NAMA : ILPAN

NPM : 41155050210046

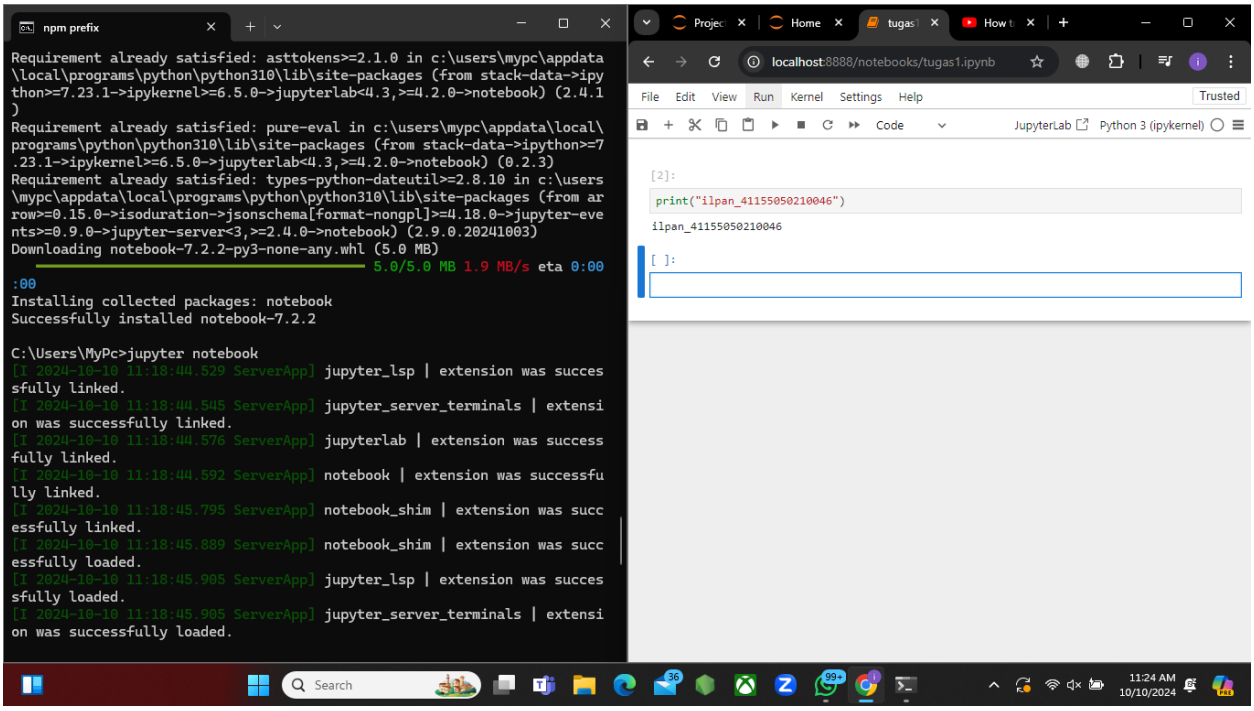
KELAS : A-2

MATA KULIAH : MACHINE LEARNING

Tugas

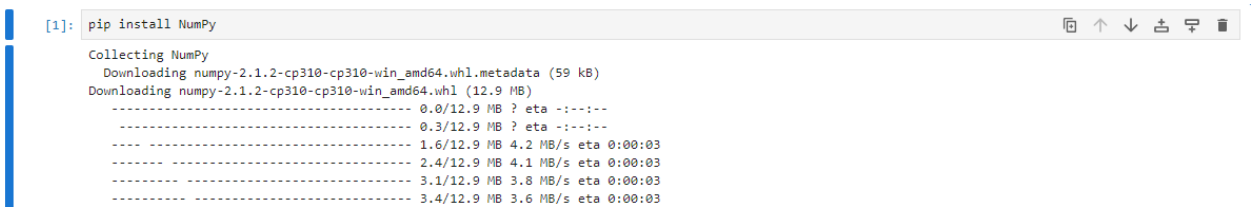
1.

Instalasi Jupyter



Liblary

NumPy



SciPy

```
[2]: pip install SciPy
```

```
Collecting SciPy
  Downloading scipy-1.14.1-cp310-cp310-win_amd64.whl.metadata (60 kB)
Requirement already satisfied: numpy<2.3,>=1.23.5 in c:\users\mypc\appdata\local\programs\python\python310\lib\site-packages (from SciPy) (2.1.2)
Downloading scipy-1.14.1-cp310-cp310-win_amd64.whl (44.8 MB)
   ---------------------------------------- 0.0/44.8 MB ? eta -:--:--
   ---------------------------------------- 0.5/44.8 MB 4.2 MB/s eta 0:00:11
    --------------------------------------- 1.0/44.8 MB 3.0 MB/s eta 0:00:15
   - -------------------------------------- 1.3/44.8 MB 2.3 MB/s eta 0:00:19
```

Pandas

```
[3]: pip install Pandas
```

```
Collecting PandasNote: you may need to restart the kernel to use updated packages.

  Downloading pandas-2.2.3-cp310-cp310-win_amd64.whl.metadata (19 kB)
Requirement already satisfied: numpy>=1.22.4 in c:\users\mypc\appdata\local\programs\python\python310\lib\site-packages (from Pandas) (2.1.2)
Requirement already satisfied: python-dateutil>=2.8.2 in c:\users\mypc\appdata\local\programs\python\python310\lib\site-packages (from Pandas) (2.9.0.pos
t0)
Collecting pytz>=2020.1 (from Pandas)
```

Matplotlib

```
[4]: pip install Matplotlib
```

```
Collecting MatplotlibNote: you may need to restart the kernel to use updated packages.

  Downloading matplotlib-3.9.2-cp310-cp310-win_amd64.whl.metadata (11 kB)
Collecting contourpy>=1.0.1 (from Matplotlib)
  Downloading contourpy-1.3.0-cp310-cp310-win_amd64.whl.metadata (5.4 kB)
Collecting cycler>=0.10 (from Matplotlib)
  Downloading cycler-0.12.1-py3-none-any.whl.metadata (3.8 kB)
```

Seaborn

```
[6]: pip install Seaborn
```

```
Collecting Seaborn
  Downloading seaborn-0.13.2-py3-none-any.whl.metadata (5.4 kB)
Requirement already satisfied: numpy!=1.24.0,>=1.20 in c:\users\mypc\appdata\local\programs\python\python310\lib\site-packages (from Seaborn) (2.1.2)
Requirement already satisfied: pandas>=1.2 in c:\users\mypc\appdata\local\programs\python\python310\lib\site-packages (from Seaborn) (2.2.3)
Requirement already satisfied: matplotlib!=3.6.1,>=3.4 in c:\users\mypc\appdata\local\programs\python\python310\lib\site-packages (from Seaborn) (3.9.2)
Requirement already satisfied: contourpy>=1.0.1 in c:\users\mypc\appdata\local\programs\python\python310\lib\site-packages (from matplotlib!=3.6.1,>=3.4-
>Seaborn) (1.3.0)
Requirement already satisfied: cycler>=0.10 in c:\users\mypc\appdata\local\programs\python\python310\lib\site-packages (from matplotlib!=3.6.1,>=3.4->Sea
born) (0.12.1)
```

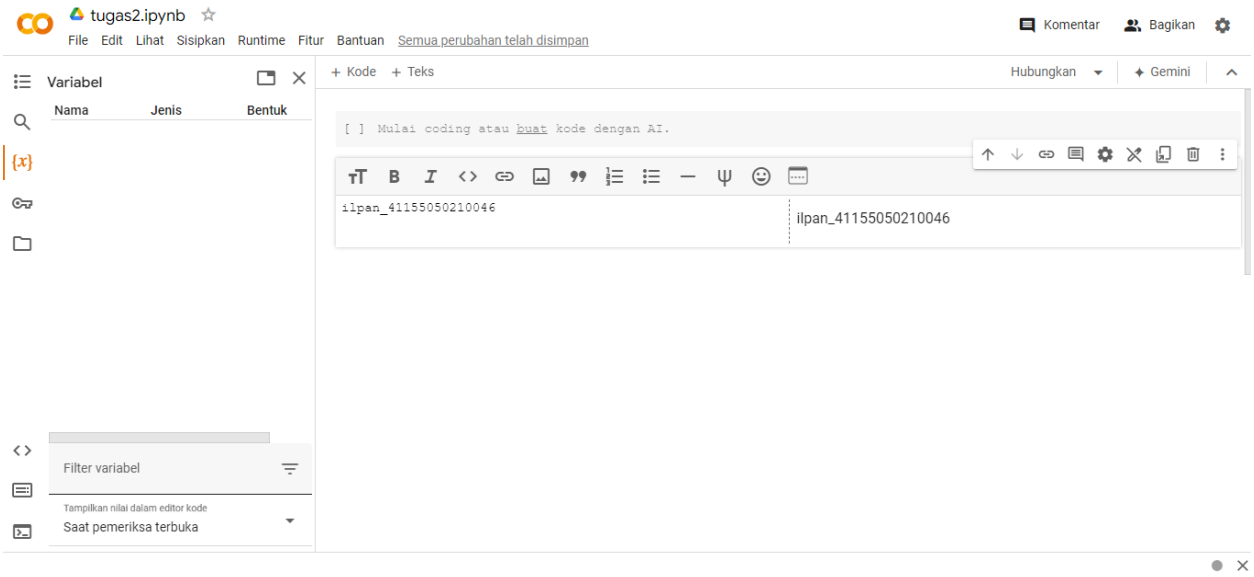
Scikit -learn

```
[7]: pip install Scikit-learn
```

```
Collecting Scikit-learn
  Downloading scikit_learn-1.5.2-cp310-cp310-win_amd64.whl.metadata (13 kB)
Requirement already satisfied: numpy>=1.19.5 in c:\users\mypc\appdata\local\programs\python\python310\lib\site-packages (from Scikit-learn) (2.1.2)
Requirement already satisfied: scipy>=1.6.0 in c:\users\mypc\appdata\local\programs\python\python310\lib\site-packages (from Scikit-learn) (1.14.1)
Collecting joblib>=1.2.0 (from Scikit-learn)
  Downloading joblib-1.4.2-py3-none-any.whl.metadata (5.4 kB)
Collecting threadpoolctl>=3.1.0 (from Scikit-learn)
  Downloading threadpoolctl-3.5.0-py3-none-any.whl.metadata (13 kB)
```

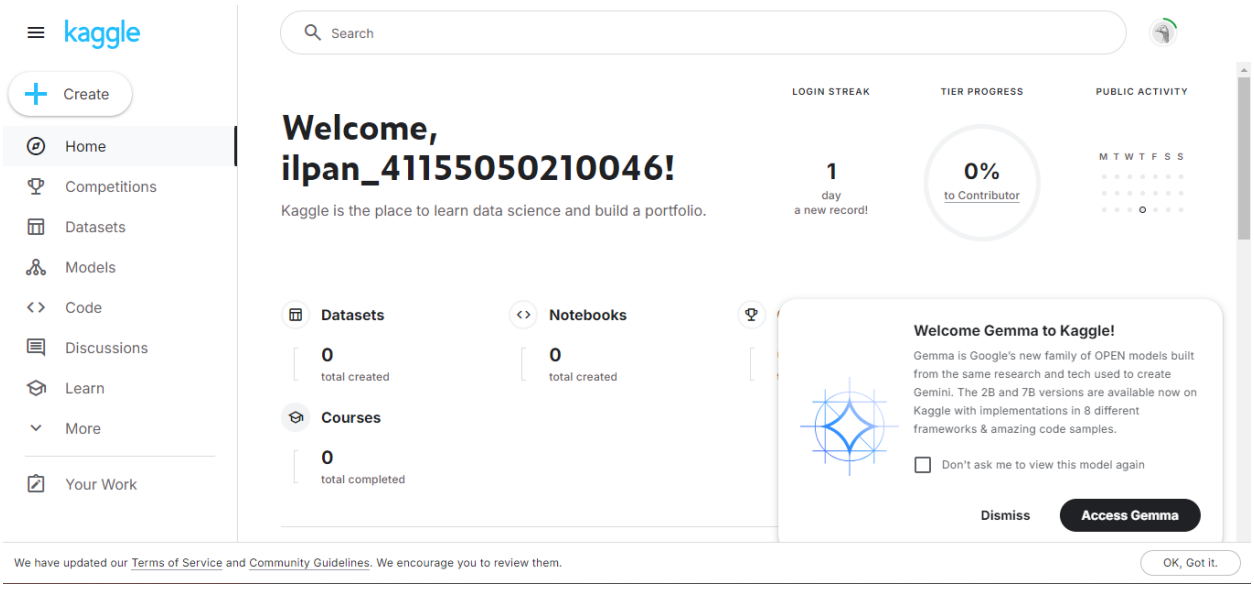
2.

Register Google Collab



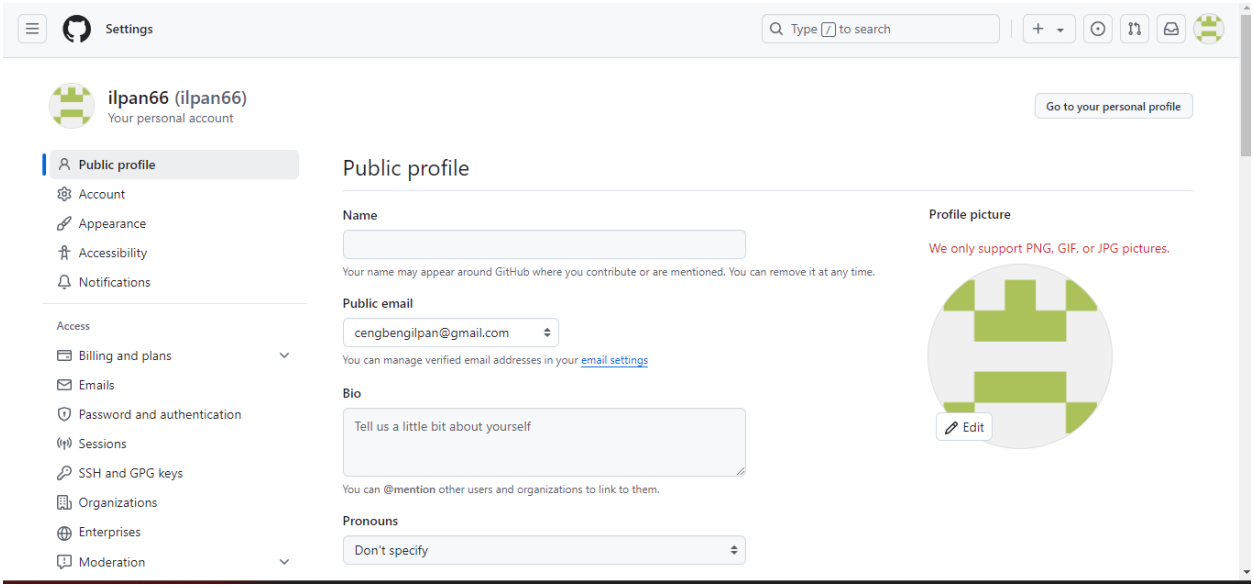
3.

Register Kaggle



4.

Register GitHub



5.

Load sample Dataset

```
[1]: from sklearn.datasets import load_iris

     iris = load_iris()
     iris
```

```
[1]: {'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],
```

```
[2]: iris.keys()
```

```
[2]: dict_keys(['data', 'target', 'frame', 'target_names', 'DESCR', 'feature_names', 'filename', 'data_module'])
```

```
[ ]:
```

Metadata | Deskripsi dari sample dataset

```
[5]: print(iris.DESCR)
```

```
.. _iris_dataset:

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!)
============== ==== ==== ======= ===== ====================
```

Explanatory & Response Variables | Features & Target

Explanatory Variables (Features)

```
[6]: x = iris.data
     x.shape
     # x
```

```
[6]: (150, 4)
```

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

```
[8]: 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, 0.3],
```

```
[ ]:
```

Response Variables (Target)

```
[11]: Y = iris.target
      Y.shape
      #Y
```

```
[11]: (150,)
```

```
[ ]:
```

```
[12]: Y = iris.target
      #Y.shape
      Y
```

```
[12]: array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
             0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
             0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
             1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
             1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 2, 2, 2, 2, 2, 2, 2, 2,
             2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
             2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2])
```

```
[ ]:
```

Feature & Target Names

```
[14]: feature_names = iris.feature_names
      feature_names
```

```
[14]: ['sepal length (cm)',
       'sepal width (cm)',
       'petal length (cm)',
       'petal width (cm)']
```

```
[16]: target_names = iris.target_names
      target_names
```

```
[16]: array(['setosa', 'versicolor', 'virginica'], dtype='<U10')
```

```
[ ]:
```

Visualisasi Data

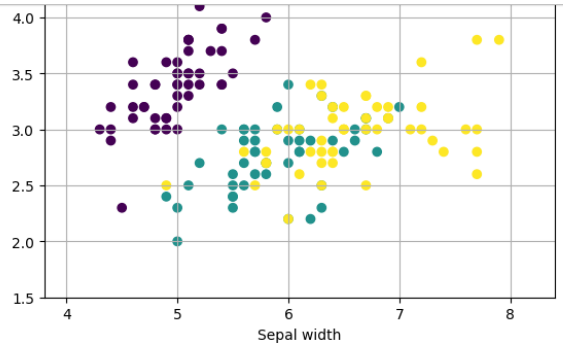
```
[7]: array(['setosa', 'versicolor', 'virginica'], dtype='<U10')
```

```
[20]: import matplotlib.pyplot as plt

X = X[:, :2]

x_min, x_max = X[:, 0].min() - 0.5, X[:, 0].max() + 0.5
y_min, y_max = X[:, 1].min() - 0.5, X[:, 1].max() + 0.5

plt.scatter(X[:, 0], X[:, 1], c=Y)
plt.xlabel('Sepal length')
plt.xlabel('Sepal width')

plt.xlim(x_min, x_max)
plt.ylim(y_min, y_max)
plt.grid(True)
plt.show()
```

jupyter Untitled3 Last Checkpoint: 31 minutes ago

File Edit View Run Kernel Settings Help Trusted

Code JupyterLab Python 3 (ipykernel)




```
[ ]:
```

Training Set & Testing Set

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

X_train, X_test, y_train, y_test = train_test_split(X,
                                                    Y,
                                                    test_size=0.3,
                                                    random_state=1)
print(f'X train: {X_train.shape}')
print(f'X test: {X_test.shape}')
print(f'y train: {y_train.shape}')
print(f'y test: {y_test.shape}')
```

```
X train: (105, 2)
X test: (45, 2)
y train: (105,)
y test: (45,)
```

```
[ ]:
```

Load sample dataset sebagai Pandas Data Frame

```
[37]: iris = load_iris(as_frame=True)

      iris_features_df = iris.data
      iris_features_df
```

[37]:

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)
0	5.1	3.5	1.4	0.2
1	4.9	3.0	1.4	0.2
2	4.7	3.2	1.3	0.2
3	4.6	3.1	1.5	0.2
4	5.0	3.6	1.4	0.2
...	...	...	...	...
145	6.7	3.0	5.2	2.3
146	6.3	2.5	5.0	1.9
147	6.5	3.0	5.2	2.0
148	6.2	3.4	5.4	2.3
149	5.9	3.0	5.1	1.8

150 rows × 4 columns

6.

Persiapan dataset | Loading & splitting dataset

Load sample dataset : Iris Dataset

```
[5]: from sklearn.datasets import load_iris

     iris = load_iris()

     x = iris.data
     y = iris.target
     x
```

```
[5]: 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],
```

```
[6]: from sklearn.datasets import load_iris

     iris = load_iris()

     x = iris.data
     y = iris.target
     y
```

```
[6]: array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
            0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
            0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
            1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
            1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 2, 2, 2, 2, 2, 2, 2, 2,
            2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
            2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2])
```

Splitting Dataset : Training & Testing Set

```
[13]: 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)
      print(f'X train: {X_train.shape}')
      print(f'X test: {X_test.shape}')
      print(f'y train: {y_train.shape}')
      print(f'y test: {y_test.shape}')

      X train: (90, 4)
      X test: (60, 4)
      y train: (90,)
      y test: (60,)

[ ]:
```

Training model Machine Learning

```
[15]: from sklearn.neighbors import KNeighborsClassifier

      model = KNeighborsClassifier(n_neighbors=3)
      model.fit(X_train, y_train)

[15]: KNeighborsClassifier
      KNeighborsClassifier(n_neighbors=3)

[ ]:
```

Evaluasi 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

[ ]:
```

Pemanfaatan trained model machine learning

```
[17]: data_baru = [[5, 5, 3, 2],
                  [2, 4, 3, 5]]
      preds = model.predict(data_baru)
      preds

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

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

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

[ ]:
```

Deploy model Machine Learning | Dumping dan Loading model Machine Learning

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

[22]: ['iris_classifier_knn.joblib']

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

[ ]:
```

7.

Persiapan sample dataset

```
[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)

[ ]:
```

Teknik data preprocessing 1: binarization

```
[4]: 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.]])

[ ]:
```

Teknik data preprocessing 2: scaling

```
[6]: sample_data

[6]: 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]])

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

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

[8]: scaled_data = preprocessor.fit_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.        ]])

[ ]:
```

Teknik data preprocessing 3: normalization

Least Absolute Deviations

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

[11]: 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]])

[ ]:
```

Least Squeres

```
[12]: sample_data

[12]: 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 ]])

[ ]:
```