

MACHINE LEARNING

(Tugas 1)



Disusun Oleh:

Nama: Sifa Maryam Rahman

NPM: 41155050210031

Kelas: TIF A1

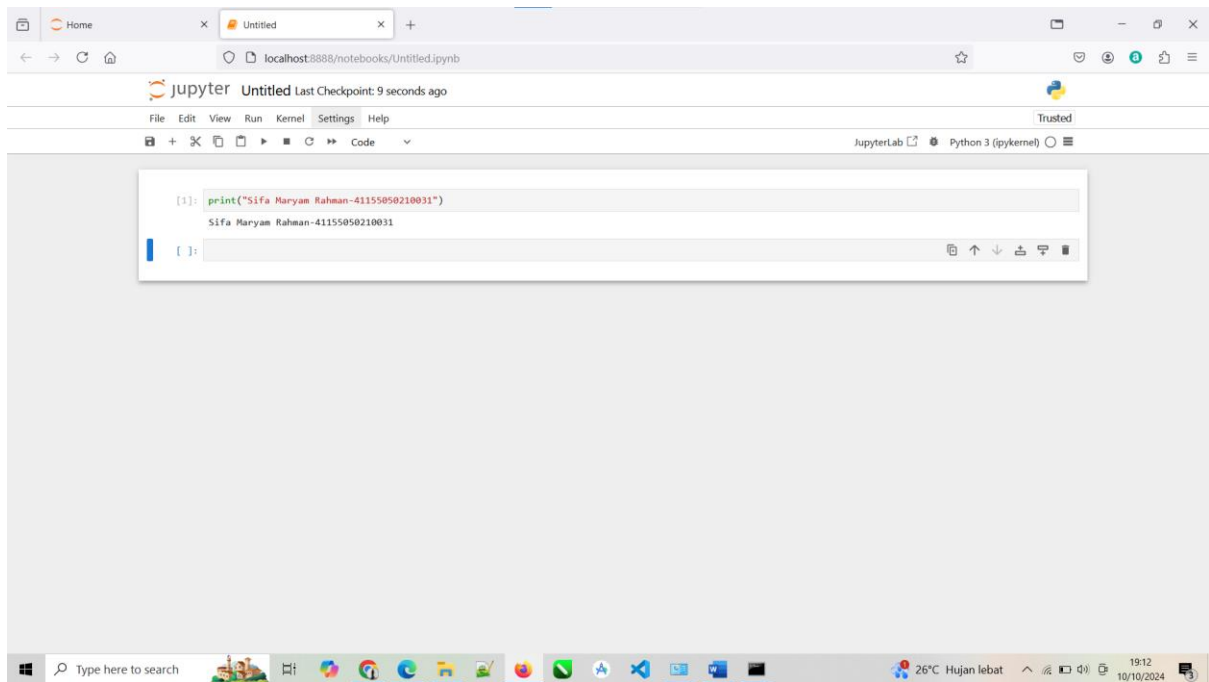
TEKNIK INFORMATIKA

FAKULTAS TEKNIK

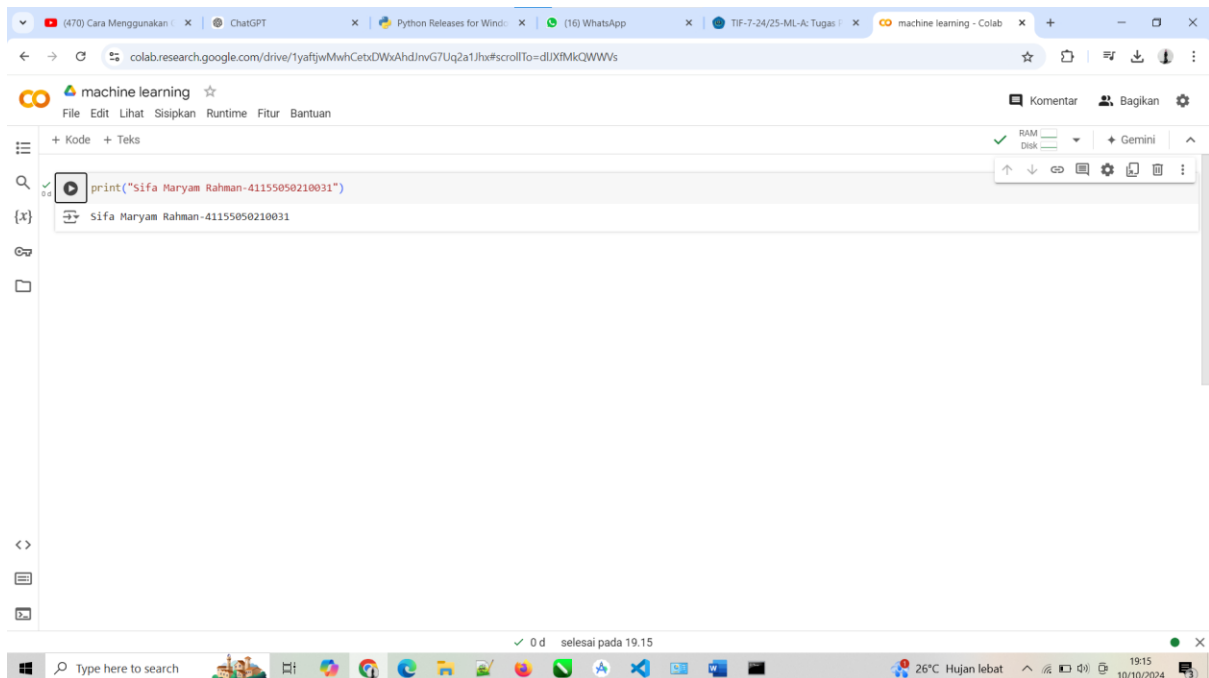
UNIVERSITAS LANGLANGBUANA

2024

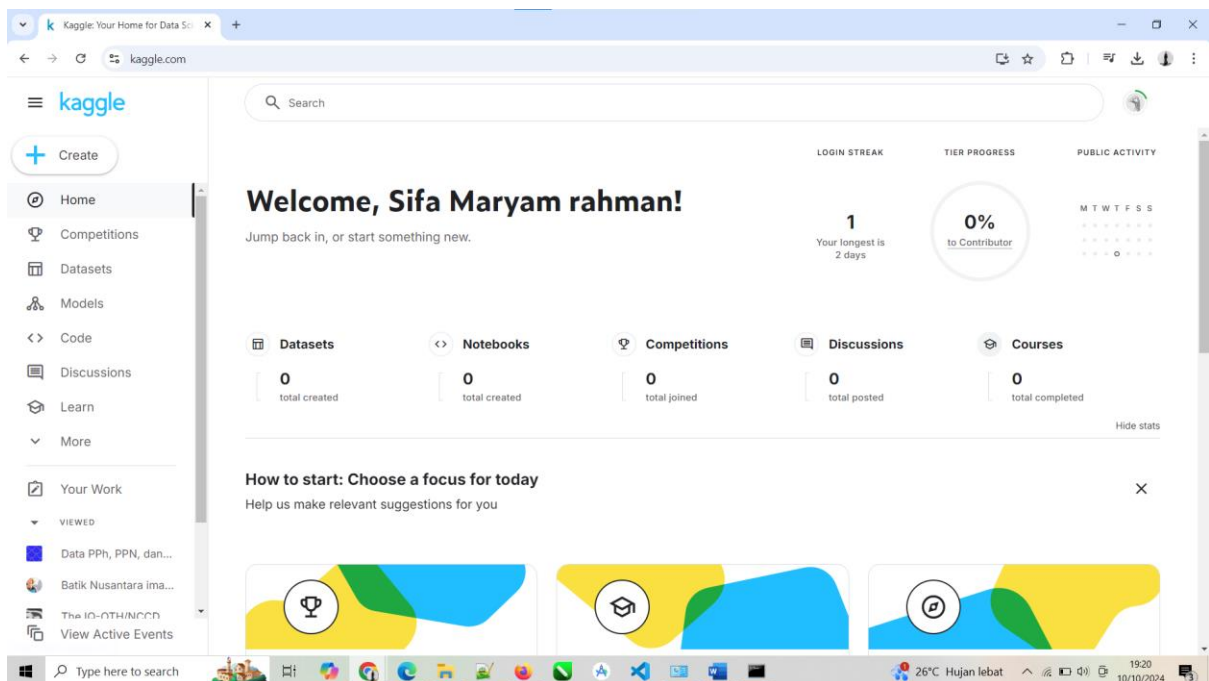
1. Hasil dari Jupiter Notebook



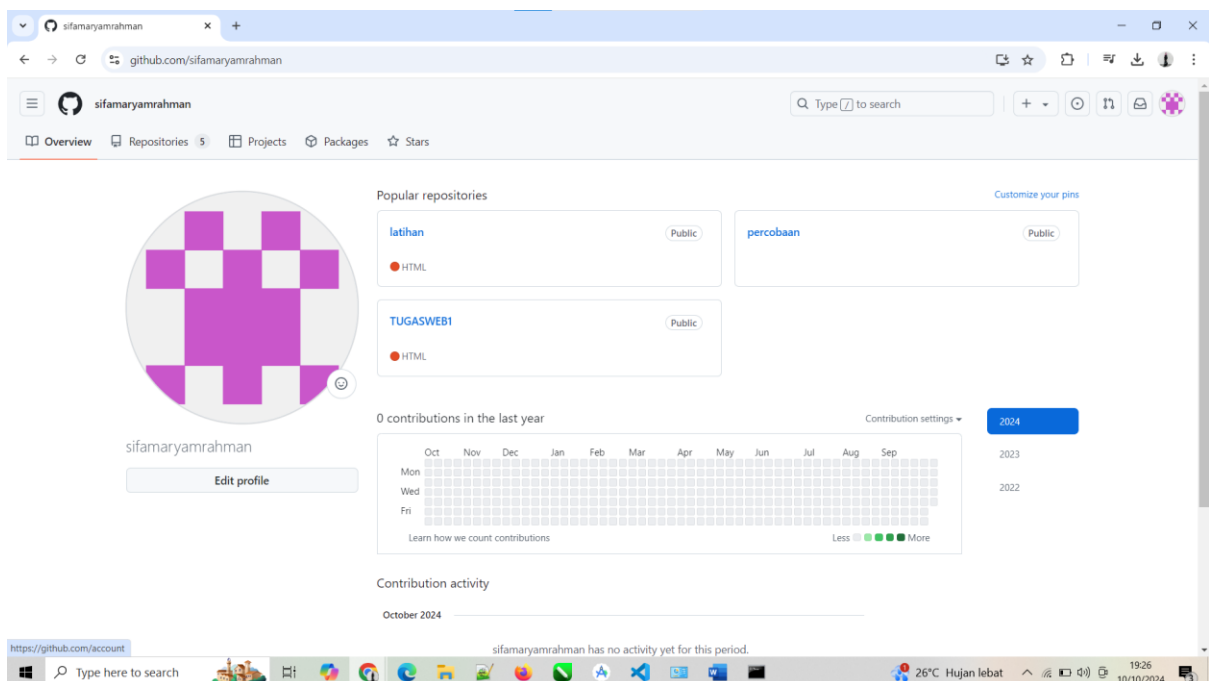
2. Hasil dari Google Colab



3. Hasil dari kaggle

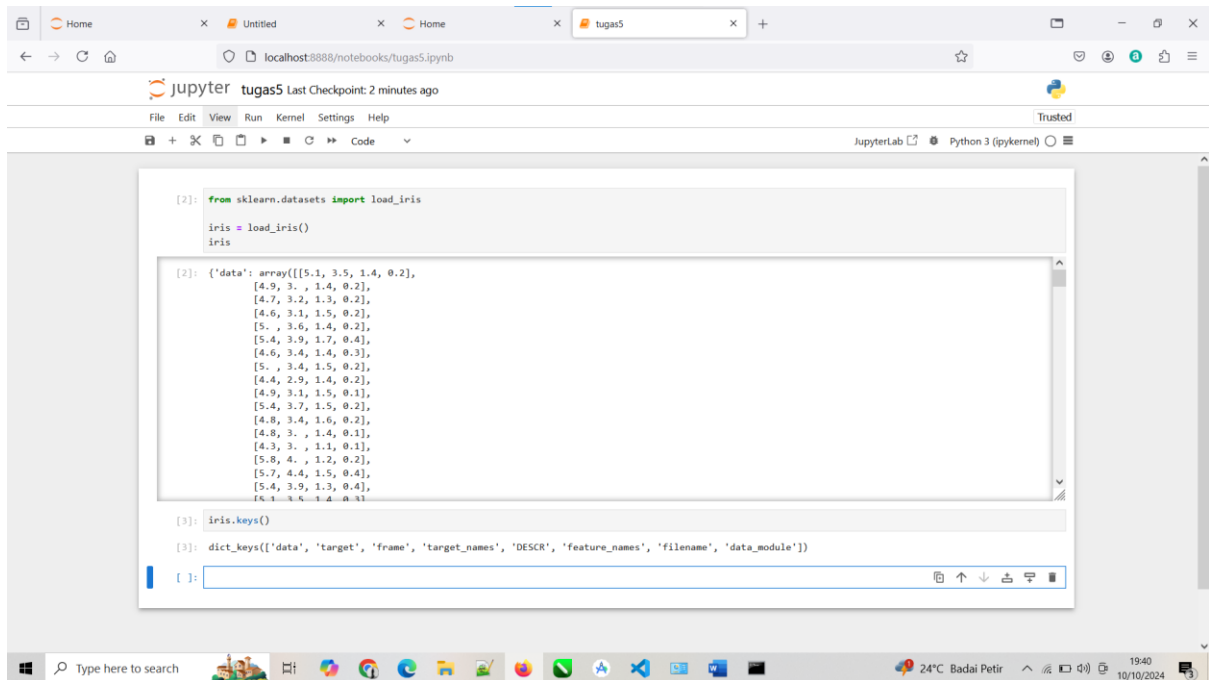


4. Hasil dari Github



5. Berikut adalah hasil dari beberapa praktek

5.1. Load sample dataset



The screenshot shows a JupyterLab notebook interface with a browser window at localhost:8888. The notebook has a single code cell with the following Python code:

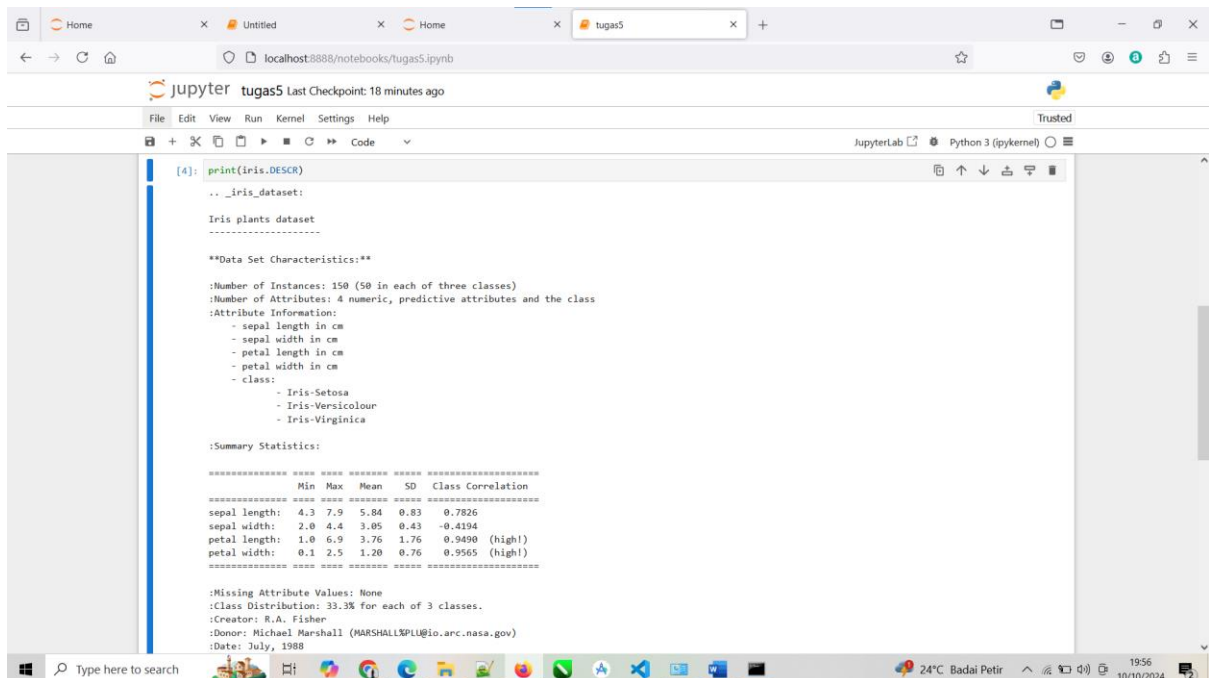
```
[2]: from sklearn.datasets import load_iris
iris = load_iris()
iris
```

The output of the code cell is displayed below the code:

```
[2]: {'data': array([[5.1, 3.5, 1.4, 0.2],
[4.9, 3. , 1.4, 0.2],
[4.7, 3.2, 1.3, 0.2],
[4.6, 3.1, 1.5, 0.2],
[5. , 3.6, 1.4, 0.2],
[5.4, 3.9, 1.7, 0.4],
[4.6, 3.4, 1.4, 0.3],
[5. , 3.4, 1.5, 0.2],
[4.4, 2.9, 1.4, 0.2],
[4.9, 3.1, 1.5, 0.1],
[5.4, 3.7, 1.5, 0.2],
[4.8, 3.4, 1.6, 0.2],
[4.8, 3. , 1.4, 0.1],
[4.3, 3. , 1.1, 0.1],
[5.8, 4. , 1.2, 0.2],
[5.7, 4.4, 1.5, 0.4],
[5.4, 3.9, 1.3, 0.4],
[5.1, 3.5, 1.4, 0.1]]),
'target': array([0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1, 2, 0]),
'frame': None,
'target_names': None,
'DESCR': None,
'feature_names': None,
'filename': 'iris.data',
'data_module': None}
```

The notebook interface includes a menu bar (File, Edit, View, Run, Kernel, Settings, Help) and a toolbar with icons for file operations, running, and saving. The bottom status bar shows the system clock as 19:40 on 10/10/2024.

5.2. Metadata | Deskripsi dari sample dataset



The screenshot shows a JupyterLab notebook interface with a browser window at localhost:8888. The notebook has a single code cell with the following Python code:

```
[4]: print(iris.DESCR)
.. _iris_dataset:
```

The output of the code cell is displayed below the code:

```
Iris plants dataset
-----

**Data Set Characteristics:**

: Number of Instances: 150 (50 in each of three classes)
: Number of Attributes: 4 numeric, predictive attributes and the class
: Attribute Information:
  - sepal length in cm
  - sepal width in cm
  - petal length in cm
  - petal width in cm
  - class:
    - Iris-Setosa
    - Iris-Versicolour
    - Iris-Virginica

: Summary Statistics:

=====
              Min  Max   Mean  SD   Class Correlation
=====
sepal length:  4.3  7.9   5.84   0.83    0.7826
sepal width:   2.0  4.4   3.05   0.43   -0.4194
petal length:   1.0  6.9   3.76   1.76    0.9490 (high)
petal width:   0.1  2.5   1.20   0.76    0.9565 (high)
=====

: Missing Attribute Values: None
: Class Distribution: 33.3% for each of 3 classes.
: Creator: R.A. Fisher
: Donor: Michael Marshall (MARSHALLXPLU@io.arc.nasa.gov)
: Date: July, 1988
```

The notebook interface includes a menu bar (File, Edit, View, Run, Kernel, Settings, Help) and a toolbar with icons for file operations, running, and saving. The bottom status bar shows the system clock as 19:56 on 10/10/2024.

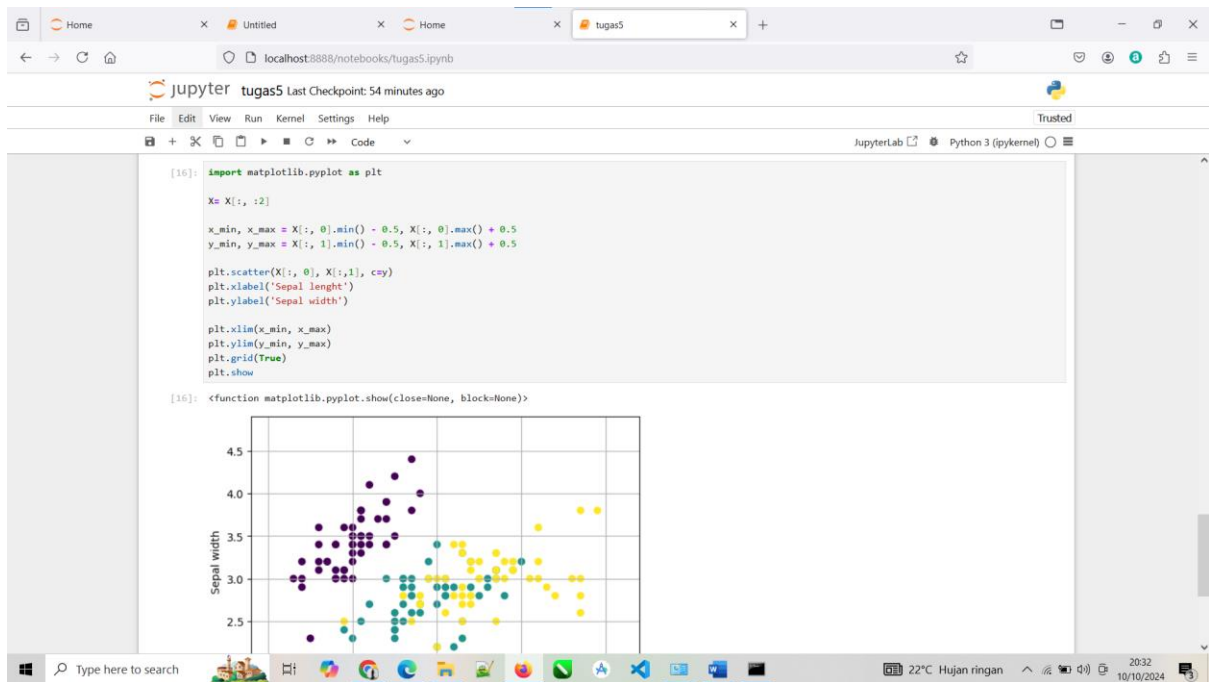
5.3. Explanatory & Response Variables | Features & Target

[illegible]

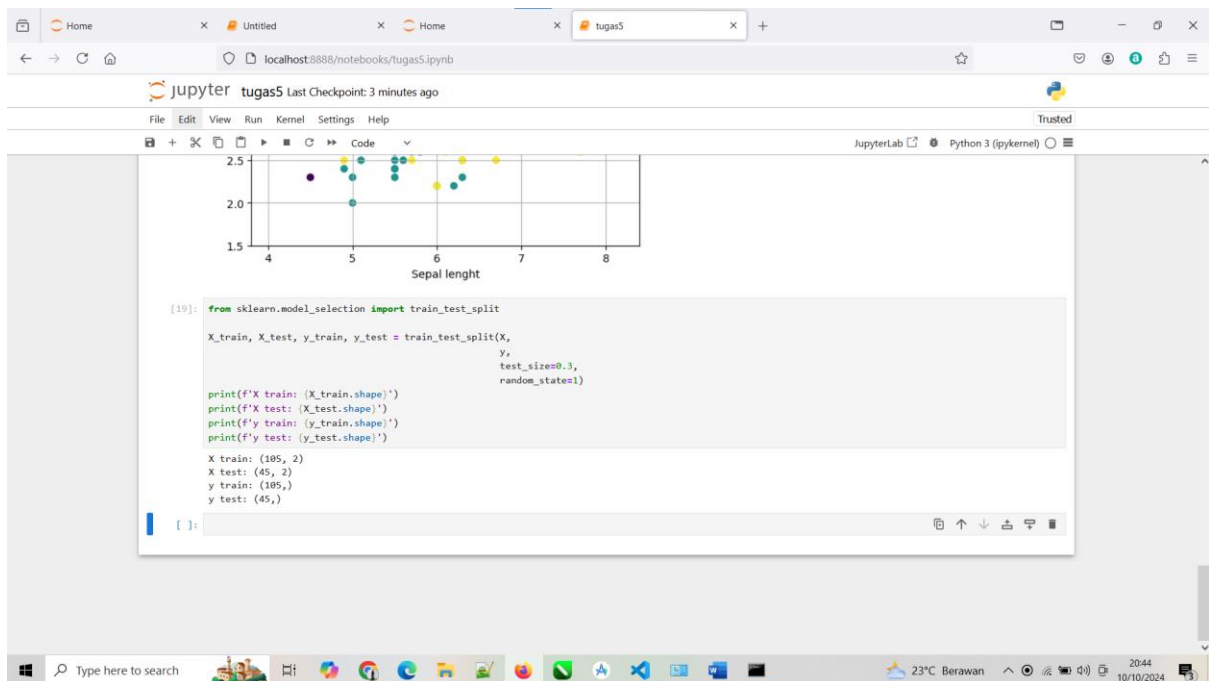
5.4. Feature & Target Names

The screenshot shows a web browser window displaying a Jupyter Notebook at localhost:8888. The notebook has three tabs: "Untitled", "Home", and "tugas5". The active tab is "tugas5", which contains a Python script. The script starts with a comment "# Importing iris dataset from sklearn" followed by two imports: "from sklearn import datasets" and "import numpy as np". Below the imports are five numbered code cells. Cell [10] defines 'y' as 'iris.target' and prints its shape. Cell [11] defines 'X' as 'iris.data' and prints its shape. Cell [12] defines 'feature_names' as 'iris.feature_names' and prints it. Cell [13] defines 'target_names' as 'iris.target_names' and prints it. Cell [14] defines 'dataset' as a dictionary containing 'X' and 'y', and prints it. The output of each cell is displayed below the code. The Jupyter interface includes a top bar with "File", "Edit", "View", "Run", "Kernel", "Settings", and "Help" menus. On the right side, there's a "Trust" button and a language selector set to "Python 3 (ipykernel)". The bottom status bar shows system icons and the date "10/10/2024".

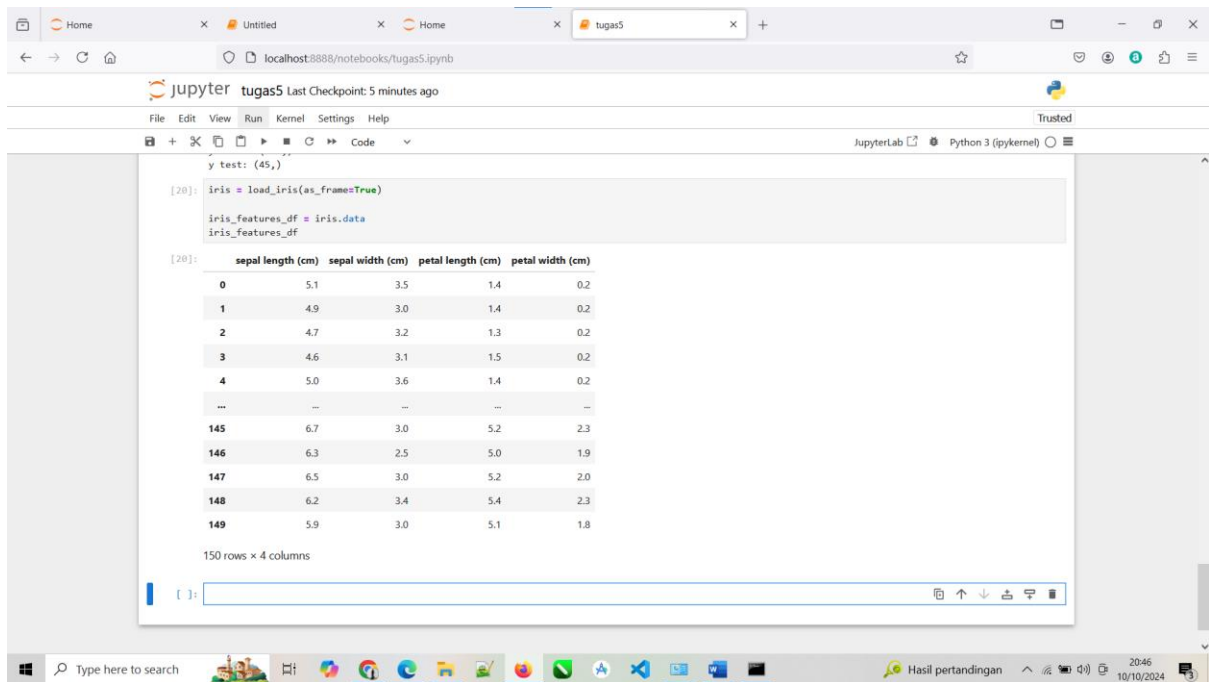
5.5. Visualisasi Data



5.6. Training Set & Testing Set

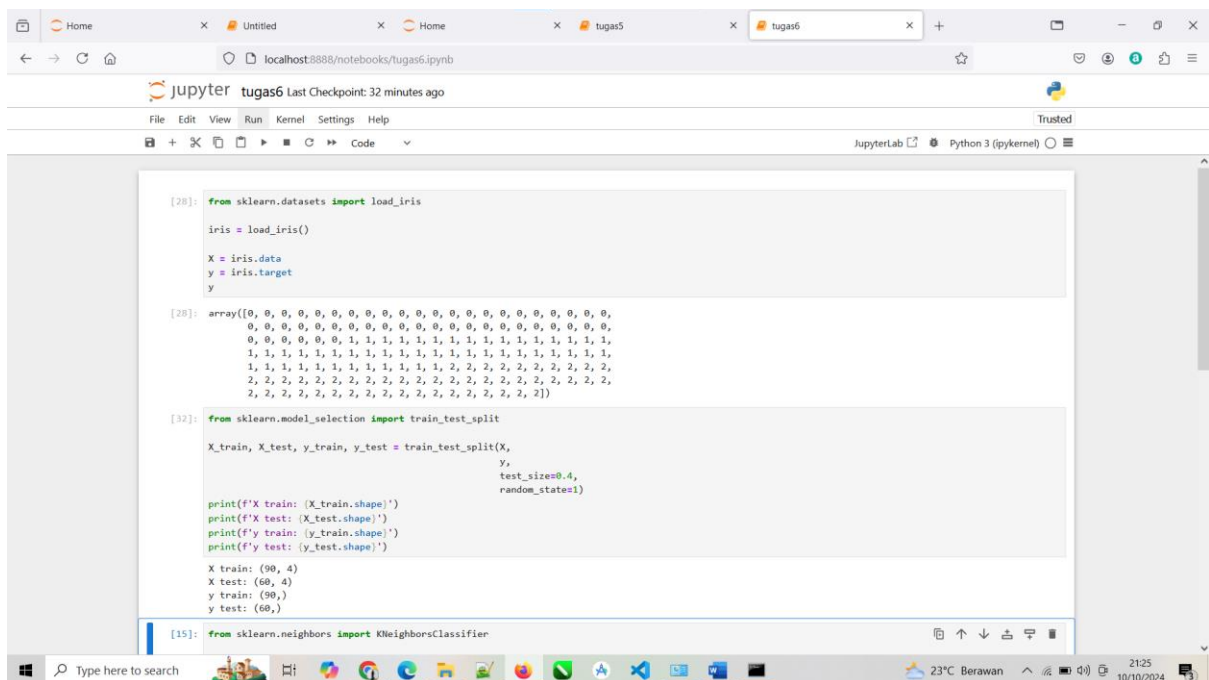


5.7. Load sample dataset sebagai Pandas Data Frame



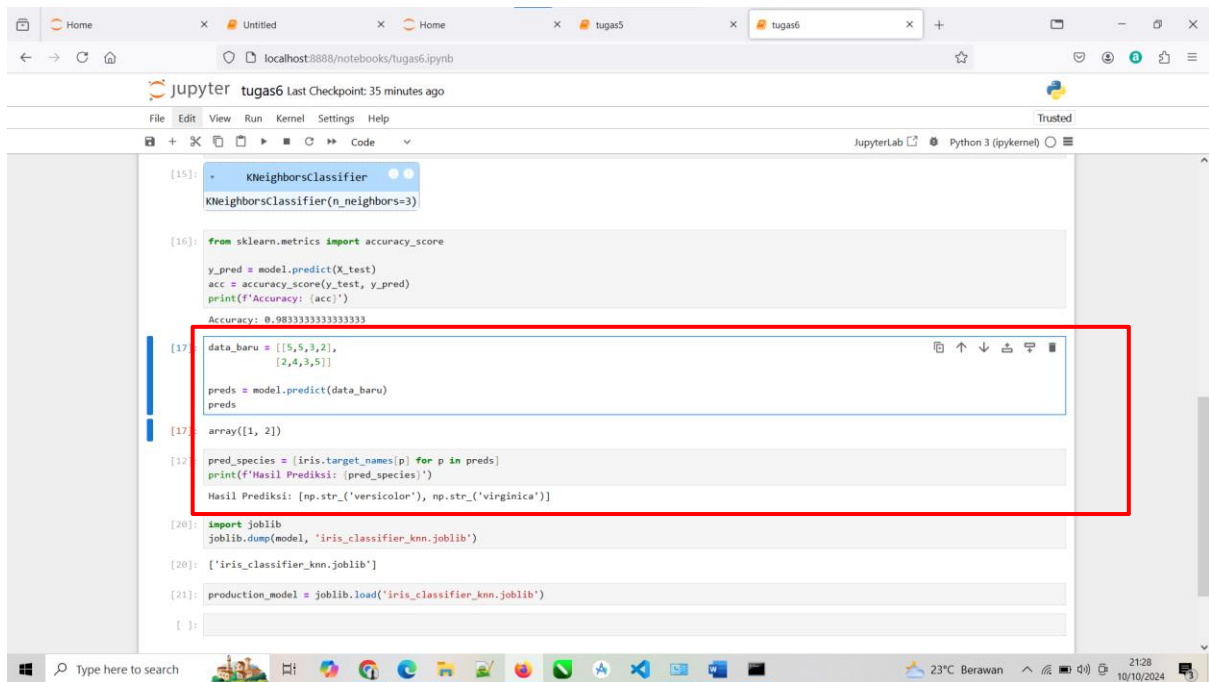
6. Berikut adalah hasil dari beberapa praktek

6.1.Persiapan dataset | Loading & splitting dataset



6.3.Evaluasi model Machine Learning

6.4. Pemanfaatan trained model machine learning



```
[15]: KNeighborsClassifier
KNeighborsClassifier(n_neighbors=3)

[16]: from sklearn.metrics import accuracy_score

y_pred = model.predict(X_test)
acc = accuracy_score(y_test, y_pred)
print(f'Accuracy: {acc}')

Accuracy: 0.9833333333333333

[17]: data_baru = [[5,5,3,2],
                 [2,4,3,5]]

preds = model.predict(data_baru)
preds

array([1, 2])

[18]: pred_species = [iris.target_names[p] for p in preds]
print(f'Hasil Prediksi: {pred_species}')

Hasil Prediksi: [np.str_('versicolor'), np.str_('virginica')]

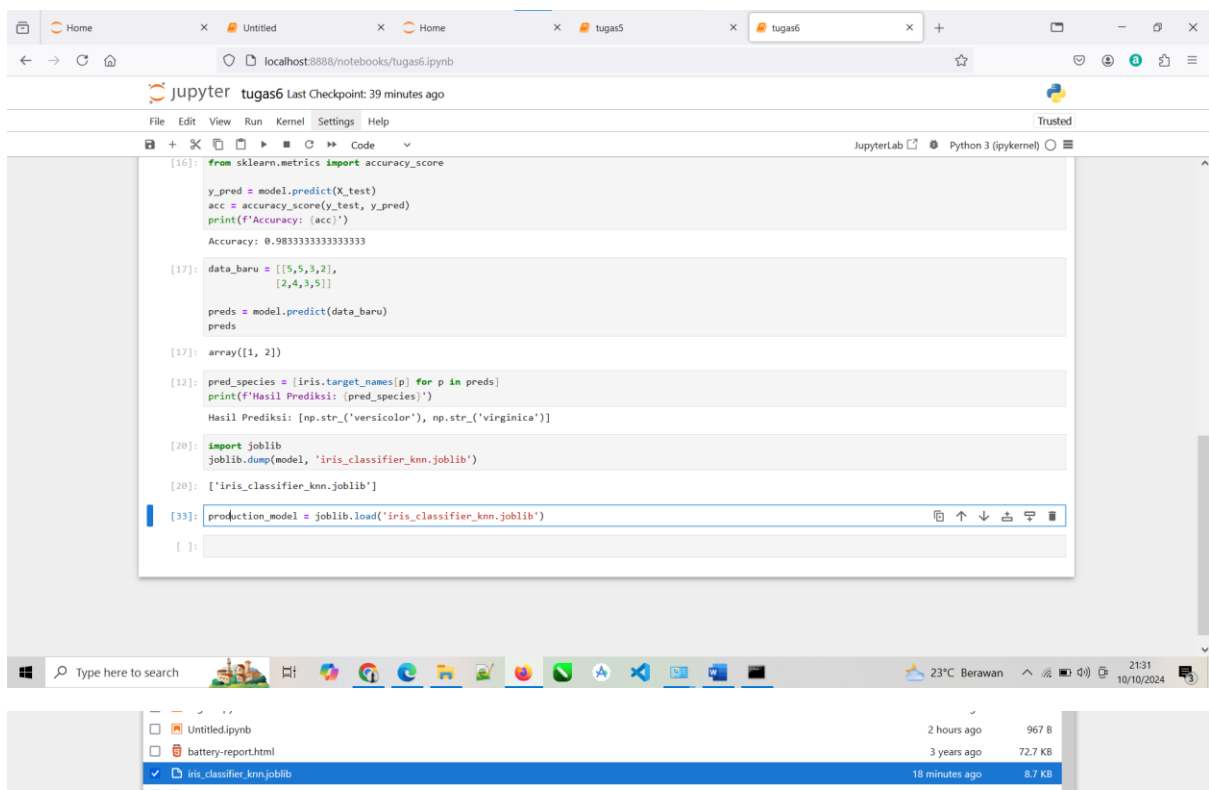
[20]: import joblib
joblib.dump(model, 'iris_classifier_knn.joblib')

[20]: ['iris_classifier_knn.joblib']

[21]: production_model = joblib.load('iris_classifier_knn.joblib')

[ ]:
```

6.5. Deploy model Machine Learning | Dumping dan Loading model Machine Learning



```
[16]: from sklearn.metrics import accuracy_score

y_pred = model.predict(X_test)
acc = accuracy_score(y_test, y_pred)
print(f'Accuracy: {acc}')

Accuracy: 0.9833333333333333

[17]: data_baru = [[5,5,3,2],
                 [2,4,3,5]]

preds = model.predict(data_baru)
preds

array([1, 2])

[18]: pred_species = [iris.target_names[p] for p in preds]
print(f'Hasil Prediksi: {pred_species}')

Hasil Prediksi: [np.str_('versicolor'), np.str_('virginica')]

[20]: import joblib
joblib.dump(model, 'iris_classifier_knn.joblib')

[20]: ['iris_classifier_knn.joblib']

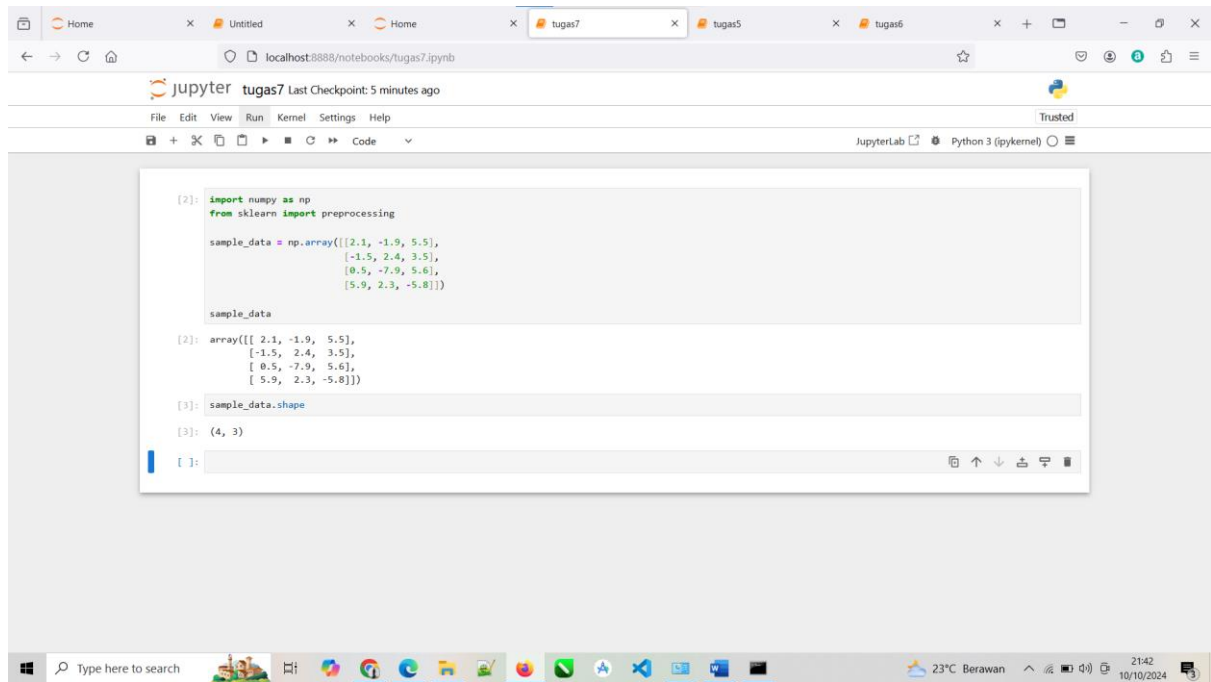
[33]: production_model = joblib.load('iris_classifier_knn.joblib')

[ ]:
```

File Name	Size	Modified
Untitled.ipynb	967 B	2 hours ago
battery-report.html	72.7 KB	3 years ago
iris_classifier_knn.joblib	8.7 KB	18 minutes ago
windows 7-3073-13-38-10-33-30.log	167.8 KB	1 day ago

7. Berikut adalah hasil dari beberapa praktek

7.1. Persiapan sample dataset



The screenshot shows a JupyterLab interface with a code editor and output area. The code defines a sample dataset and checks its shape.

```
[2]: import numpy as np
from sklearn import preprocessing

sample_data = np.array([[2.1, -1.9, 5.5],
                        [-1.5, 2.4, 3.5],
                        [0.5, -7.9, 5.6],
                        [5.9, 2.3, -5.8]])

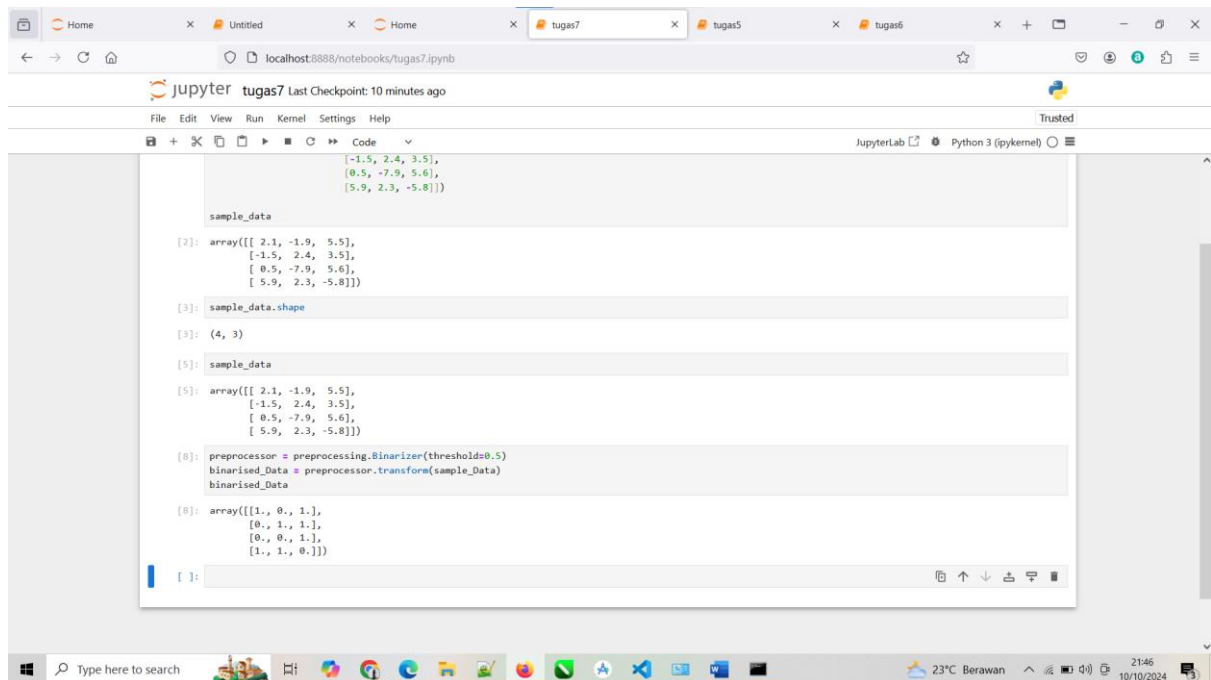
sample_data

[2]: array([[ 2.1, -1.9,  5.5],
          [-1.5,  2.4,  3.5],
          [ 0.5, -7.9,  5.6],
          [ 5.9,  2.3, -5.8]])

[3]: sample_data.shape

[3]: (4, 3)
```

7.2. Teknik data preprocessing 1: binarisation



The screenshot shows a JupyterLab interface with a code editor and output area. The code applies a Binarizer to the sample dataset and checks the result.

```
[2]: array([[ 2.1, -1.9,  5.5],
          [-1.5,  2.4,  3.5],
          [ 0.5, -7.9,  5.6],
          [ 5.9,  2.3, -5.8]])

sample_data

[2]: array([[ 2.1, -1.9,  5.5],
          [-1.5,  2.4,  3.5],
          [ 0.5, -7.9,  5.6],
          [ 5.9,  2.3, -5.8]])

[3]: sample_data.shape

[3]: (4, 3)

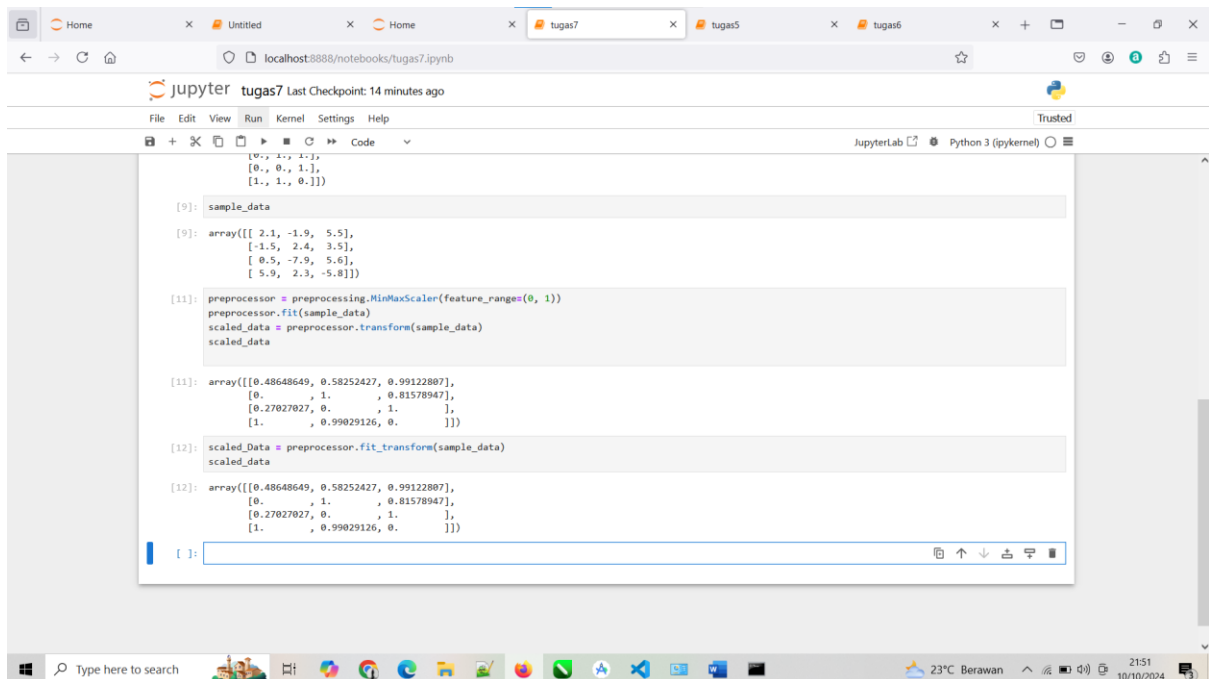
[5]: sample_data

[5]: array([[ 2.1, -1.9,  5.5],
          [-1.5,  2.4,  3.5],
          [ 0.5, -7.9,  5.6],
          [ 5.9,  2.3, -5.8]])

[8]: preprocessor = preprocessing.Binarizer(threshold=0.5)
binarised_Data = preprocessor.transform(sample_Data)
binarised_Data

[8]: array([[1., 0., 1.],
          [0., 1., 1.],
          [0., 0., 1.],
          [1., 1., 0.]])
```

7.3. Teknik data preprocessing 2: scaling



The screenshot shows a JupyterLab interface with a notebook titled 'tugas7'. The code in the notebook defines 'sample_data' as a 4x3 array of integers. It then uses 'MinMaxScaler' to scale the data. The output shows the scaled data as a 4x3 array of floats, where each column has a different range (e.g., the first column ranges from approximately 0.48 to 1.0).

```
[9]: sample_data
[9]: array([[ 2.1, -1.9,  5.5],
          [-1.5,  2.4,  3.5],
          [ 0.5, -7.9,  5.6],
          [ 5.9,  2.3, -5.8]])

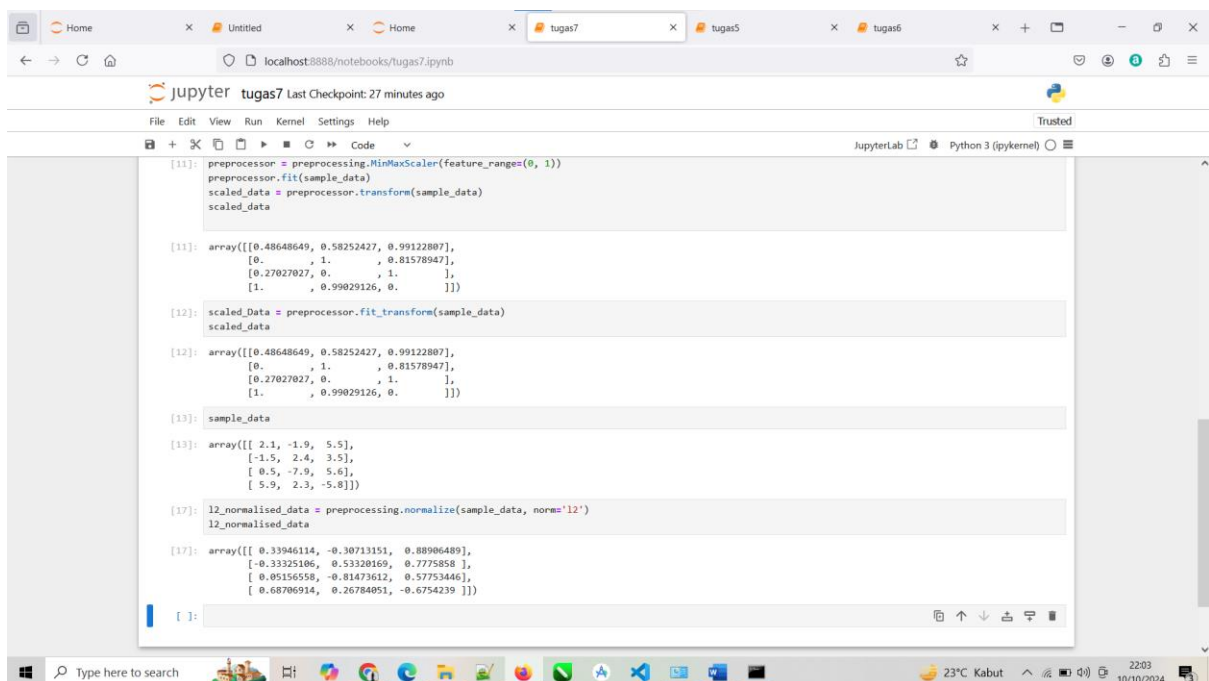
[11]: preprocessor = preprocessing.MinMaxScaler(feature_range=(0, 1))
preprocessor.fit(sample_data)
scaled_data = preprocessor.transform(sample_data)
scaled_data

[11]: array([[0.48648649, 0.58252427, 0.99122807],
          [0.         , 1.         , 0.81578947],
          [0.27027027, 0.         , 1.         ],
          [1.         , 0.99029126, 0.         ]])

[12]: scaled_Data = preprocessor.fit_transform(sample_data)
scaled_data

[12]: array([[0.48648649, 0.58252427, 0.99122807],
          [0.         , 1.         , 0.81578947],
          [0.27027027, 0.         , 1.         ],
          [1.         , 0.99029126, 0.         ]])
```

7.4. Teknik data preprocessing 3: normalisation



The screenshot shows a JupyterLab interface with a notebook titled 'tugas7'. The code defines 'sample_data' and 'scaled_data' (from the previous step). It then uses 'normalize' to normalize the scaled data. The output shows the normalized data as a 4x3 array of floats, where each column has a range from approximately -0.33 to 0.99.

```
[11]: preprocessor = preprocessing.MinMaxScaler(feature_range=(0, 1))
preprocessor.fit(sample_data)
scaled_data = preprocessor.transform(sample_data)
scaled_data

[11]: array([[0.48648649, 0.58252427, 0.99122807],
          [0.         , 1.         , 0.81578947],
          [0.27027027, 0.         , 1.         ],
          [1.         , 0.99029126, 0.         ]])

[12]: scaled_Data = preprocessor.fit_transform(sample_data)
scaled_data

[12]: array([[0.48648649, 0.58252427, 0.99122807],
          [0.         , 1.         , 0.81578947],
          [0.27027027, 0.         , 1.         ],
          [1.         , 0.99029126, 0.         ]])

[13]: sample_data
[13]: array([[ 2.1, -1.9,  5.5],
          [-1.5,  2.4,  3.5],
          [ 0.5, -7.9,  5.6],
          [ 5.9,  2.3, -5.8]])

[17]: l2_normalised_data = preprocessing.normalize(sample_data, norm='l2')
l2_normalised_data

[17]: array([[0.33946114, -0.30713151, 0.88906489],
          [-0.33325106, 0.53320169, 0.7775858 ],
          [0.05156558, -0.81473612, 0.5753446 ],
          [0.68706914, 0.26784051, -0.6754239 ]])
```