

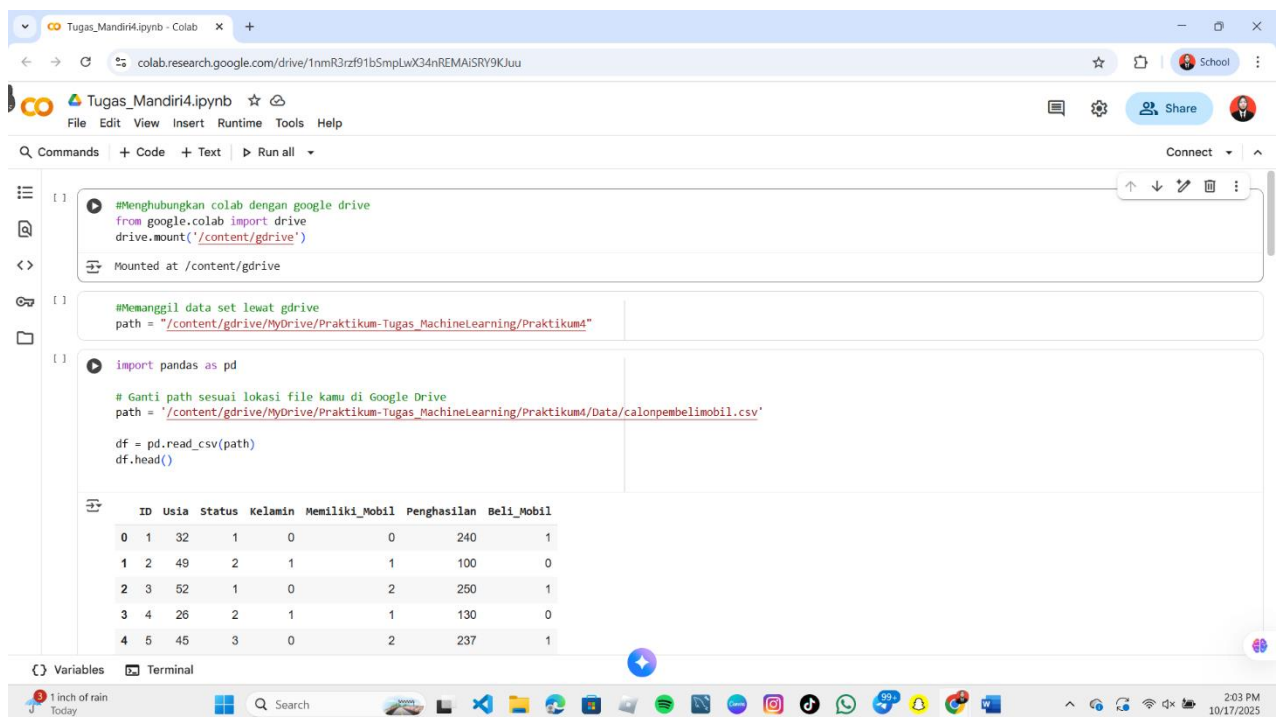
Tugas 4: Tugas Praktikum Mandiri 4 – Machine Learning

Yurida Yahsyia 1 - 0110224100 1*

¹ Teknik Informatika, STT Terpadu Nurul Fikri, Depok

*E-mail: 0110224100@student.nurulfikri.ac.id – email mahasiswa 1

1. Latihan Mandiri 4



The screenshot shows a Google Colab notebook titled 'Tugas_Mandiri4.ipynb'. The code is as follows:

```
#Menghubungkan colab dengan google drive
from google.colab import drive
drive.mount('/content/gdrive')

#Memanggil data set lewat gdrive
path = "/content/gdrive/MyDrive/Praktikum-Tugas_MachineLearning/Praktikum4"

import pandas as pd

# Ganti path sesuai lokasi file kamu di Google Drive
path = '/content/gdrive/MyDrive/Praktikum-Tugas_MachineLearning/Praktikum4/Data/calonpembelimobil.csv'

df = pd.read_csv(path)
df.head()
```

The output shows the file is mounted at /content/gdrive. Below the code, the head of the CSV file is displayed as a table:

	ID	Usia	Status	Kelamin	Memiliki_Mobil	Penghasilan	Belis_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. from google.colab import drive**
Memanggil modul bawaan Colab.
- 2. drive.mount('/content/gdrive')**
Membuat “jembatan” supaya Colab bisa membaca file yang ada di Drive, dan semua file bisa diakses lewat folder /content/gdrive/MyDrive/.
- 3. path = "/content/gdrive/MyDrive/Praktikum/Praktikum4"**
path = alamat menuju folder Praktikum4 di dalam Google Drive.
- 4. import pandas as pd**
memanggil library Pandas.
- 5. path = '/content/gdrive/MyDrive/Praktikum-Tugas Machinelearning/Praktikum/Data/calonpembelimobil.csv'**
Membuat variabel bernama path yang menyimpan jalur lengkap (path) menuju berkas dataset bernama calonpembelimobil.csv yang tersimpan di Google Drive.

6. **df = pd.read_csv(path)**

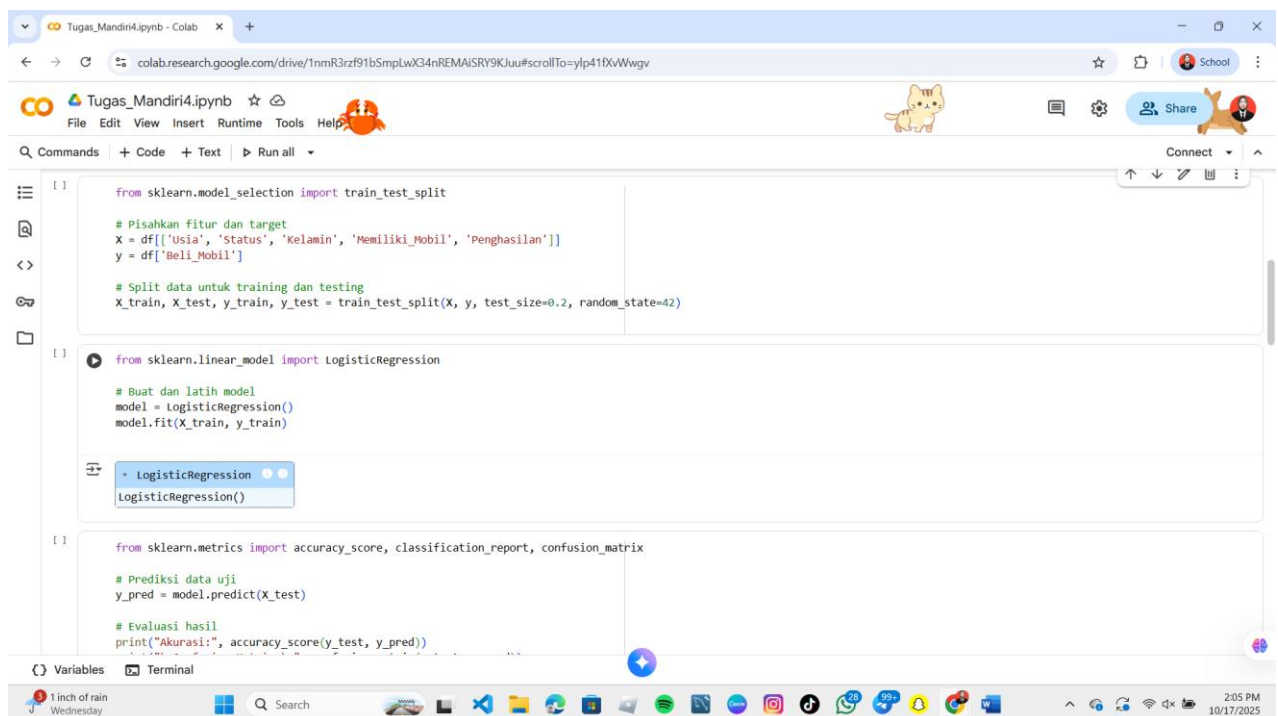
Membaca (membuka) berkas CSV yang ditunjuk oleh variabel path menggunakan fungsi `pd.read_csv()` dari pustaka `pandas`

7. **df.head()**

Fungsi yang menampilkan 5 baris pertama dari DataFrame `df`

Output:

Hasil keluaran dari baris kode terakhir (`df.head()`) adalah tampilan tabular yang menunjukkan lima baris data awal dari dataset `calonpembelimobil.csv`, beserta nama-nama kolomnya.



```
[1]: from sklearn.model_selection import train_test_split

# Pisahkan fitur dan target
X = df[['Usia', 'Status', 'Kelamin', 'Memiliki Mobil', 'Penghasilan']]
y = df['Beli Mobil']

# Split data untuk training dan testing
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

[2]: from sklearn.linear_model import LogisticRegression

# Buat dan latih model
model = LogisticRegression()
model.fit(X_train, y_train)

[3]: from sklearn.metrics import accuracy_score, classification_report, confusion_matrix

# Prediksi data uji
y_pred = model.predict(X_test)

# Evaluasi hasil
print("Akurasi:", accuracy_score(y_test, y_pred))
```

1. **from sklearn.model_selection import train_test_split**

Mengimpor fungsi `train_test_split` dari pustaka Scikit-learn (`sklearn`). Fungsi ini digunakan untuk membagi dataset menjadi subset pelatihan (training) dan pengujian (testing).

2. **X = df[['Usia', 'Status', 'Kelamin', 'Memiliki Mobil', 'Penghasilan']]**

Membuat variabel `X` yang berisi Fitur (Features) atau variabel independen (kolom-kolom yang akan digunakan model untuk memprediksi).

3. **y = df['Beli Mobil']**

Membuat variabel `y` yang berisi Target atau variabel dependen. Ini adalah kolom yang akan diprediksi oleh model (dalam kasus ini: apakah seseorang akan 'Beli Mobil' atau tidak).

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

Melakukan pemisahan data:

`X` dan `y` dibagi menjadi empat bagian.

- `test_size=0.2` berarti 20% data akan dialokasikan untuk pengujian (`X_test`, `y_test`), dan

80% untuk pelatihan (X_{train} , y_{train}).

- `random_state=42` memastikan bahwa pemisahan data akan selalu menghasilkan hasil yang sama setiap kali kode dijalankan (untuk reproduktibilitas).

5. `from sklearn.linear_model import LogisticRegression`

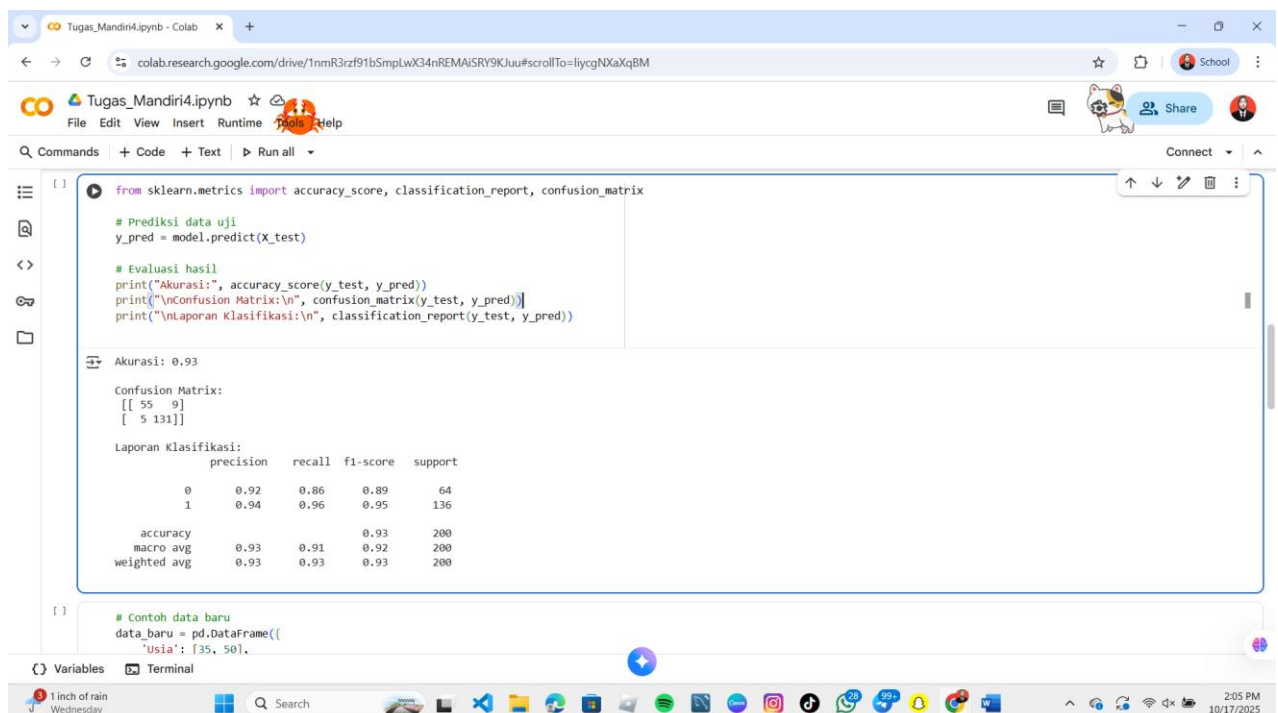
Mengimpor kelas `LogisticRegression` dari pustaka Scikit-learn.

6. `model = LogisticRegression()`

Membuat sebuah objek model Regresi Logistik dan menyimpannya dalam variabel `model`.

7. `model.fit(X_train, y_train)`

Melakukan pelatihan (training) model. Model belajar pola dari data fitur pelatihan (X_{train}) dan label target pelatihan (y_{train}). Hasil keluaran dari baris ini adalah objek `LogisticRegression()` yang telah dilatih.



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

# Prediksi data uji
y_pred = model.predict(X_test)

# Evaluasi hasil
print("Akurasi:", accuracy_score(y_test, y_pred))
print("\nConfusion Matrix:\n", confusion_matrix(y_test, y_pred))
print("\nLaporan Klasifikasi:\n", classification_report(y_test, y_pred))
```

Akurasi: 0.93

Confusion Matrix:

```
[[ 55  9]
 [  5 131]]
```

Laporan Klasifikasi:

	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

```
# Contoh data baru
data_baru = pd.DataFrame({
    "usia": [35, 50],
```

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

Mengimpor tiga fungsi metrik evaluasi dari Scikit-learn

2. `y_pred = model.predict(X_test)`

Menggunakan model Regresi Logistik yang telah dilatih (`model`) untuk membuat prediksi pada data fitur pengujian (X_{test}). Hasil prediksi (misalnya 0 atau 1) disimpan dalam variabel `y_pred`.

3. `print("Akurasi:", accuracy_score(y_test, y_pred))`

Menghitung dan mencetak Akurasi model.

4. `print("\nConfusion Matrix:\n", confusion_matrix(y_test, y_pred))`

Menghitung dan mencetak Matriks Konfusi.

5. `print("\nLaporan Klasifikasi:\n", classification_report(y_test, y_pred))`

Menghitung dan mencetak Laporan Klasifikasi. Ini memberikan metrik yang lebih rinci

untuk setiap kelas (0 dan 1).

Output:

1. Akurasi: 0.93

Penjelasan: Nilai 0.93 (atau 93%) menunjukkan bahwa model Regresi Logistik mampu memprediksi label target ("Beli Mobil" atau tidak) dengan benar sebanyak 93% dari total data pengujian.

2. Confusion Matrix: [[55 9] [5 131]]

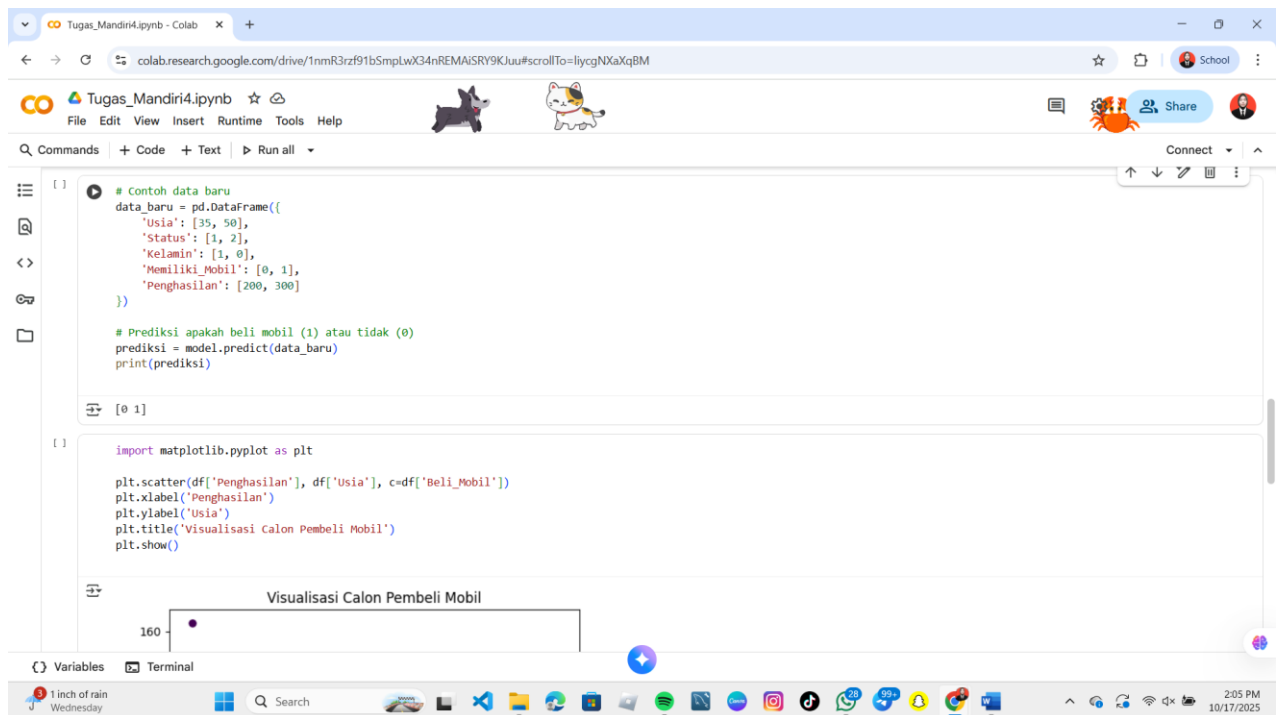
Penjelasan: Matriks 2x2 ini sangat penting untuk klasifikasi.

- Baris merepresentasikan nilai sebenarnya (y_{test}).
- Kolom merepresentasikan nilai prediksi (y_{pred}).
- [55]: True Negatives (TN). Jumlah kasus di mana kelas sebenarnya 0 (Tidak Beli Mobil) dan diprediksi 0 dengan benar.
- [9]: False Positives (FP). Jumlah kasus di mana kelas sebenarnya 0 tetapi diprediksi 1 (salah prediksi).
- [5]: False Negatives (FN). Jumlah kasus di mana kelas sebenarnya 1 (Beli Mobil) tetapi diprediksi 0 (salah prediksi).
- [131]: True Positives (TP). Jumlah kasus di mana kelas sebenarnya 1 dan diprediksi 1 dengan benar.

3. Laporan Klasifikasi

Laporan ini memberikan metrik evaluasi kinerja per kelas (0 dan 1):

- **support:** Jumlah kasus sebenarnya untuk setiap kelas di data pengujian (64 untuk kelas 0, 136 untuk kelas 1).
- **precision (Presisi):** Seberapa akurat prediksi positif model (dari semua yang diprediksi positif, berapa yang benar-benar positif).
 - **Kelas 0:** 0.92 (92% dari yang diprediksi '0' adalah benar-benar '0').
 - **Kelas 1:** 0.94 (94% dari yang diprediksi '1' adalah benar-benar '1').
- **recall (Sensitivitas/Cakupan):** Seberapa baik model menemukan semua kasus positif (dari semua yang sebenarnya positif, berapa yang diprediksi dengan benar).
 - Kelas 0: 0.86.
 - Kelas 1: 0.96.
- **f1-score:** Rata-rata harmonik dari precision dan recall. Nilai yang tinggi menunjukkan model seimbang.
- **macro avg:** Rata-rata sederhana dari metrik di setiap kelas (mengabaikan ketidakseimbangan kelas).
- **weighted avg:** Rata-rata dari metrik di setiap kelas, dibobotkan oleh support (mempertimbangkan ketidakseimbangan kelas).



1. `data_baru = pd.DataFrame({`

Membuat DataFrame pandas baru bernama `data_baru`. DataFrame ini berisi dua baris (dua individu) dan lima kolom fitur ('Usia', 'Status', 'Kelamin', 'Memiliki Mobil', 'Penghasilan') yang diperlukan oleh model untuk prediksi.

2. `prediksi = model.predict(data_baru)`

Menggunakan model yang sudah dilatih untuk membuat prediksi pada data baru (`data_baru`). Hasil prediksi (0 atau 1) disimpan dalam variabel `prediksi`.

3. `print(prediksi)`

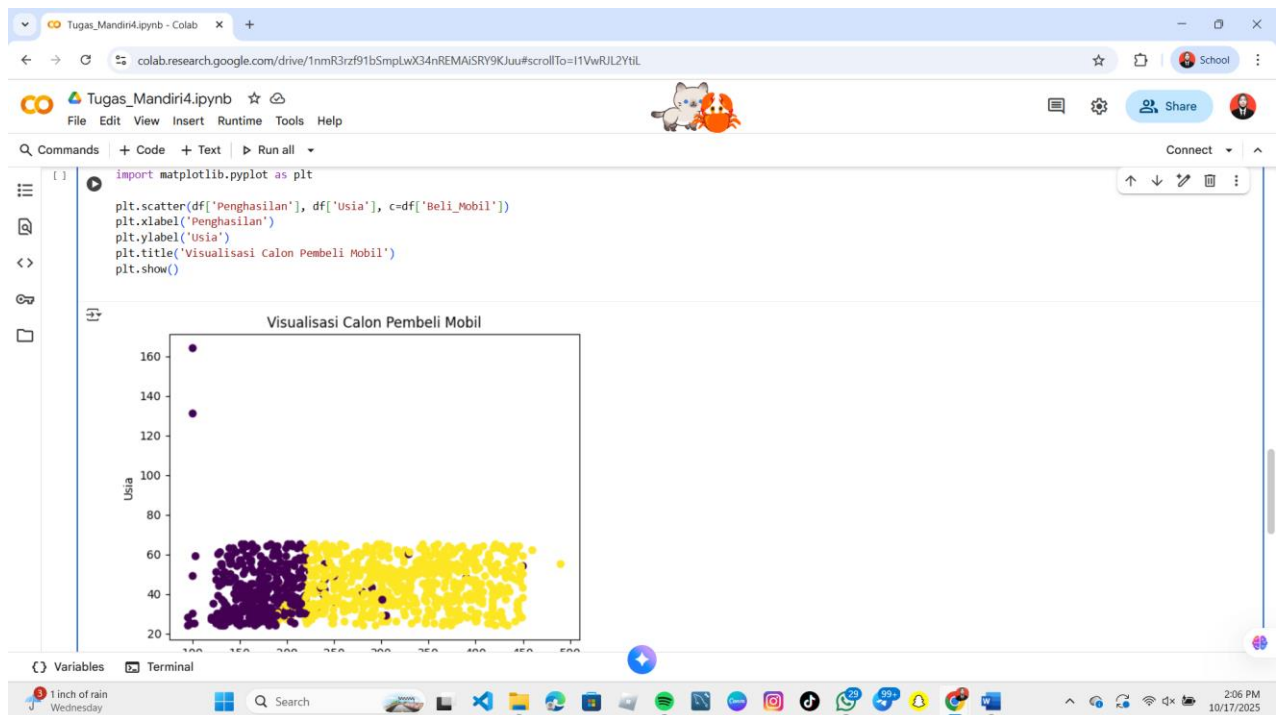
Mencetak isi variabel `prediksi`.

Output:

[0 1]

Penjelasan: Array ini adalah hasil prediksi model untuk dua individu di dalam `data_baru`.

- 0: Model memprediksi bahwa individu pertama Tidak akan membeli mobil.
- 1: Model memprediksi bahwa individu kedua Akan membeli mobil.



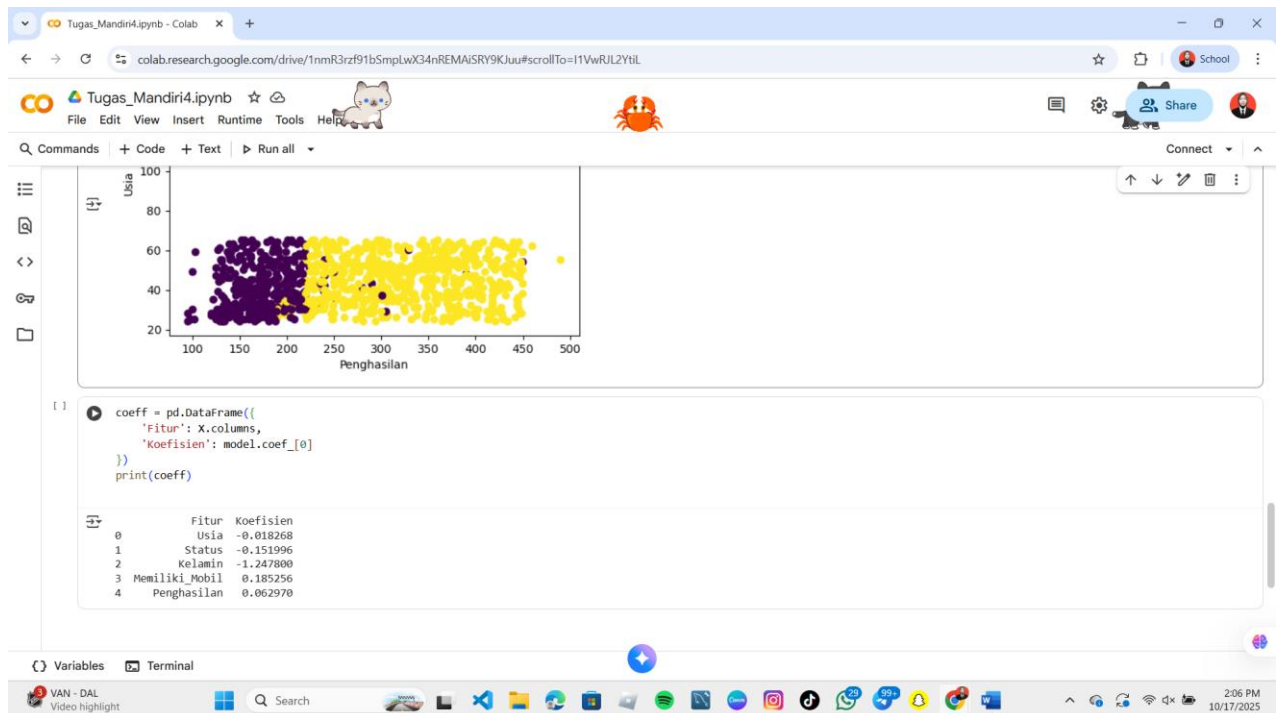
1. **import matplotlib.pyplot as plt**
Mengimpor modul pyplot dari pustaka Matplotlib dengan alias plt. Pustaka ini digunakan untuk menggambar grafik.
2. **plt.scatter(df['Penghasilan'], df['Usia'], c=df['Beli Mobil'])**
Membuat Scatter Plot (diagram sebaran):
 - Sumbu X: 'Penghasilan'.
 - Sumbu Y: 'Usia'.
3. **plt.xlabel('Penghasilan')**
Memberi label pada Sumbu X dengan teks 'Penghasilan'.
4. **plt.ylabel('Usia')**
Memberi label pada Sumbu Y dengan teks 'Usia'.
5. **plt.title('Visualisasi Calon Pembeli Mobil')**
Memberi judul pada plot.
6. **plt.show()**
Menampilkan plot di notebook.

Output:

Tampilannya adalah Scatter Plot berjudul "Visualisasi Calon Pembeli Mobil" yang menunjukkan:

- **Pengelompokan Data:** Terlihat bahwa data terbagi menjadi dua kelompok warna (misalnya ungu/hitam dan kuning) yang merepresentasikan kelas target (Beli Mobil: 0 atau 1).
 - Kelompok Usia rendah (sekitar 20-60 tahun) dan Penghasilan rendah (sekitar 100-250) cenderung didominasi oleh satu kelas (kemungkinan Tidak Beli Mobil, 0).

- Kelompok Usia rendah dan Penghasilan tinggi (sekitar 250-500) cenderung didominasi oleh kelas lainnya (kemungkinan Beli Mobil, 1).
- **Pengecualian/Outlier:** Terdapat beberapa titik di Usia yang sangat tinggi (sekitar 130-160) yang terlihat terpisah dari kelompok utama.



1. `coeff = pd.DataFrame({`

Membuat DataFrame baru bernama coeff untuk menampilkan koefisien model.

2. `'Fitur': X.columns`

Mengisi kolom pertama dengan nama-nama fitur yang digunakan untuk pelatihan (diambil dari kolom X yang sudah didefinisikan sebelumnya).

3. `'Koefisien': model.coef_[0]`

Mengisi kolom kedua dengan koefisien yang dipelajari oleh model Regresi Logistik. Karena model ini memprediksi satu kelas (klasifikasi biner), koefisien diambil dari baris pertama ([0]).

4. `print(coeff)`

Mencetak DataFrame coeff.

Output:

Tabel ini menampilkan seberapa besar dan ke arah mana setiap fitur memengaruhi prediksi target (Beli Mobil). Dalam Regresi Logistik, tanda (positif/negatif) dan nilai absolut koefisien adalah kuncinya:

- Koefisien Positif (Misalnya: Penghasilan, Memiliki Mobil): Peningkatan nilai fitur ini meningkatkan probabilitas seseorang untuk membeli mobil (kelas 1).
- Koefisien Negatif (Misalnya: Usia, Status, Kelamin): Peningkatan nilai fitur ini menurunkan probabilitas seseorang untuk membeli mobil.
- Magnitudo (Nilai Absolut): Nilai absolut yang lebih besar menunjukkan dampak yang lebih kuat. Contohnya, Kelamin memiliki nilai absolut terbesar (-1.2478), yang menunjukkan bahwa fitur ini memiliki pengaruh paling signifikan terhadap keputusan

pembelian dibandingkan fitur lainnya dalam model ini.

Kesimpulan hasil implementasi algoritma: Berdasarkan implementasi yang telah dilakukan, model Regresi Logistik berhasil dilatih untuk memprediksi keputusan "Beli Mobil" dengan tingkat Akurasi 93% pada data uji. Analisis Matriks Konfusi menunjukkan kinerja yang kuat, terutama dalam memprediksi kelas positif (Beli Mobil), ditunjukkan oleh True Positives (TP) 131 dan Recall 96% untuk kelas 1, serta Precision 94%. Sementara itu, analisis Koefisien Model mengungkapkan bahwa fitur Kelamin memiliki dampak negatif paling signifikan terhadap keputusan pembelian, diikuti oleh Status dan Usia (semakin tinggi nilainya, semakin kecil kemungkinan beli). Sebaliknya, fitur Penghasilan dan Memiliki Mobil menunjukkan korelasi positif, meningkatkan probabilitas seseorang untuk membeli mobil. Visualisasi data memperkuat temuan ini dengan jelas memisahkan kelompok pembeli dan non-pembeli berdasarkan kombinasi fitur Usia dan Penghasilan.

LINK GITHUB UPLOAD TUGAS : <https://github.com/Yurida26/Machine-Learning/tree/1bd719dcce05bdf74766b327c47343c012b3a3a3/Praktikum4>

