Tugas 1: Laporan Praktikum Tugas Mandiri

Aan Adriyana - 0110224014

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

*E-mail: 0110224014@student.nurulfikri.ac.id

Abstract. Regresi logistik adalah teknik statistik dan pembelajaran mesin untuk memprediksi probabilitas hasil biner (dua kemungkinan), seperti "ya" atau "tidak", berdasarkan satu atau lebih variabel independent. Regresi logistik di Python digunakan untuk klasifikasi biner, di mana tujuannya adalah memprediksi probabilitas suatu peristiwa terjadi dengan dua kemungkinan hasil.

1. Prediksi dari kasus Dataset

1.1 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.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
)
```

1.2 Membaca data file csv

```
import pandas as pd

df = pd.read_csv("../content/drive/MyDrive/praktikum_ml/praktikum04/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

1.3 Melihat Informasi Umum Dataset

```
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
```

1.4 Cek Missing Value

```
#cek missing values
df.isnull().sum()
```

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

dtype: int64

1.5 Cek Nilai Unik

```
df['Memiliki_Mobil'].unique()
```

```
array([0, 1, 2, 4, 3])
```

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

```
array([0, 1])
```

1.6 Mapping Kolom Kategorik ke Bentuk Numerik

```python
# 1. Mapping kolom Beli_Mobil -> biner
# Mapping: 1 jika Beli_Mobil is 1, 0 otherwise
map_membeli_mobil = {0: 0, 1: 1} # Corrected mapping: 1 if Beli_Mobil is 1, 0 otherwise
df['Membeli_Mobil_bin'] = df['Beli_Mobil'].map(map_membeli_mobil).astype('Int64')

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

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

```
Distribusi Membeli_Mobil_bin:
 Membeli_Mobil_bin
1    633
0    367
Name: count, dtype: Int64

Distribusi JK_bin:
 JK_bin
1    519
0    481
Name: count, dtype: int64
```

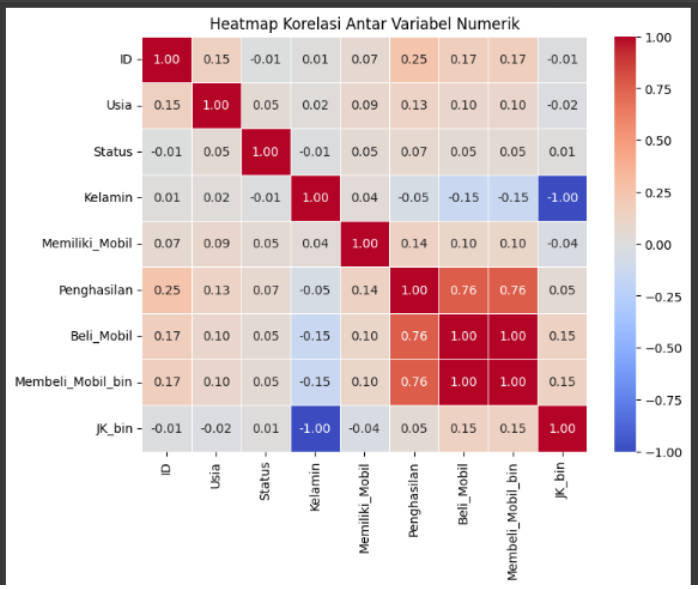
1.7 Analisis Korelasi Antar Variabel Numerik

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

	ID	Usia	Status	Kelamin	Memiliki_Mobil	Penghasilan	Beli_Mobil	Membeli_Mobil_bin	JK_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
Membeli_Mobil_bin	0.168614	0.100127	0.048584	-0.147301	0.102005	0.763930	1.000000	1.000000	0.147301
JK_bin	-0.014646	-0.019454	0.008561	-1.000000	-0.035199	0.054211	0.147301	0.147301	1.000000

1.8 Visualisasi Heatmap Korelasi

```
#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")
plt.show()
```



1.9 Menentukan Fitur & Target

```
# Fitur numerik dan binary
feature_num = ['Usia', 'Penghasilan']
feature_bin = ['Kelamin'] # Use 'Kelamin' as a binary feature

# Gabungkan & drop missing
use_cols = feature_num + feature_bin + ['Membeli_Mobil_bin'] # Include 'Membeli_Mobil_bin' as the target
df_model = df[use_cols].dropna().copy()

X = df_model[feature_num + feature_bin]
y = df_model['Membeli_Mobil_bin'] # Set 'Membeli_Mobil_bin' as the target

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

```
X shape: (1000, 3)
y shape: (1000,)
```

1.10 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, 3)
Data uji: (200, 3)
```

1.11 Pembangunan Model Logistic Regression

```
# Scale hanya fitur numerik, gender langsung passthrough
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
)

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

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

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

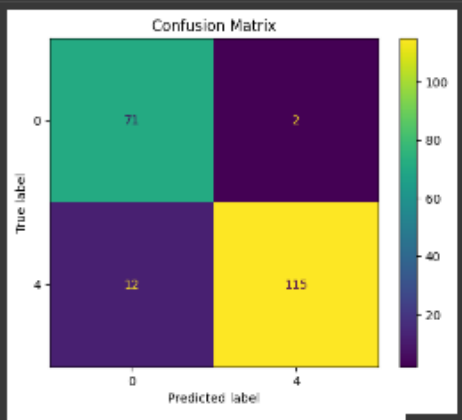
1.12 Prediksi Model dan Evaluasi Model

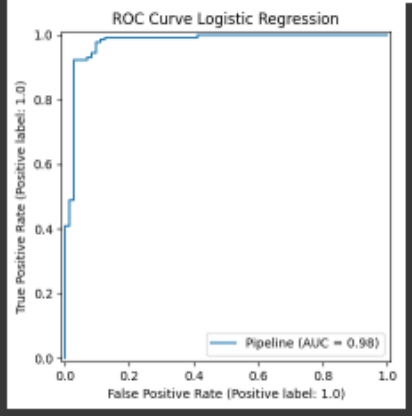
```
# Prediksi & probabilitas
y_pred = clf.predict(X_test)
y_prob = clf.predict_proba(X_test)[:, 1]

# 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 : 0.9300
Precision : 0.9829
Recall : 0.9055
F1-Score : 0.9426
ROC-AUC : 0.9771
```

1.13 Visualisasi Hasil Evaluasi

```
# Confusion Matrix
ConfusionMatrixDisplay(confusion_matrix(y_test, y_pred),
                       display_labels=["0", "4"]
                       ).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()
```





1.14 Classification Report

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

```
                        precision    recall  f1-score   support

Tidak Memiliki Mobil (0)       0.86      0.97      0.91        73
      Memiliki Mobil (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
```

1.15 Classification Report

```
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: [0.79  0.915 0.955 0.955 0.94 ]
Rata-rata akurasi: 0.9109999999999999
Standar deviasi: 0.062241465278381716
```

1.16 Interpretasi Model Logistic Regression

```
# 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-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)
1	Penghasilan	4.526215	92.408123
0	Usia	-0.037045	0.963633
2	Kelamin	-1.140991	0.319502

1.17 Prediksi Data Baru

```
data_baru = pd.DataFrame({
    'Usia': [24, 10],
    'Kelamin': [79.0, 72.5],
    'Penghasilan': [9.2, 7.8],
    'JK_bin': [1, 0] # 1=Laki-Laki, 0=Perempuan
})

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

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

	Usia	Kelamin	Penghasilan	JK_bin	Beli Mobil	Pred (0=Tidak,1=Ya)
0	24	79.0	9.2	1	3.193232e-44	0.0
1	10	72.5	7.8	0	5.172335e-41	0.0

Link Colab:
https://colab.research.google.com/drive/1AntYJ0KI7HybZVKWZBc3PiVfUlv0RJsM?usp=sharing

Referensi:

https://www.datacamp.com/tutorial/understanding-logistic-regression-pythons:/