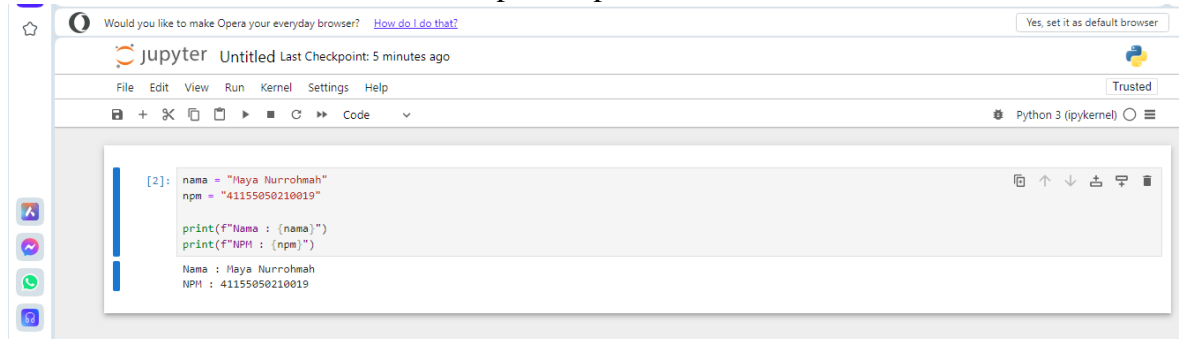


NAMA : MAYA NURROHMAH

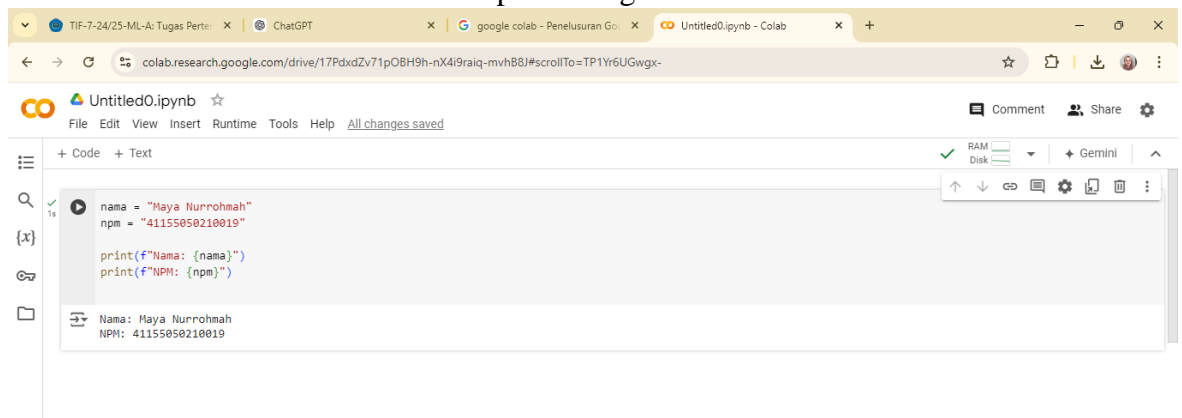
NPM : 41155050210019

KELAS : INF_A1

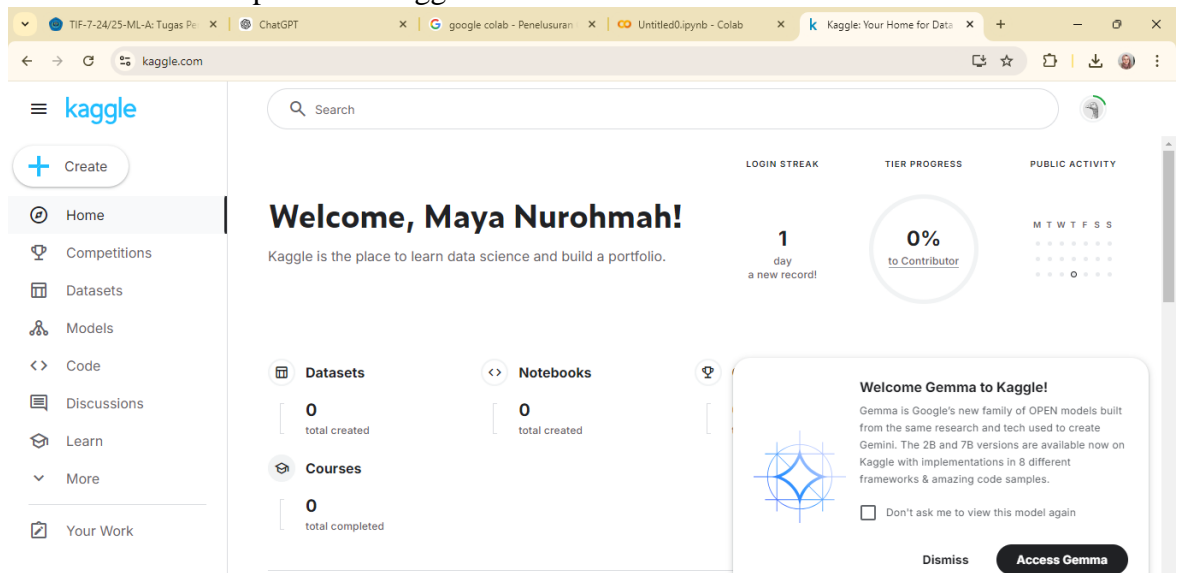
1. Tuliskan nama dan nomor NPM anda pada Jupiter Notebook.



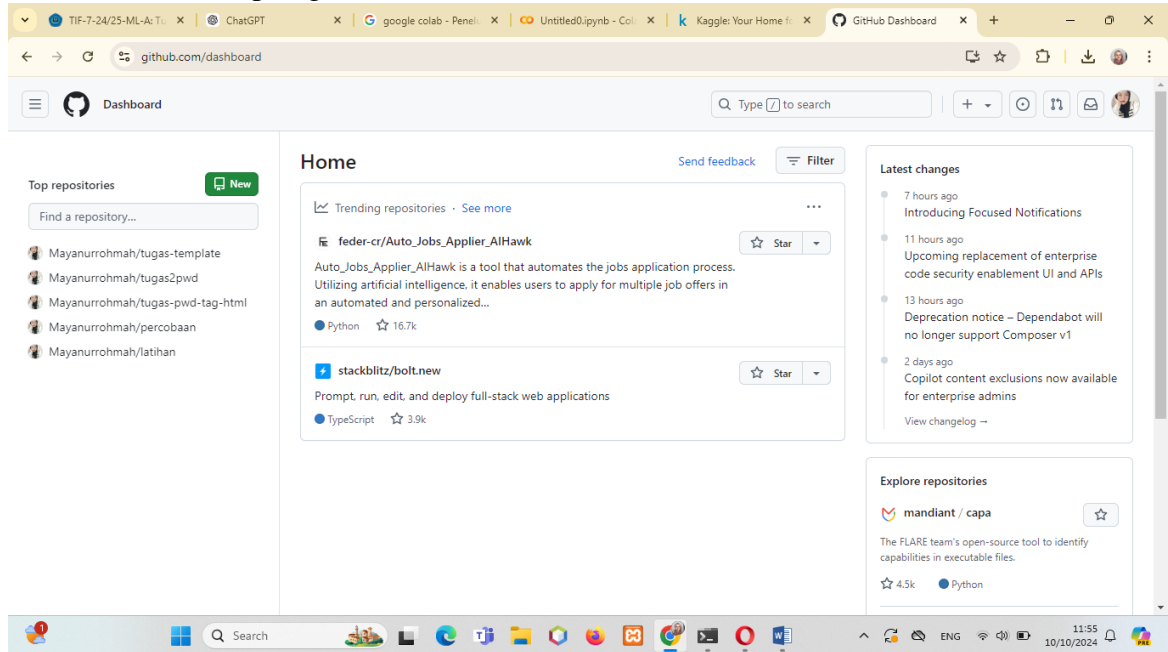
2. Tuliskan nama dan nomor NPM anda pada Google Colab.



3. Buatlah akun di <https://www.kaggle.com/>.

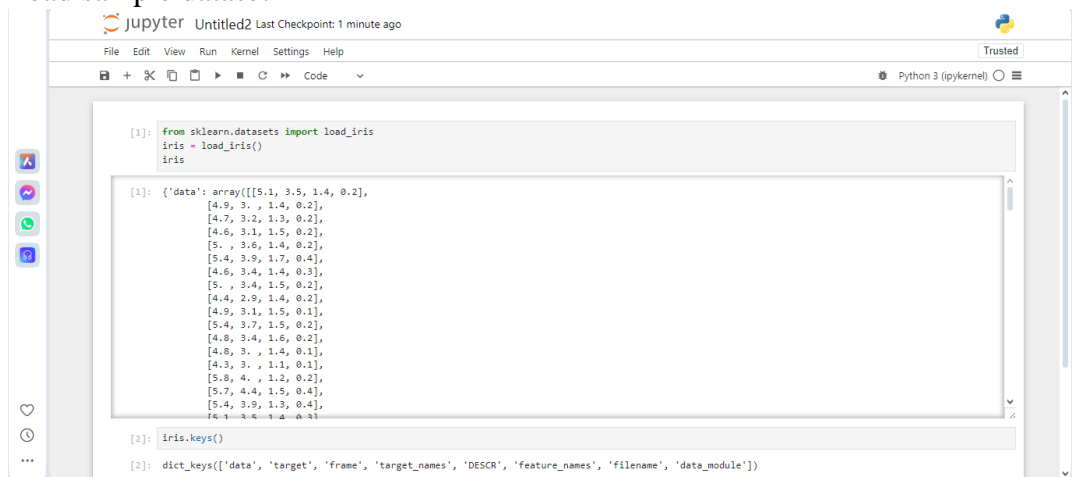


4. Buatlah akun di <https://github.com/>.

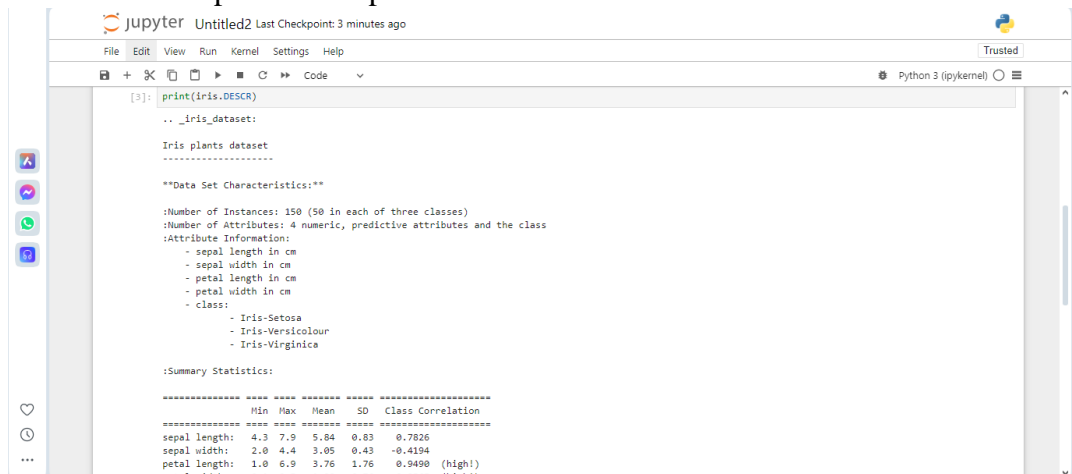


5. Lakukan praktek dari <https://youtu.be/mSO2hJln0OY?feature=shared>. Praktek tersebut yaitu:

1) Load sample dataset



2) Metadata deskripsi dari sample dataset



3) Explanatory & Response Variables | Features & Target

The screenshot shows a Jupyter Notebook interface. At the top, the title bar reads "Untitled2 Last Checkpoint: 10 minutes ago". Below the title bar is a menu bar with "File", "Edit", "View", "Run", "Kernel", "Settings", and "Help". The "Kernel" menu is open, showing options like "Restart", "Interrupt", "Shutdown", and "Restart & Clear Output". The "Restart" option is highlighted. Below the menu bar is a toolbar with icons for saving, undo, redo, and running code. The main area of the notebook contains a code cell with the following text:

```
data, test = train_test_split(X, y, test_size=0.2, random_state=42)

- See also: 1988 JMLC Proceedings, 54-64. Cheeseman et al's AUTOCLASS II
conceptual clustering system finds 3 classes in the data.
- Many, many more ...
```

Below the code cell, there are four output cells showing the results of the code execution:

```
[4]: X = iris.data
     X.shape
     # X
     (150, 4)
```

```
[6]: y = iris.target
     y.shape
     # y
     (150,)
```

At the bottom of the notebook, there is a status bar showing the current cell's index and content: "[]:".

The screenshot shows the Jupyter Notebook application window titled "Untitled2 Last Checkpoint: 11 minutes ago". The top menu bar includes File, Edit, View, Run, Kernel, Settings, and Help. On the right, there's a "Trusted" status indicator and a language selector set to "Python 3 (ipykernel)".

The left sidebar contains icons for home, search, recent notebooks, and other functions.

The main area displays two code cells:

```
[7]: X = iris.data
      # X.shape
      X
```

```
[7]: 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]])
```

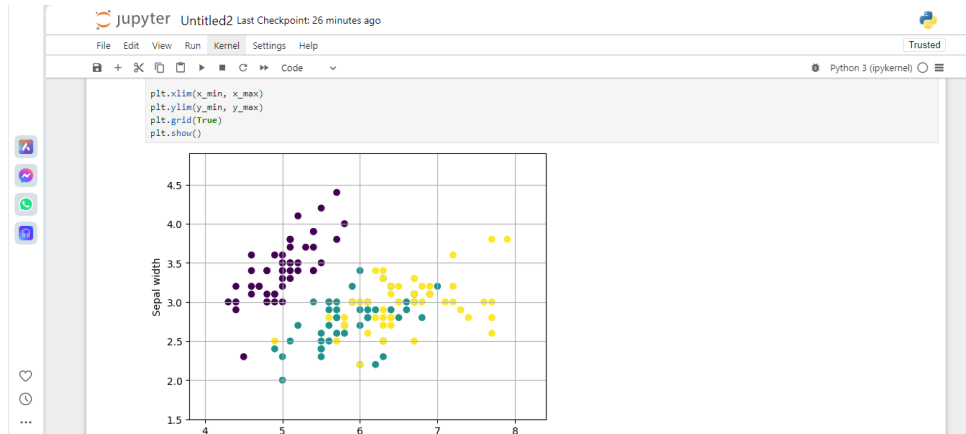
```
[8]: y = iris.target
      # y.shape
      y
```

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

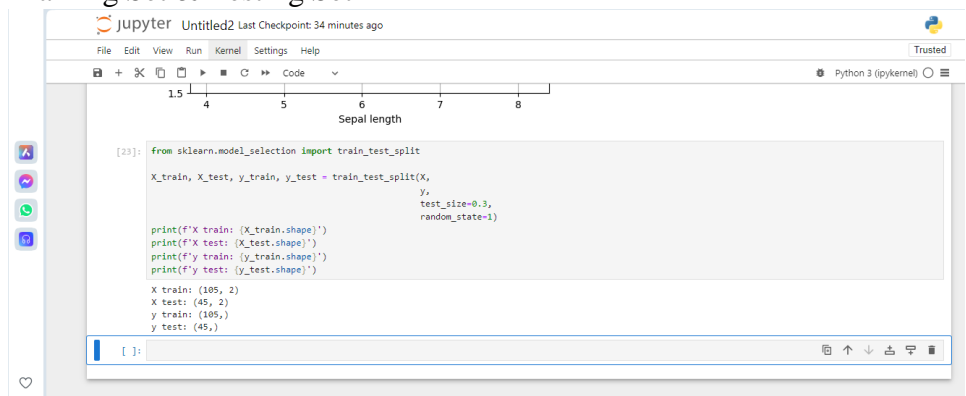
4) Feature & Target Names

[illegible]

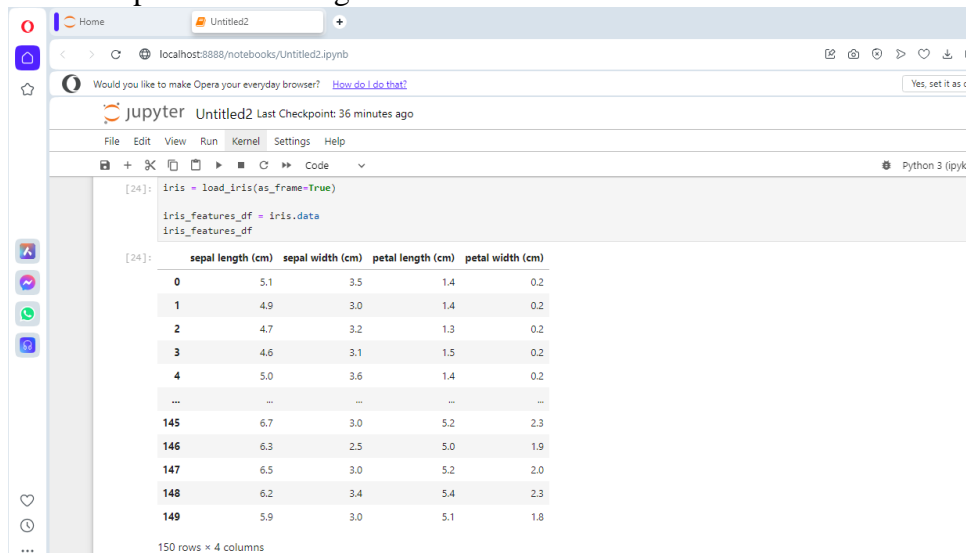
5) Visualisasi Data



6) Training Set & Testing Set

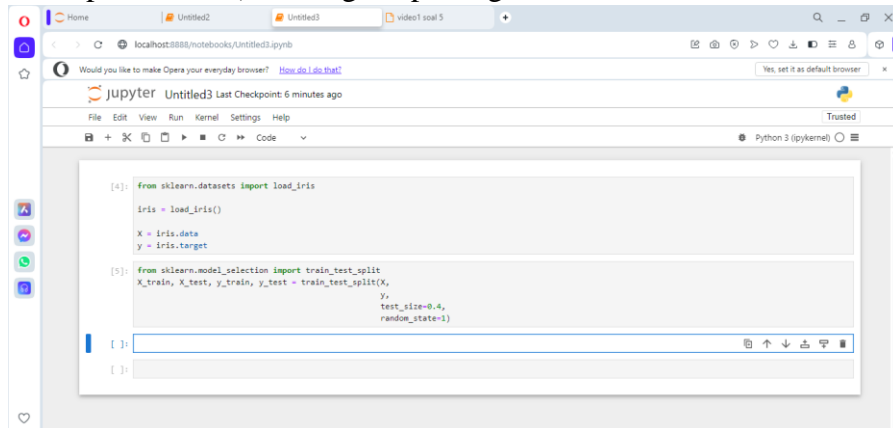


7) Load sample dataset sebagai Pandas Data Frame



6. Lakukan praktek dari <https://youtu.be/tiREcHrtDLo?feature=shared> . Praktek tersebut yaitu:

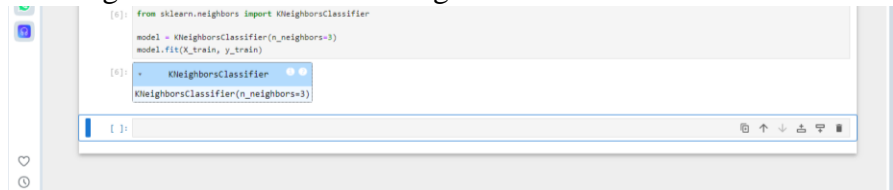
1) Persiapan dataset | Loading & splitting dataset



```
[4]: from sklearn.datasets import load_iris
iris = load_iris()
X = iris.data
y = iris.target

[5]: from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X,
                                                    y,
                                                    test_size=0.4,
                                                    random_state=1)
```

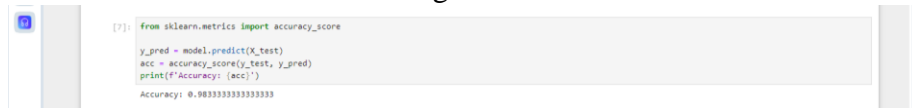
2) Training model Machine Learning



```
[6]: from sklearn.neighbors import KNeighborsClassifier
model = KNeighborsClassifier(n_neighbors=3)
model.fit(X_train, y_train)

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

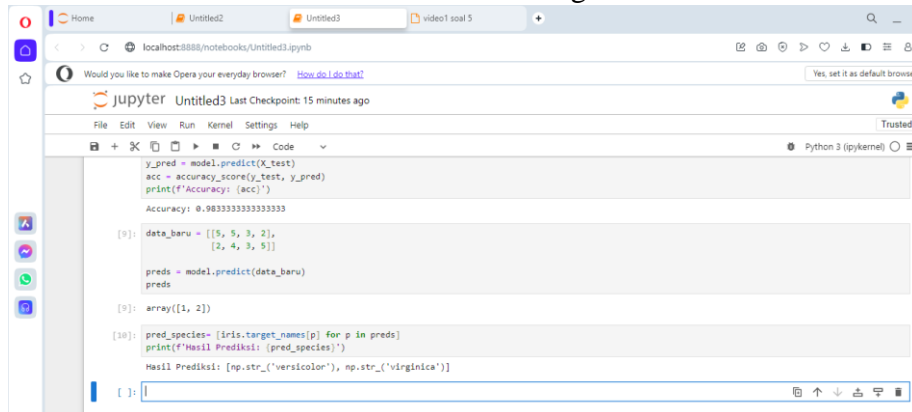
3) Evaluasi model Machine Learning



```
[7]: 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
```

4) Pemanfaatan trained model machine learning



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

Accuracy: 0.9833333333333333

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

preds = model.predict(data_baru)
preds

[10]: array([1, 2])

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

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

5) Deploy model Machine Learning | Dumping dan Loading model Machine Learning



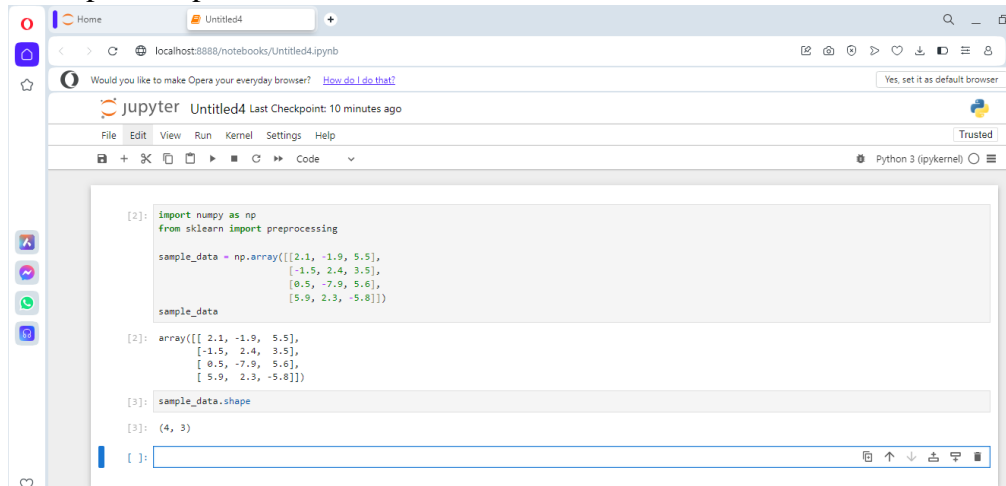
```
[11]: import joblib
joblib.dump(model, 'iris_classifier_knn.joblib')

[12]: ['iris_classifier_knn.joblib']

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

7. Lakukan praktek dari <https://youtu.be/smNnhEd26Ek?feature=shared> . Praktek tersebut yaitu:

1) Persiapan sample dataset



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

```
[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]])

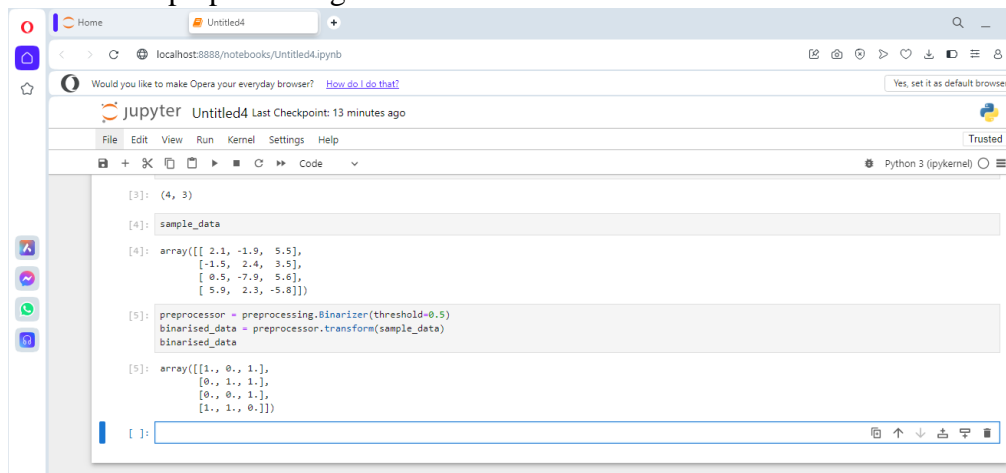
sample_data
```

```
[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)
```

2) Teknik data preprocessing 1: binarization



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

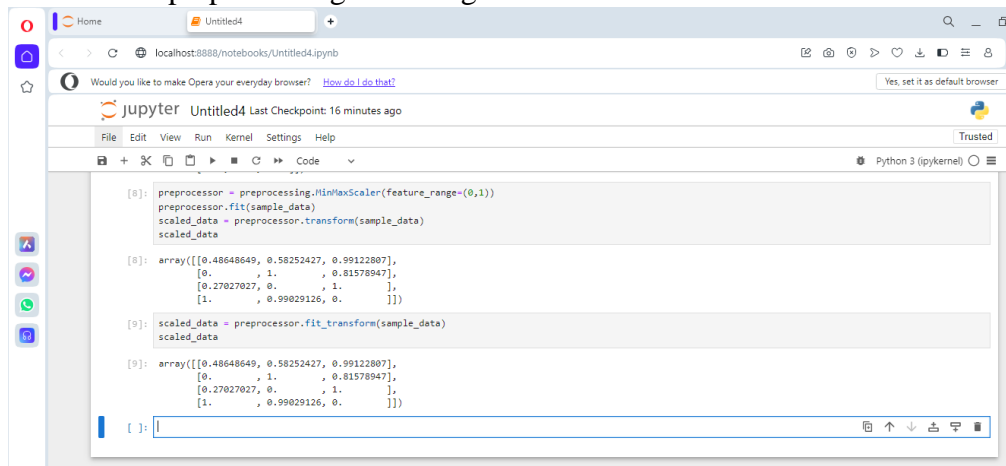
```
[3]: (4, 3)
sample_data
```

```
[4]: 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]])
```

```
[5]: preprocessor = preprocessing.Binarizer(threshold=0.5)
binarised_data = preprocessor.transform(sample_data)
binarised_data
```

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

3) Teknik data preprocessing 2: scaling



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

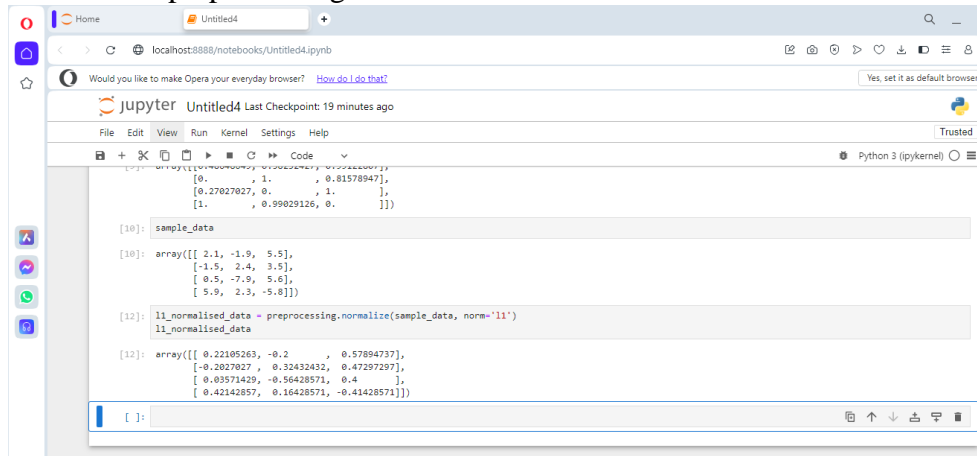
```
[8]: preprocessor = preprocessing.MinMaxScaler(feature_range=(0,1))
preprocessor.fit(sample_data)
scaled_data = preprocessor.transform(sample_data)
scaled_data
```

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

```
[9]: scaled_data = preprocessor.fit_transform(sample_data)
scaled_data
```

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

4) Teknik data preprocessing 3: normalization



This screenshot shows a Jupyter Notebook interface with the following code and output:

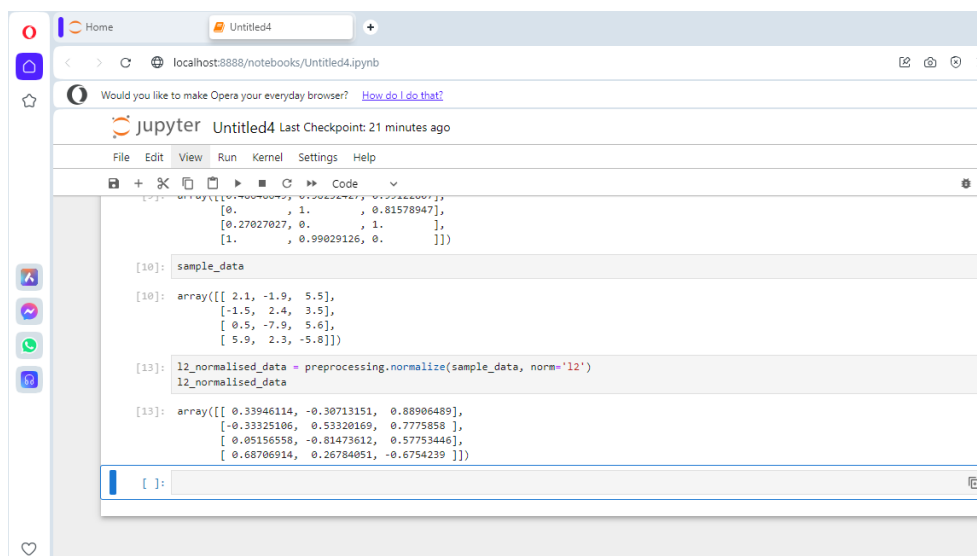
```
[7]: array([[1.0000000e+00, 0.0000000e+00, 0.0000000e+00],
        [0. , 1. , 0.81578947],
        [0.27027027, 0. , 1. ],
        [1. , 0.99029126, 0. ]])
```

```
[10]: sample_data
```

```
[10]: 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]])
```

```
[12]: l1_normalised_data = preprocessing.normalize(sample_data, norm='l1')
l1_normalised_data
```

```
[12]: array([[ 0.22105263, -0.2 ,  0.57894737],
        [-0.2027027 ,  0.32432432,  0.47297297],
        [ 0.03571429, -0.56428571,  0.4 ],
        [ 0.42142857,  0.16428571, -0.41428571]])
```



This screenshot shows a Jupyter Notebook interface with the following code and output:

```
[7]: array([[1.0000000e+00, 0.0000000e+00, 0.0000000e+00],
        [0. , 1. , 0.81578947],
        [0.27027027, 0. , 1. ],
        [1. , 0.99029126, 0. ]])
```

```
[10]: sample_data
```

```
[10]: 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]])
```

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

```
[13]: array([[ 0.33946114, -0.30713151,  0.88906489],
        [-0.33325106,  0.53320169,  0.7775858 ],
        [ 0.05156558, -0.81473612,  0.57753446],
        [ 0.68706914,  0.26784051, -0.6754239 ]])
```