

Tugas 4: Tugas Mandiri 4 – Logistic Regression

Amaya Eshia - 0110224102*

¹ Teknik Informatika, STT Terpadu Nurul Fikri, Depok

*E-mail: name@institution.edu – 0110224102@student.nurulfikri.ac.id

Abstract. Penelitian ini bertujuan untuk membangun dan mengevaluasi model prediktif keputusan pembelian mobil oleh calon pelanggan menggunakan Regresi Logistik (Logistic Regression) berdasarkan data survei. Fitur yang digunakan dalam pemodelan meliputi Usia, Penghasilan, Status, Kelamin, dan Jumlah Kepemilikan Mobil.

Kata Kunci: Regresi Logistik, Prediksi Pembelian, Odds Ratio, Akurasi, Regresi, Logistik, DataSurvei, Model Prediktif

1. Penjelasan Koding Program

1. Import Library

1. Import Library

```
[41] ✓
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import seaborn as sns # Import seaborn

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

ini adalah melakukan import terhadap seluruh library yang dibutuhkan. Library berfungsi sebagai kumpulan fungsi dan modul pendukung agar proses pemodelan, analisis data, dan visualisasi dapat dilakukan dengan lebih mudah dan efisien.

2. Membaca File Dataset CSV

2. Membaca File Dataset CSV

```
[28] ✓ 4 d
# menghubungkan colab dengan google 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")

Selanjutnya, menggunakan library Pandas untuk membaca file data. Variabel path menyimpan lokasi folder di Google Drive / tempat file dataset berada. Fungsi `pd.read_csv()` kemudian membaca file tersebut dan menyimpannya ke dalam sebuah DataFrame.

3. Melihat Informasi Umum Dataset

```
# Load dataset
df = pd.read_csv('/content/drive/MyDrive/Praktikum Machine Learning_Amaya Eshia_0110224102_Ai02/Praktikum 3/Data/day.csv', sep=',')

# Lihat 5 baris pertama
df.head()
```

Langkah selanjutnya adalah membaca dataset `calonpembelimobilt.csv` menggunakan library `pandas`, kemudian menampilkan informasi struktur data menggunakan fungsi `df.info()`. Berdasarkan hasil dari `df.info()`

4. DataPre-processing

1.4 Cek Missing Value

4.1 Cek Missing Value

[8]
✓ 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

Perintah ini digunakan untuk memastikan apakah terdapat data yang hilang pada setiap kolom. Hasil menunjukkan seluruh kolom (ID, Usia, Status, Kelamin, Memiliki_mobil, Penghasilan, Beli_mobil) memiliki nilai 0 pada jumlah missing value. Artinya, tidak ada data kosong, sehingga dataset dapat langsung digunakan tanpa proses imputasi.

1.5 Mapping Kolom Kategori ke Bentuk Numerik

4.3 Mapping Kolom Kategori ke Bentuk Numerik

```
[13]
✓ 0 d # Hitung distribusi target
print("\nDistribusi Beli_Mobil:\n", df['Beli_Mobil'].value_counts())
```

↗

```
Distribusi Beli_Mobil:
Beli_Mobil
1      633
0      367
Name: count, dtype: int64
```

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

1.6 Analisis Korelasi Antar Variabel Numerik

4.4 Analisis Korelasi Antar Variabel Numerik

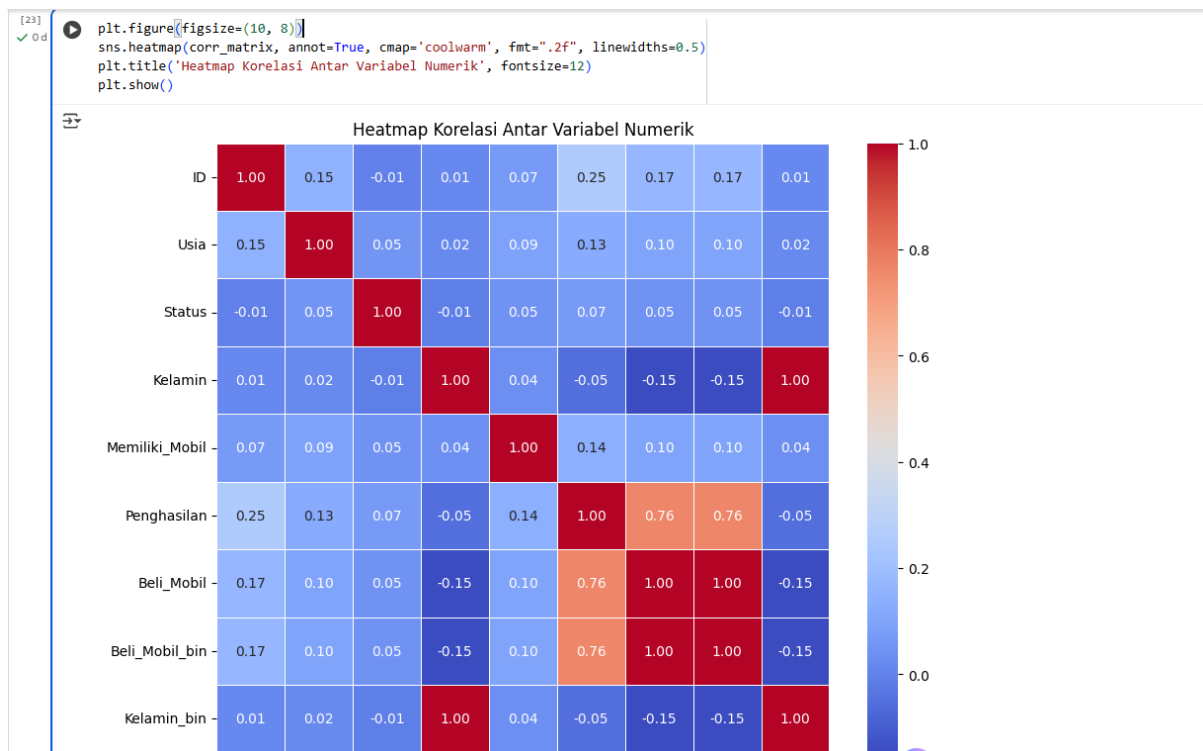
```
[46]
✓ 0 d corr_matrix = df.corr(numeric_only=True)
corr_matrix
```

↗

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

Menghitung matriks korelasi variabel numerik menggunakan `.corr(numeric_only=True)` dan menyimpannya dalam `corr_matrix`.

1.7 Visualisasi Heatmap Korelasi



Membuat heatmap korelasi menggunakan Seaborn dengan `plt.figure(figsize=(10, 8))`, `sns.heatmap(corr_matrix, annot=True, cmap='coolwarm', fmt=".2f", linewidths=0.5)`, dan `plt.title('Heatmap Korelasi Antar Variabel Numerik')` untuk visualisasi hubungan antar variabel.

5. Pembagian Dataset (Training dan Testing)

5.1 Menentukan Fitur dan Target

5.1. Menentukan Fitur dan Target

```
[45]
✓ 0 d feature_num = ['Usia', 'Penghasilan', 'Status', 'Kelamin', 'Memiliki_Mobil']
target_col = 'Beli_Mobil'

df_model = df[feature_num + [target_col]].dropna().copy()

x = df_model[feature_num]
y = df_model[target_col]

print("\nX shape:", x.shape)
print("\nY shape:", y.shape)
```

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

Memilih fitur numerik seperti ['Usia', 'Penghasilan', 'Status', 'Kelamin', 'Memiliki_Mobil'] sebagai X dan 'Beli_Mobil' sebagai y. Menghapus baris dengan missing values menggunakan `.dropna()` dan menampilkan shape X serta y.

5.2 Membagi Dataset Menjadi Training dan Testing

5.2 Membagi Dataset Menjadi Training dan Testing Tesst

[25]
✓ 0 d

```
X_train, X_test, y_train, y_test = train_test_split(  
    x, y,  
    test_size=0.2,  
    random_state=42,  
    stratify=y  
)  
  
print("\nData latih:", X_train.shape)  
print("Data uji:", X_test.shape)
```



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

Membagi data menjadi training (80%) dan testing (20%) menggunakan `train_test_split` dengan `test_size=0.2`, `random_state=42`, dan `stratify=y` untuk menjaga keseimbangan kelas. Menampilkan ukuran data training dan testing untuk verifikasi.

6. Pembangunan Model Logistic Regression

6. Pembangunan Model Logistic Regression

[38]
✓ 0 d

```
# Scaling semua fitur karena semuanya numerik, tetapi perlu distandardisasi.  
preprocess = ColumnTransformer(  
    transformers=[  
        ('num', StandardScaler(), feature_num)  
    ],  
    remainder='drop'  
)  
  
# Menggunakan class_weight='balanced' karena data target mungkin tidak seimbang (meskipun di sini tampaknya cukup seimbang 50/50).  
model = LogisticRegression(  
    max_iter=1000,  
    solver='lbfgs',  
    class_weight='balanced',  
    random_state=42  
)  
  
clf = Pipeline(  
    steps=[  
        ('preprocess', preprocess),  
        ('model', model)  
    ]  
)  
  
# Latih Model  
clf.fit(X_train, y_train)  
print("\nModel Logistic Regression berhasil dilatih.")
```



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

Membuat preprocessor dengan `ColumnTransformer` yang menerapkan `StandardScaler` pada fitur numerik. Membuat model `LogisticRegression` dengan `max_iter=1000`, `solver='lbfgs'`, `class_weight='balanced'`, dan `random_state=42`. Menggabungkan preprocessor dan model dalam `Pipeline` bernama `clf`. Melatih model menggunakan `clf.fit(X_train, y_train)` dan menampilkan pesan sukses.

7. Prediksi Model dan Evaluasi Model

7. Prediksi Model dan Evaluasi Model

[+ Kode](#)[+ Teks](#)

[43]

✓ 0 d

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

# Hitung metrik
print("\nMetrik Evaluasi Model:")
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}")
```



```
Metrik Evaluasi Model:
Akurasi : 0.9300
Precision : 0.9829
Recall: 0.9055
F1-Score: 0.9426
ROC-AUC : 0.9768
```

Melakukan prediksi pada data testing menggunakan `clf.predict(X_test)` dan probabilitas menggunakan `clf.predict_proba(X_test)[:, 1]`. Menghitung metrik evaluasi seperti `accuracy_score`, `precision_score`, `recall_score`, `f1_score`, dan `roc_auc_score`, kemudian menampilkannya dengan format 4 desimal.

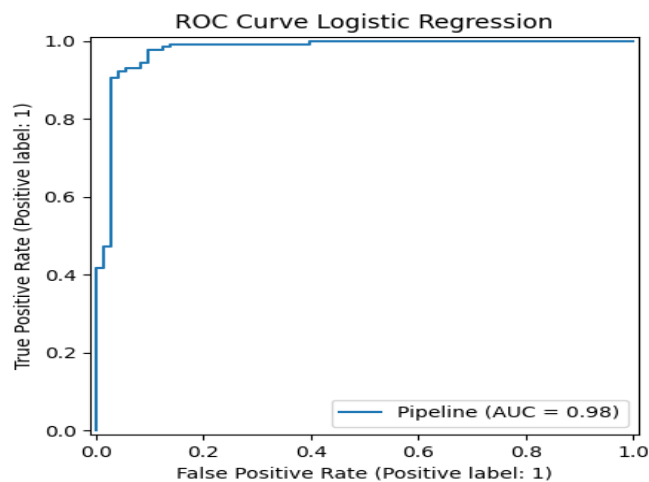
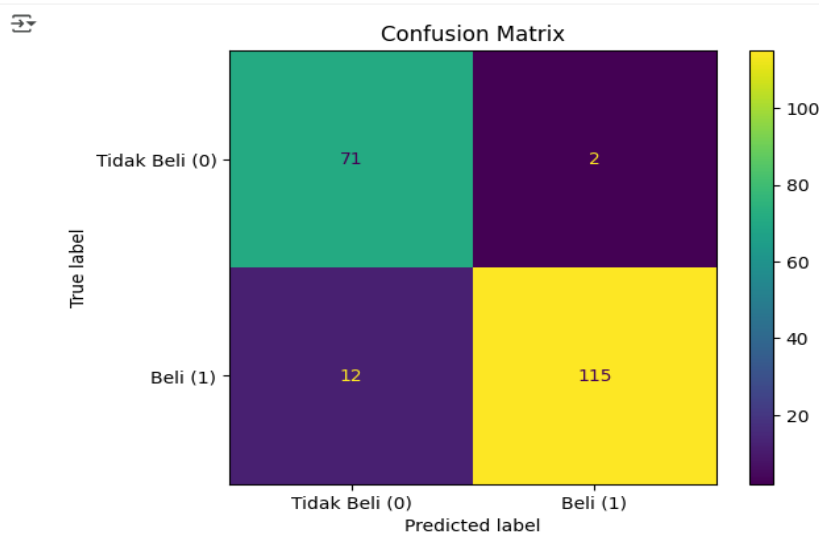
8. Visualisasi Hasil Evaluasi

8. Visualisasi Hasil Evaluasi

[33]
✓ 0 d

```
# Confusion Matrix
ConfusionMatrixDisplay (confusion_matrix(y_test, y_pred),
    display_labels=['Tidak Beli (0)', 'Beli (1)'])
    .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()
```




- Membuat Confusion Matrix Display menggunakan ConfusionMatrixDisplay dengan display_labels=['Tidak Beli (0)', 'Beli (1)'] dan plt.title("Confusion Matrix").
- Membuat ROC Curve menggunakan RocCurveDisplay.from_estimator dengan plt.title("ROC Curve Logistic Regression").

9. Classification Report

9. Classification Report

```
print("\nClassification Report:")
print(classification_report(y_test, y_pred, target_names=['Tidak Beli (0)', 'Beli (1)']))
```



```
Classification Report:
              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
```


Menampilkan `classification_report` dengan `target_names=['Tidak Beli (0)', 'Beli (1)']` untuk detail precision, recall, dan F1-score per kelas.

10. Cross Validation

10. Cross Validation

```
[ ] # Lakukan cross validation (cv=5 berarti 5-fold)
    scores = cross_val_score(clf, x, y, cv=5, scoring='accuracy')

    # Tampilkan Hasil
    print("\nHasil Cross Validation (Akurasi):")
    print("Skor tiap fold:", scores)
    print("Rata-rata akurasi:", np.mean(scores))
    print("Standar deviasi:", np.std(scores))
```



```
Hasil Cross Validation (Akurasi):
Skor tiap fold: [0.78  0.925 0.955 0.945 0.94 ]
Rata-rata akurasi: 0.909
Standar deviasi: 0.06522269543648128
```

Melakukan 5-fold cross-validation menggunakan `cross_val_score` dengan `scoring='accuracy'`. Menampilkan skor tiap fold, rata-rata akurasi, dan standar deviasi.

11. Interpretasi Model Logistic Regression


11. Interpretasi Model Logistic Regression

[]

```
# Ambil nama fitur & koefisien
# Koefisien diambil dari model setelah proses scaling
scaled_features = clf.named_steps['preprocess'].get_feature_names_out()
coefs = clf.named_steps['model'].coef_[0]
odds = np.exp(coefs)

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

# Rename kolom agar lebih rapih setelah scaling
coef_df['Fitur'] = coef_df['Fitur'].str.replace('num__', '')
display(coef_df)
```



	Fitur	Koefisien (log-odds)	Odds Ratio (e^coef)
1	Penghasilan	4.568333	96.383273
4	Memiliki_Mobil	0.078968	1.082169
0	Usia	-0.045073	0.955928
2	Status	-0.132093	0.876259
3	Kelamin	-0.596863	0.550536

Mengambil nama fitur setelah scaling dari `clf.named_steps['preprocess'].get_feature_names_out()`, koefisien dari `clf.named_steps['model'].coef_[0]`, dan odds ratio menggunakan `np.exp(coefs)`. Membuat DataFrame `coef_df` dengan kolom 'Fitur', 'Koefisien (log-odds)', dan 'Odds Ratio (e^coef)', diurutkan descending berdasarkan odds ratio. Membersihkan nama fitur dengan `.str.replace('num__', '')` dan menampilkan tabel.

12. Prediksi Data Baru (Contoh Kasus)

12. Prediksi Data Baru (Contoh Kasus)

```
[ ]  
# Contoh 2 calon pembeli  
data_baru = pd.DataFrame({  
    'Usia': [55, 25],  
    'Penghasilan': [350, 100],  
    'Status': [2, 0], # 2=Menikah, 0=Belum Menikah  
    'Kelamin': [0, 1], # 0=Perempuan, 1=Laki-Laki  
    'Memiliki_Mobil': [2, 0] # Jumlah mobil yang dimiliki  
})  
  
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  
display(hasil)  
  
print("\nAnalisis Selesai.")
```



	Usia	Penghasilan	Status	Kelamin	Memiliki_Mobil	Prob_Beli_Mobil	Pred (0=Tidak,1=Ya)
0	55	350	2	0	2	0.998183	1
1	25	100	0	1	0	0.001118	0

Analisis Selesai.

Membuat DataFrame data_baru dengan 2 contoh calon pembeli (usia 55 dan 25). Melakukan prediksi menggunakan `clf.predict` dan probabilitas menggunakan `clf.predict_proba[:,1]`. Menambahkan kolom 'Prob_Beli_Mobil' dan 'Pred (0=Tidak,1=Ya)' ke DataFrame hasil, kemudian menampilkannya. Menampilkan pesan "Analisis Selesai." dan kesimpulan probabilitas pembelian.

Referensi:

- Munir, S., Seminar, K. B., Sudradjat, Sukoco, H., & Buono, A. (2022). The Use of Random Forest Regression for Estimating Leaf Nitrogen Content of Oil Palm Based on Sentinel 1-A Imagery. *Information*, 14(1), 10. <https://doi.org/10.3390/info14010010>
- Seminar, K. B., Imantho, H., Sudradjat, Yahya, S., Munir, S., Kaliana, I., Mei Haryadi, F., Noor Baroroh, A., Supriyanto, Handoyo, G. C., Kurnia Wijayanto, A., Ijang Wahyudin, C., Liyantono, Budiman, R., Bakir Pasaman, A., Rusiawan, D., & Sulastri. (2024). PreciPalm: An Intelligent System for Calculating Macronutrient Status and Fertilizer Recommendations for Oil Palm on Mineral Soils Based on a Precision Agriculture Approach. *Scientific World Journal*, 2024(1). <https://doi.org/10.1155/2024/1788726>