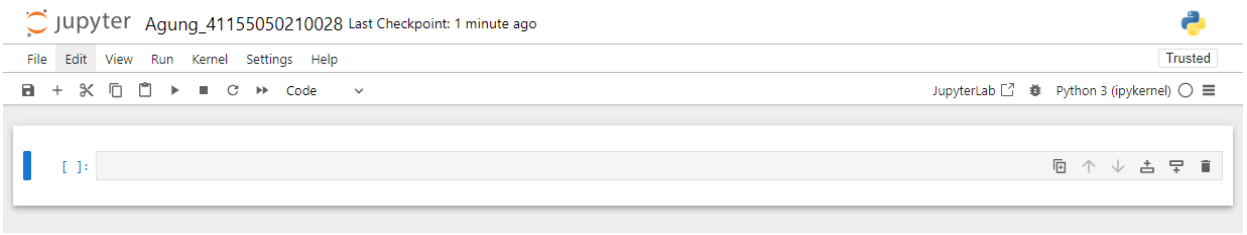
Nama : Agung Purnama Sandi

NIM : 41155050210028

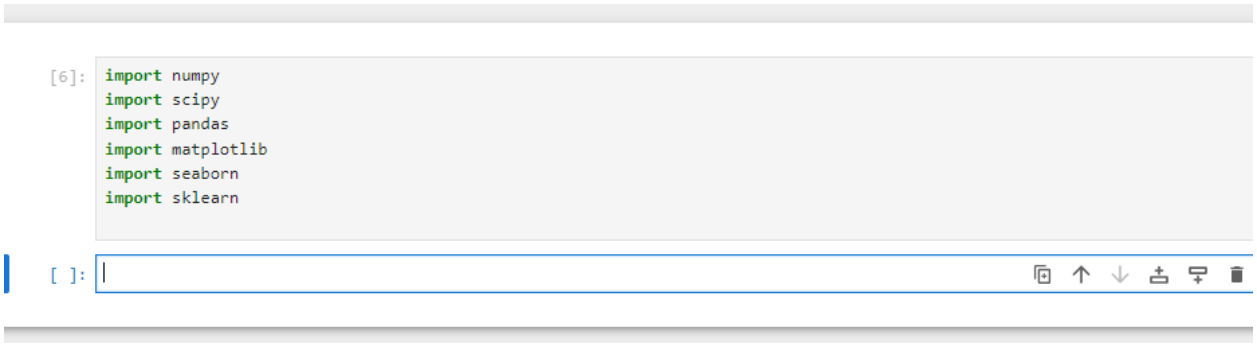
Mata Kuliah : Machine Learning

1. Instalasi Jupyter Noterbook
   1.1. Jupyter Notebook

```
C:\Users\Ren>jupyter --version
Selected Jupyter core packages...
IPython          : 8.28.0
ipykernel        : 6.29.5
ipywidgets       : not installed
jupyter_client   : 8.6.3
jupyter_core     : 5.7.2
jupyter_server   : 2.14.2
jupyterlab       : 4.2.5
nbclient         : 0.10.0
nbconvert        : 7.16.4
nbformat         : 5.10.4
notebook         : 7.2.2
qtconsole        : not installed
traitlets        : 5.14.3

C:\Users\Ren>
```



   1.2. Import Library

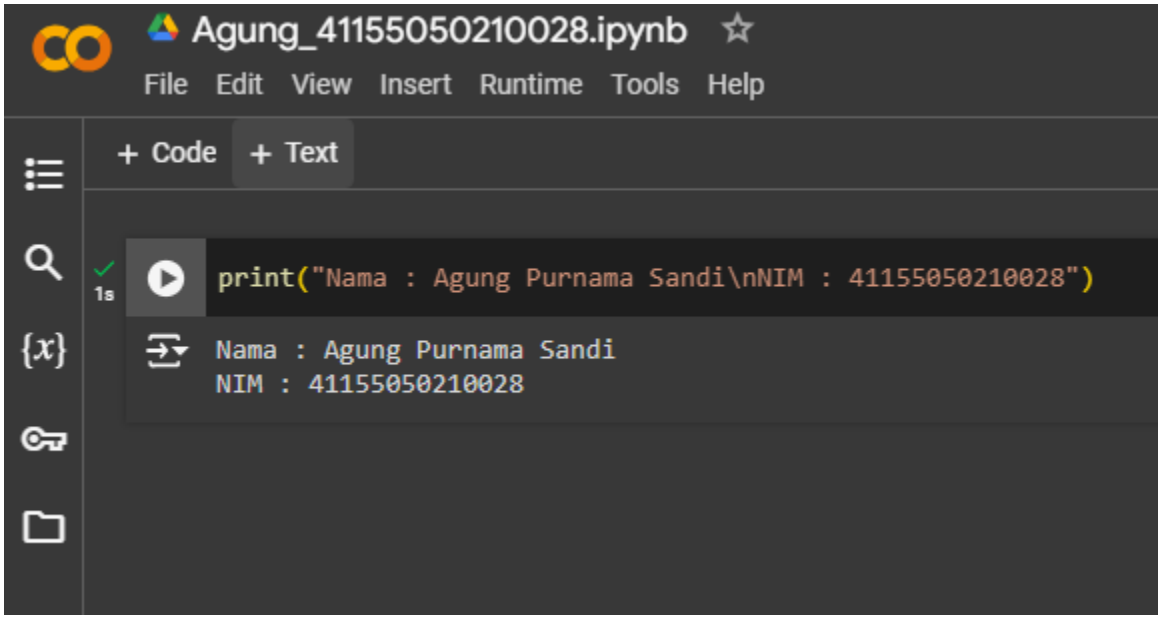


1.3. Nama & NPM

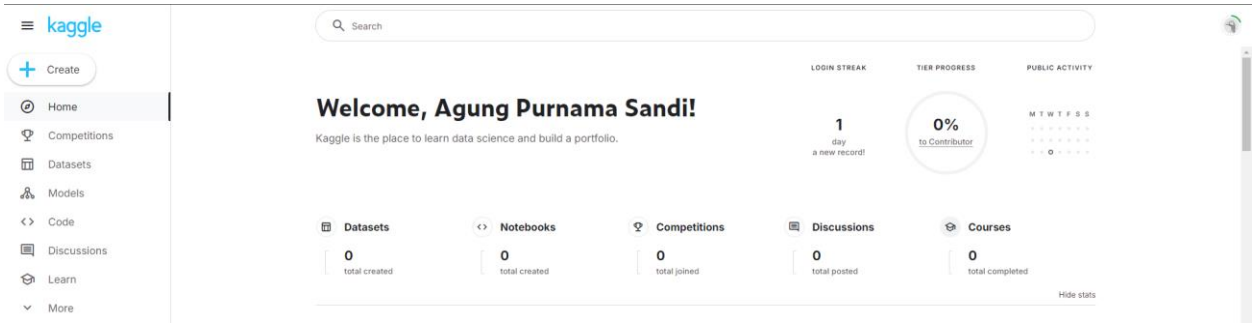
```
import numpy
import scipy
import pandas
import matplotlib
import seaborn
import sklearn

print("Nama : Agung Purnama Sandi\nNIM : 41155050210028")
```

```
Nama : Agung Purnama Sandi
NIM : 41155050210028
```

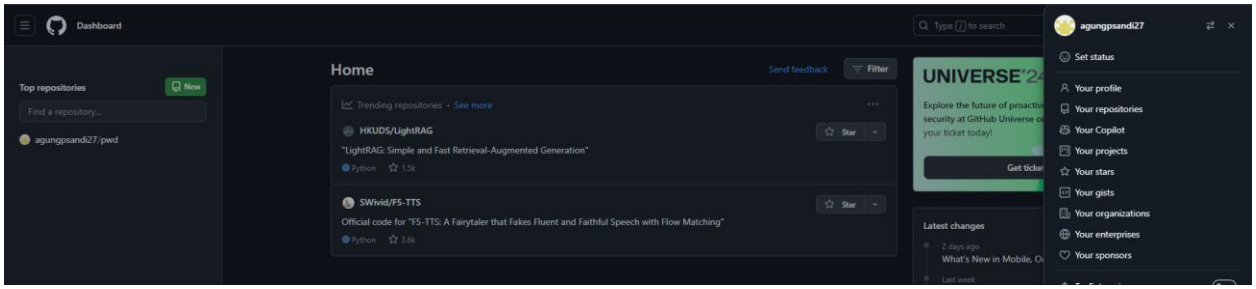
2. Google Collab



3. Kaggle



4. GitHub



5. Praktek Sklearn 02

5.1. Sample Dataset

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

     iris = load_iris()
     iris
```

```
====== ===== ====================\n\n:Missing Attribute Values: None\n:Class Distribution: 33.3% for each of 3 classes.\n:Creator: R.A. Fisher\n:Donor: M
ichael Marshall (MARSHALL%PLU@io.arc.nasa.gov)\n:Date: July, 1988\n\nThe famous Iris database, first used by Sir R.A. Fisher. The dataset is taken\nfrom
Fisher\'s paper. Note that it\'s the same as in R, but not as in the UCI\nMachine Learning Repository, which has two wrong data points.\n\nThis is perhap
s the best known database to be found in the\npattern recognition literature.  Fisher\'s paper is a classic in the field and\nis referenced frequently to
this day.  (See Duda & Hart, for example.)  The\ndata set contains 3 classes of 50 instances each, where each class refers to a\ntype of iris plant.  One
class is linearly separable from the other 2; the\nlatter are NOT linearly separable from each other.\n\n.. dropdown:: References\n\n  - Fisher, R.A. "Th
e use of multiple measurements in taxonomic problems"\n    Annual Eugenics, 7, Part II, 179-188 (1936); also in "Contributions to\n    Mathematical Stati
stics" (John Wiley, NY, 1950).\n  - Duda, R.O., & Hart, P.E. (1973) Pattern Classification and Scene Analysis.\n    (Q327.D83) John Wiley & Sons.  ISBN 0
-471-22361-1.  See page 218.\n  - Dasarathy, B.V. (1980) "Nosing Around the Neighborhood: A New System\n    Structure and Classification Rule for Recogni
tion in Partially Exposed\n    Environments".  IEEE Transactions on Pattern Analysis and Machine\n    Intelligence, Vol. PAMI-2, No. 1, 67-71.\n  - Gate
s, G.W. (1972) "The Reduced Nearest Neighbor Rule".  IEEE Transactions\n    on Information Theory, May 1972, 431-433.\n  - See also: 1988 MLC Proceeding
s, 54-64.  Cheeseman et al"s AUTOCLASS II\n    conceptual clustering system finds 3 classes in the data.\n  - Many, many more ...\n',
 'feature_names': ['sepal length (cm)',
  'sepal width (cm)',
  'petal length (cm)',
  'petal width (cm)'],
 'filename': 'iris.csv',
 'data_module': 'sklearn.datasets.data'}
```

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

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

5.2. Deskripsi dari sample dataset

```
[4]: 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!)
============== ==== ==== ======= ===== ====================

:Missing Attribute Values: None
:Class Distribution: 33.3% for each of 3 classes.
```

5.3. Explanatory & Response Variables | Features & Target

```
[5]: X = iris.data
     X.shape
```

```
[5]: (150, 4)
```

```
[6]: Y = iris.target
     Y.shape
```

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

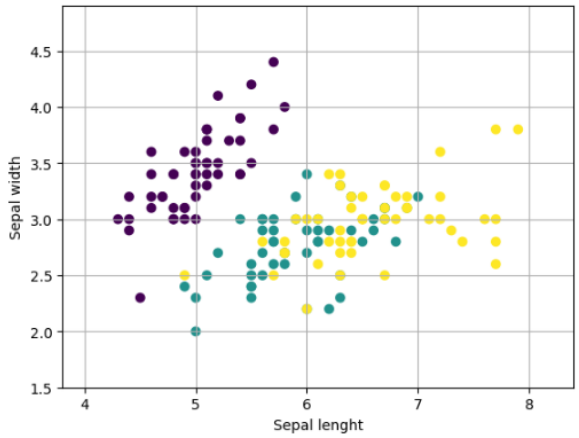
```
[ ]:
```

5.4. Feature & Target Names

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

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

[8]: target_names = iris.target_names
     target_names

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

[ ]:
```

5.5. Visualisasi Data

```
[31]: 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 lenght')
      plt.ylabel('Sepal width')

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



5.6. Training Set & Testing Set

```
[42]: 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: {X_train.shape}')
      print(f'y test: {X_test.shape}')

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

5.7. Load smple dataset sebagai Pandas Data Frame

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

      iris_features_df = iris.data
      iris_features_df
```

[44]:

	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. Prakter Sklearn 03

6.1. Training model Machine Learning

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

     iris = load_iris()

     X = iris.data
     y = iris.target
```

```
[6]: 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)
```

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

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

[9]:

```
KNeighborsClassifier
KNeighborsClassifier(n_neighbors=3)
```

```
[ ]:
```

6.2. Evaluasi model Machine Learning

```
[11]: 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
```

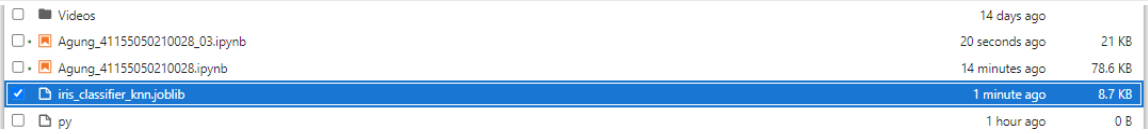
```
[ ]:
```

6.3. Pemanfaatan trained model machine learning

```
[12]: data_baru = [[5, 5, 3, 2],[2, 4, 3, 5]]

      preds = model.predict(data_baru)
      preds

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

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

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

[ ]:
```

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

```
[14]: import joblib

      joblib.dump(model, 'iris_classifier_knn.joblib')

[14]: ['iris_classifier_knn.joblib']

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

[ ]:
```



7. Praktek SKlearn 04

7.1. Persiapan sample dataset

```
[5]: 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

[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]])

[6]: sample_data.shape

[6]: (4, 3)

[ ]:
```

7.2. Teknik data preprocessing 1: binarisasi | binarisation | binarizarion

```
[7]: sample_data

[7]: 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]])

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

[9]: array([[1., 0., 1.],
            [0., 1., 1.],
            [0., 0., 1.],
            [1., 1., 0.]])

[ ]:
```

7.3. Teknik data preprocessing 2: scaling

```
[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]])

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

[13]: scaled_data = preprocessor.fit_transform(sample_data)
      scaled_data

[13]: 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: normalisasi | normalisation | normalization

```
[14]: sample_data

[14]: 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]])

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

[15]: 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]])

[ ]:
```