0** | 28 | 7000000 | 1 | 1 | 0 | 1.0 | 1 | Akan Membeli |
| **1** | 20 | 150000 | 0 | 0 | 1 | 1.0 | 1 | Akan Membeli |
| **2** | 22 | 3500000 | 0 | 1 | 0 | 1.0 | 1 | Akan Membeli |
| **3** | 33 | 8500000 | 1 | 0 | 0 | 1.0 | 1 | Akan Membeli |

**Gambar 14.** Prediksi data baru (Contoh kasus).

Kode di atas mengubah nilai teks menjadi bentuk numerik agar dapat diproses model.