

# Lembar Jawaban UTS

## Machine Learning

Nama : AWANDA BINTANG PAMUNGKAS

NIM : 5210411366

Kelas : C

### Pernyataan:

Dengan mengerjakan ujian ini, maka saya menyatakan bahwa semua jawaban **SAYA KERJAKAN SENDIRI** tanpa bekerjasama maupun meminta bantuan siapapun. Saya bersedia diberi **nilai E** jika pernyataan ini terbukti salah.

Tanda tangan:

1. Convert data dari table menjadi bentuk data tabular yaitu excel atau CSV

The screenshot shows a Jupyter Notebook interface with the following details:

- Title Bar:** UTS\_5210411366\_AWANDA BINTANG PAMUNGKAS.ipynb
- File Menu:** File, Edit, View, Insert, Runtime, Tools, Help, All changes saved
- Toolbar:** Comment, Share, RAM, Disk
- Code Cell:** Contains Python code for importing libraries (pandas, numpy, matplotlib, sklearn, seaborn), reading the dataset from 'data\_uts.xlsx', displaying the first few rows of the DataFrame, and checking for missing values.
- Output Cell:** Displays the first 20 rows of the DataFrame 'df' with columns: Gender, Age, Annual\_Salary, Purchased.
- Bottom Status Bar:** Shows Disk usage (81.55 GB available) and a completed execution message at 11:26 AM.

	Gender	Age	Annual_Salary	Purchased
0	Male	35.0	20000.0	NO
1	Male	40.0	43500.0	NO
2	Male	49.0	74000.0	NO
3	Male	40.0	107500.0	YES
4	Male	25.0	79000.0	NO
5	Female	47.0	33500.0	NO
6	NaN	46.0	132500.0	YES
7	Male	42.0	NaN	YES
8	Female	30.0	84500.0	NO
9	Male	NaN	52000.0	NO
10	Male	42.0	80000.0	NO
11	Male	47.0	23000.0	NaN
12	Female	32.0	72500.0	NO
13	NaN	27.0	57000.0	NO
14	Female	42.0	108000.0	YES
15	Female	NaN	149000.0	YES
16	Male	39.0	75000.0	NO
17	Male	35.0	NaN	NO
18	Male	46.0	79000.0	YES
19	Female	39.0	134000.0	NO

## 2. Perbaiki data yg kosong atau missing pada setiap kolom

### MISSING VALUE PADA KOLOM “GENDER”

```
File Edit View Insert Runtime Tools Help All changes saved
+ Code + Text
1 # CEK MISSING VALUE
2
3 df.isnull().values.any()
True

[ ] 1 # CEK JUMLAH MISSING VALUE
2
3 df.isnull().sum().sum()

7

1 # CEK MISSING VALUE PADA KOLOM GENDER
2
3 df['Gender']

0    Male
1    Male
2    Male
3    Male
4    Male
5    Female
6    NaN
7    Male
8    Female
9    Male
10   Male
11   Male
12   Female
13   NaN
14   Female
15   Female
16   Male
17   Male
18   Male
19   Female
Name: Gender, dtype: object

[ ] 1 df['Gender'].isnull()

0    False
1    False
2    False
3    False
4    False
5    False
6    True
7    False
8    False
9    False
10   False
11   False
12   False
13   True
14   False
15   False
16   False
17   False
18   False
19   False
Name: Gender, dtype: bool
```

This screenshot shows a Jupyter Notebook interface with a dark theme. The code cell at the top contains three lines of Python code: `df.isnull().values.any()` which returns `True`, indicating there are missing values; `df.isnull().sum().sum()` which returns 7, indicating there are 7 missing values in total; and `df['Gender']` which displays the 'Gender' column with 20 entries: Male (15), Female (4), and one NaN value. The bottom cell contains the command `df['Gender'].isnull()` followed by its output, which shows 19 rows with `False` and one row with `True` (index 6), indicating one missing value in the 'Gender' column.

```
File Edit View Insert Runtime Tools Help All changes saved
+ Code + Text
1 df['Gender'].isnull()

0    False
1    False
2    False
3    False
4    False
5    False
6    True
7    False
8    False
9    False
10   False
11   False
12   False
13   True
14   False
15   False
16   False
17   False
18   False
19   False
Name: Gender, dtype: bool

[ ] 1 # CEK MISSING VALUE PADA KOLOM AGE
2
3 df['Age']

0    35.0
1    40.0
2    43.0
3    40.0
4    25.0
5    47.0
6    46.0
7    42.0
8    30.0
9    NaN
10   42.0
11   47.0
12   32.0
13   27.0
14   43.0
15   NaN
16   35.0
17   35.0
18   46.0
19   39.0
Name: Age, dtype: float64
```

This screenshot shows a Jupyter Notebook interface with a dark theme. The code cell at the top contains the command `df['Gender'].isnull()` followed by its output, which shows 19 rows with `False` and one row with `True` (index 6), indicating one missing value in the 'Gender' column. The bottom cell contains the command `df['Age']` followed by its output, which displays the 'Age' column with 20 entries ranging from 25.0 to 47.0, with one NaN value at index 9.

## MISSING VALUE PADA KOLOM “AGE”

The screenshot shows a Jupyter Notebook interface with the following code in the code cell:

```
[ ] 1 df['Age'].isnull()
0 False
1 False
2 False
3 False
4 False
5 False
6 False
7 False
8 False
9 True
10 False
11 False
12 False
13 False
14 False
15 True
16 False
17 False
18 False
19 False
Name: Age, dtype: bool
```

Below the code cell, the output cell displays:

```
1 # MENGECEK MISSING VALUE PADA KOLOM ANNUAL SALARY
2
3 df['Annual Salary']
0 20000.0
1 43500.0
2 74000.0
3 107500.0
4 79000.0
5 33500.0
6 132500.0
7 NaN
8 84500.0
9 52000.0
10 80000.0
11 23000.0
12 72500.0
13 57000.0
14 108000.0
15 149000.0
16 75000.0
17 NaN
18 79000.0
19 134000.0
Name: Annual Salary, dtype: float64
```

The status bar at the bottom indicates "0s completed at 11:26 AM".

## MISSING VALUE PADA KOLOM “ANNUAL SALARY”

The screenshot shows a Jupyter Notebook interface with the following code in the code cell:

```
[ ] 1 df['Annual Salary'].isnull()
0 False
1 False
2 False
3 False
4 False
5 False
6 False
7 True
8 False
9 False
10 False
11 False
12 False
13 False
14 False
15 False
16 False
17 True
18 False
19 False
Name: Annual Salary, dtype: bool
```

Below the code cell, the output cell displays:

```
1 # MENGECEK MISSING VALUE PADA KOLOM PURCHASED
2
3 df['Purchased']
0 NO
1 NO
2 NO
3 YES
4 NO
5 NO
6 YES
7 YES
8 NO
9 NO
10 NO
11 NaN
12 NO
13 NO
14 YES
15 YES
16 NO
17 NO
18 YES
19 NO
Name: Purchased, dtype: object
```

The status bar at the bottom indicates "0s completed at 11:26 AM".

## MISSING VALUE PADA KOLOM “PURCHASED”

The screenshot shows a Jupyter Notebook interface with the following code:

```
[ ] 1 df['Purchased'].isnull()
0    False
1    False
2    False
3    False
4    False
5    False
6    False
7    False
8    False
9    False
10   False
11   True
12   False
13   False
14   False
15   False
16   False
17   False
18   False
19   False
Name: Purchased, dtype: bool

[ ] 1 # HITUNG KEMBALI TOTAL MISSING VALUE PADA SETIAP KOLOM
2
3 df.isnull().sum()

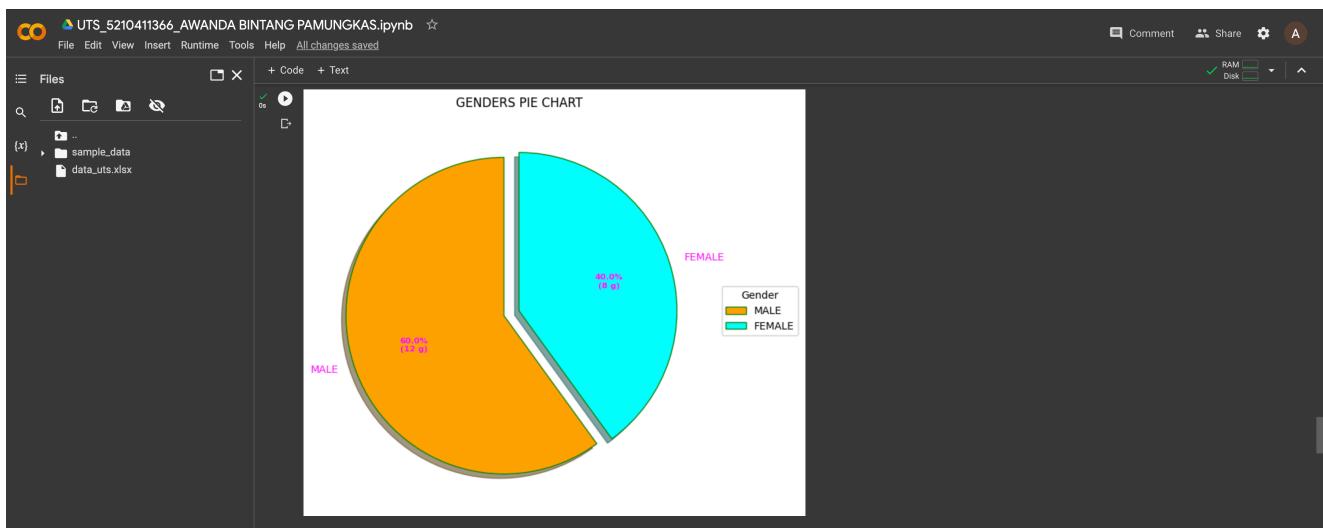
Gender      2
Age         2
Annual Salary 2
Purchased   1
dtype: int64

[ ] 1 # BANYAKNYA MISSING VALUES
2
3 df.isnull().sum().sum()

D 7
```

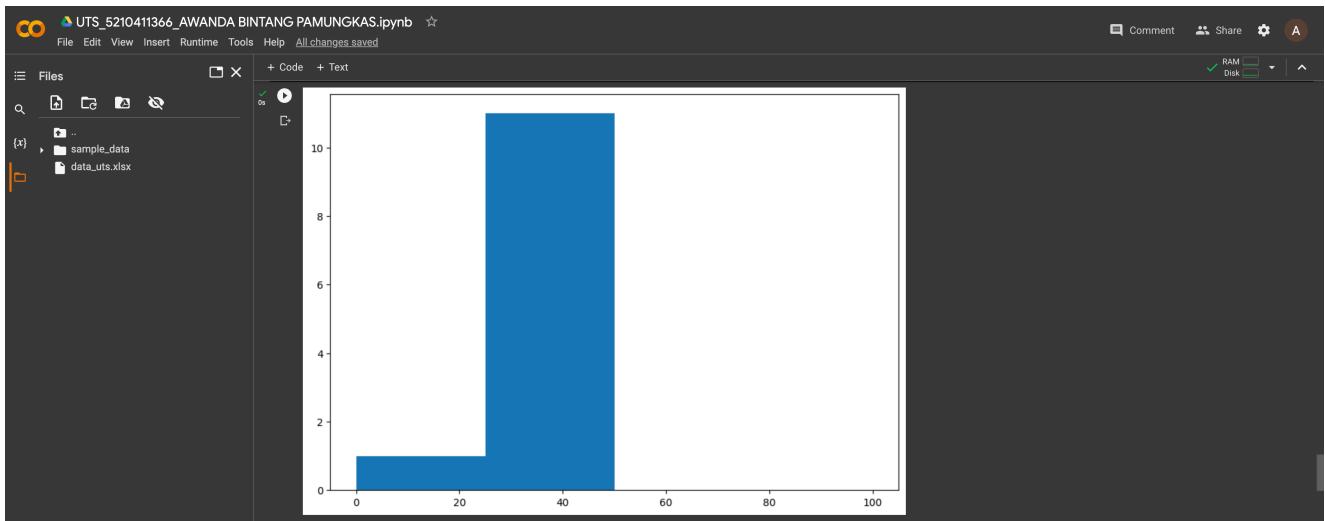
3. Visualisasikan setiap kolom dengan diagram yang cocok dan berikan alasan kenapa diagram itu cocok dengan kolom yang divisualisasikan

### Genders



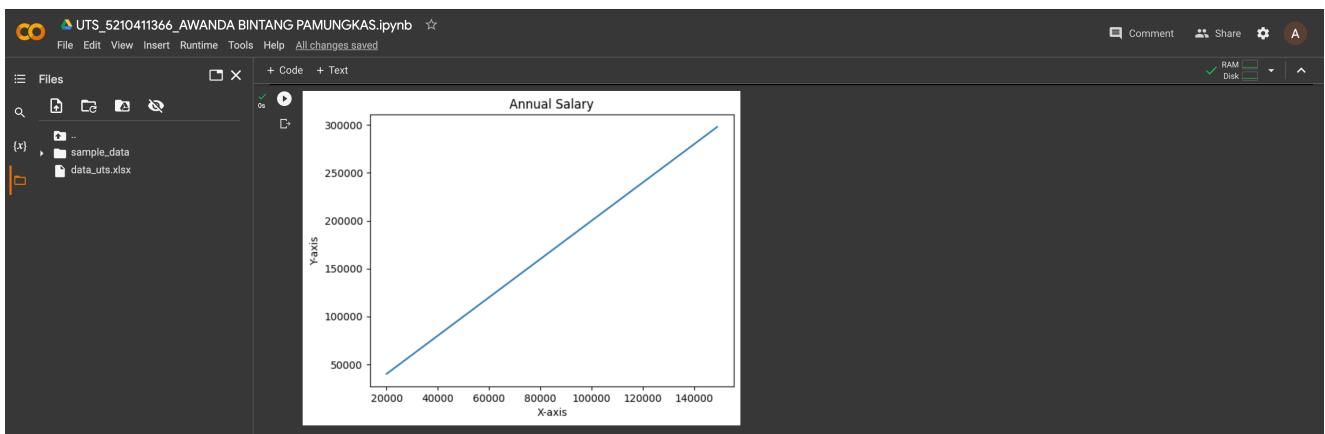
PENJELASAN : Kolom "Gender": Diagram pie atau bar chart dapat digunakan untuk memvisualisasikan proporsi atau jumlah individu berdasarkan jenis kelamin. Pie chart cocok jika hanya ada dua jenis kelamin yang terlibat, sementara bar chart lebih cocok jika lebih dari dua jenis kelamin yang terlibat.

## Age



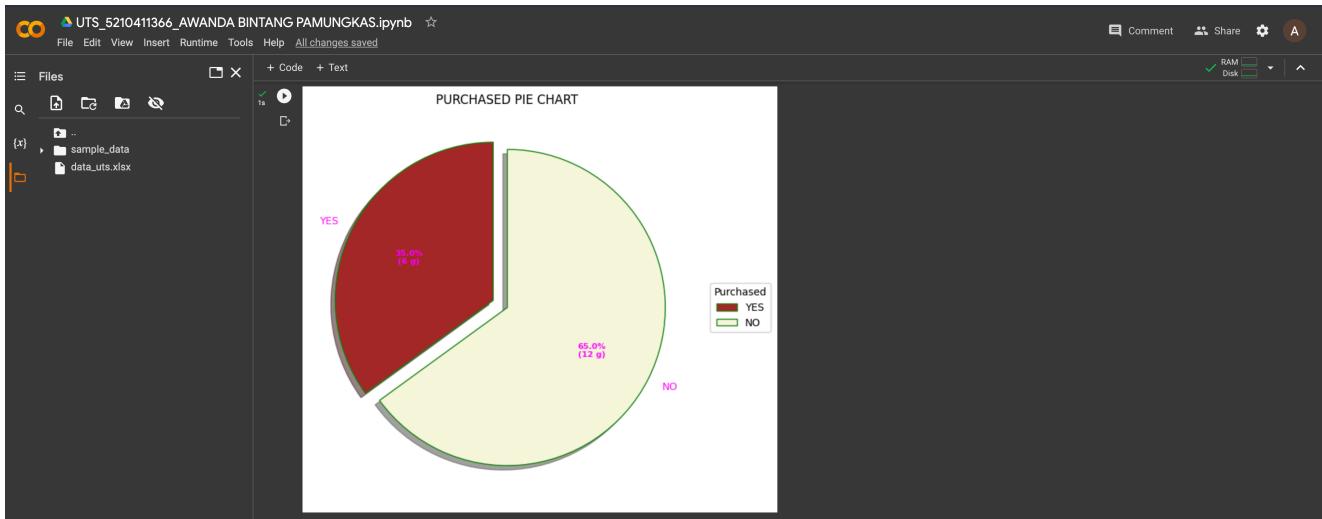
Kolom "Age": Histogram atau box plot dapat digunakan untuk memvisualisasikan distribusi usia dalam dataset. Histogram cocok untuk melihat frekuensi usia dalam interval tertentu, sementara box plot cocok untuk melihat kuartil, rentang, dan outlier dalam distribusi usia.

## Annual Salary



Kolom "AnnualSalary": Box plot atau line chart dapat digunakan untuk memvisualisasikan distribusi atau tren gaji tahunan dalam dataset. Box plot cocok untuk melihat kuartil, rentang, dan outlier dalam distribusi gaji, sementara line chart cocok untuk melihat tren gaji dari waktu ke waktu.

## Purchased



Kolom "Purchased": Bar chart atau pie chart dapat digunakan untuk memvisualisasikan jumlah pembelian atau proporsi individu yang melakukan pembelian. Bar chart cocok jika ingin mengetahui jumlah pembelian berdasarkan kategori tertentu, seperti jenis produk atau waktu, sementara pie chart cocok jika ingin mengetahui proporsi pembelian dalam dua kategori atau lebih.

Lakukan proses yang direkomendasikan oleh peneliti tersebut agar data bisa digunakan. Setelah proses pengolahan data selesai. Kemudian peneliti ingin mengklasifikasikan data pembelian modil atau tidak dengan menggunakan metode K-NN. Adapun kriteria penyusunan metode K-NN antara lain:

### 1. Nilai K sebesar 4

```
[25]: 1 # Import necessary modules
2 from sklearn.neighbors import KNeighborsClassifier
3 from sklearn.model_selection import train_test_split
4 from sklearn.datasets import load_iris
5
6 # Loading data
7 irisData = load_iris()
8
9 # Create feature and target arrays
10 X = irisData.data
11 y = irisData.target
12
13 # Split into training and test set
14 X_train, X_test, y_train, y_test = train_test_split(
15 |   |   |   | X, y, test_size = 0.2, random_state=42)
16
17 knn = KNeighborsClassifier(n_neighbors=7)
18
19 knn.fit(X_train, y_train)
20
21 # Predict on dataset which model has not seen before
22 print(knn.predict(X_test))
23
[1 0 2 1 1 0 1 2 2 1 2 0 0 0 1 2 1 1 2 0 2 0 2 2 2 2 2 0 0]
```

SPLITTING DATASET

```
[22]: 1 # import modules
2 import pandas as pd
3 from sklearn.linear_model import LinearRegression
4 from sklearn.model_selection import train_test_split
5
6 # read the dataset
7 df = pd.read_excel('data_uts.xlsx')
8
9 # get the locations
10 X = df.iloc[:, :-1]
11 y = df.iloc[:, -1]
12
13 # split the dataset
14 X_train, X_test, y_train, y_test = train_test_split(
15 |   |   |   | X, y, test_size=0.05, random_state=0)
```

**X\_train**

```
[6. , 2.7, 5.1, 1.6],
[6. , 2.6, 5.6, 1.4],
[7.7, 3. , 6.1, 2.3],
[5.5, 2.5, 4. , 1.3],
[4.4, 2.9, 1.4, 0.2],
[4.3, 3. , 1.1, 0.1],
[6. , 2.2, 5. , 1.5],
[7.2, 3.2, 6. , 1.8],
[4.6, 3.1, 1.5, 0.2],
[5.1, 3.5, 1.4, 0.3],
[4.4, 3.1, 1.3, 0.2],
[6.5, 2.5, 4.5, 1.5],
[6.3, 3.4, 5.6, 2.4],
[4.6, 3.4, 1.4, 0.3],
[6.8, 3. , 5.5, 2.1],
[6.3, 3.3, 6. , 2.5],
[4.7, 3.2, 1.3, 0.2],
[6.1, 2.9, 4.7, 1.4],
[6.5, 2.8, 4.6, 1.5],
[6.2, 2.8, 4.8, 1.8],
[7. , 3.2, 4.7, 1.4],
[6.4, 2.2, 5.2, 2.0],
[5.1, 3.5, 5.4, 2.2],
[6.9, 3.1, 5.4, 2.1],
[5.9, 3. , 4.2, 1.5],
[6.5, 3. , 5.2, 2. ],
[5.7, 2.6, 3.5, 1. ],
[5.2, 2.7, 3.9, 1.4],
[6.1, 3. , 4.6, 1.4],
[4.5, 2.3, 1.3, 0.3],
[6.6, 2.9, 4.6, 1.3],
[5.5, 2.6, 4.4, 1.2],
[5.3, 3.7, 1.5, 0.2],
[5.6, 2.9, 4.5, 1.3],
[7.3, 2.9, 6.3, 1.8],
[6.7, 3.3, 5.7, 2.1],
[5.1, 3.7, 1.5, 0.4],
[4.9, 2.4, 3.3, 1. ],
[6.7, 3.3, 5.7, 2.5],
[7.2, 3. , 5.8, 1.6],
[4.9, 3.6, 1.4, 0.1],
[6.7, 3.1, 5.6, 2.4],
[4.9, 3.2, 1.4, 0.1],
[6.9, 3.1, 5.9, 2.3],
[7.4, 2.0, 6.1, 1.9],
[6.3, 2.9, 5.6, 1.8],
[5.7, 2.8, 4.1, 1.3],
[6.5, 3. , 5.5, 1.8],
[6.3, 2.3, 4.4, 1.3],
[6.4, 2.9, 4.3, 1.3],
```

**X\_test**

```
[27] array([[6.1, 2.8, 4.7, 1.2],
[5.7, 3.8, 1.7, 0.3],
[7.7, 2.6, 6.9, 2.3],
[6. , 2.9, 4.5, 1.5],
[6.8, 2.8, 4.8, 1.4],
[5.4, 3.4, 1.5, 0.4],
[5.6, 2.9, 3.6, 1.3],
[6.9, 3.1, 5.3, 2.0],
[6.2, 2.2, 4.5, 1.5],
[5.8, 2.7, 3.9, 1.2],
[6.5, 3.2, 5.1, 2. ],
[4.8, 3. , 1.4, 0.1],
[5.5, 3.5, 1.3, 0.2],
[4.9, 3.1, 1.5, 0.1],
[5.1, 3.8, 1.5, 0.3],
[6.3, 3.3, 4.7, 1.6],
[6.5, 3. , 5.8, 2.2],
[5.6, 2.9, 3.9, 1.1],
[5.7, 2.0, 6.1, 1.3],
[6.4, 2.0, 5.6, 2.2],
[4.7, 3.2, 1.6, 0.2],
[6.1, 3. , 4.9, 1.8],
[5. , 3.4, 1.6, 0.4],
[6.4, 2.8, 5.6, 2.1],
[7.9, 3.8, 6.4, 2. ],
[6.7, 3. , 5.2, 2.3],
[6.7, 2.5, 5.8, 1.8],
[6.8, 3.2, 5.9, 2.3],
[4.8, 3. , 1.4, 0.3],
[4.8, 3.1, 1.6, 0.2]])
```

**MODELLING**

```
[48] 1 from sklearn.neighbors import KNeighborsClassifier
2 k=4
3 #Train Model and Predict
4 knn = KNeighborsClassifier(n_neighbors = k).fit(X_train,y_train)
```

**y\_test**

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

**y\_train**

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

**y\_test**

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

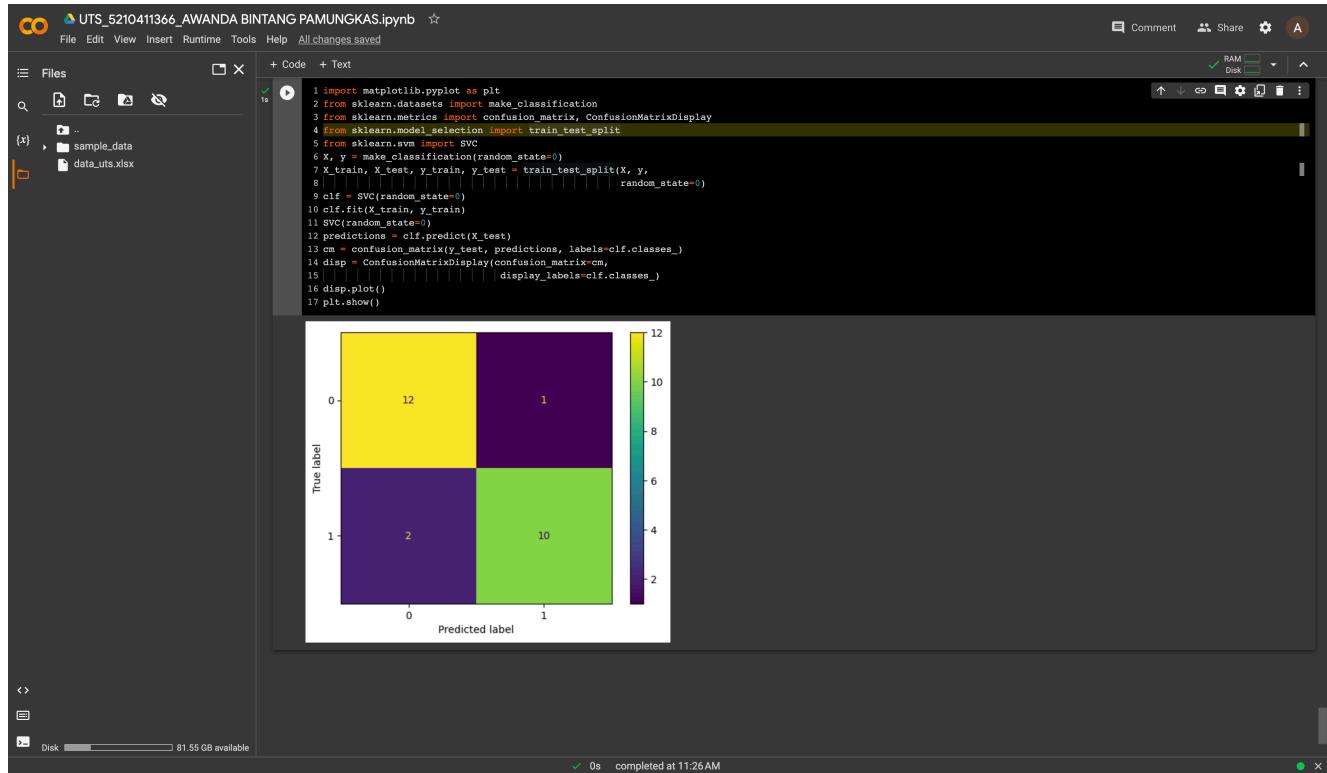
**y\_train**

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

```
[51] 1 from sklearn import metrics
2
3 y_predict = knn.predict(X_test)
4 print(y_predict)
```

```
[52] 1 print("Test set Accuracy: ", metrics.accuracy_score(y_test, y_predict))
Test set Accuracy: 0.68
```

### 3.



The screenshot shows a Jupyter Notebook interface with a code cell containing Python code for generating a confusion matrix. The code imports necessary libraries (matplotlib.pyplot, sklearn.datasets, sklearn.metrics, sklearn.model\_selection, and sklearn.svm) and performs a train-test split on a dataset. It then creates an SVC model, fits it to the training data, and uses it to predict the test data. Finally, it generates a ConfusionMatrixDisplay object and plots it. The resulting confusion matrix is a 2x2 grid with labels 0 and 1 on both axes. The values in the matrix are: True label 0, Predicted label 0: 12; True label 0, Predicted label 1: 1; True label 1, Predicted label 0: 2; True label 1, Predicted label 1: 10. A color bar on the right indicates values from 2 (dark purple) to 12 (yellow).

```
import matplotlib.pyplot as plt
from sklearn.datasets import make_classification
from sklearn.metrics import make_classification
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC
X, y = make_classification(random_state=42)
X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=0)
clf = SVC(random_state=42)
clf.fit(X_train, y_train)
SVC(random_state=42)
predictions = clf.predict(X_test)
cm = confusion_matrix(y_test, predictions, labels=clf.classes_)
disp = ConfusionMatrixDisplay(confusion_matrix=cm,
                                display_labels=clf.classes_)
disp.plot()
plt.show()
```

$$\text{Akurasi} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{FP} + \text{FN} + \text{TN}}$$

$$= \frac{12 + 10}{12 + 1 + 2 + 10}$$

$$= \frac{22}{25}$$

$$= 0.88$$

