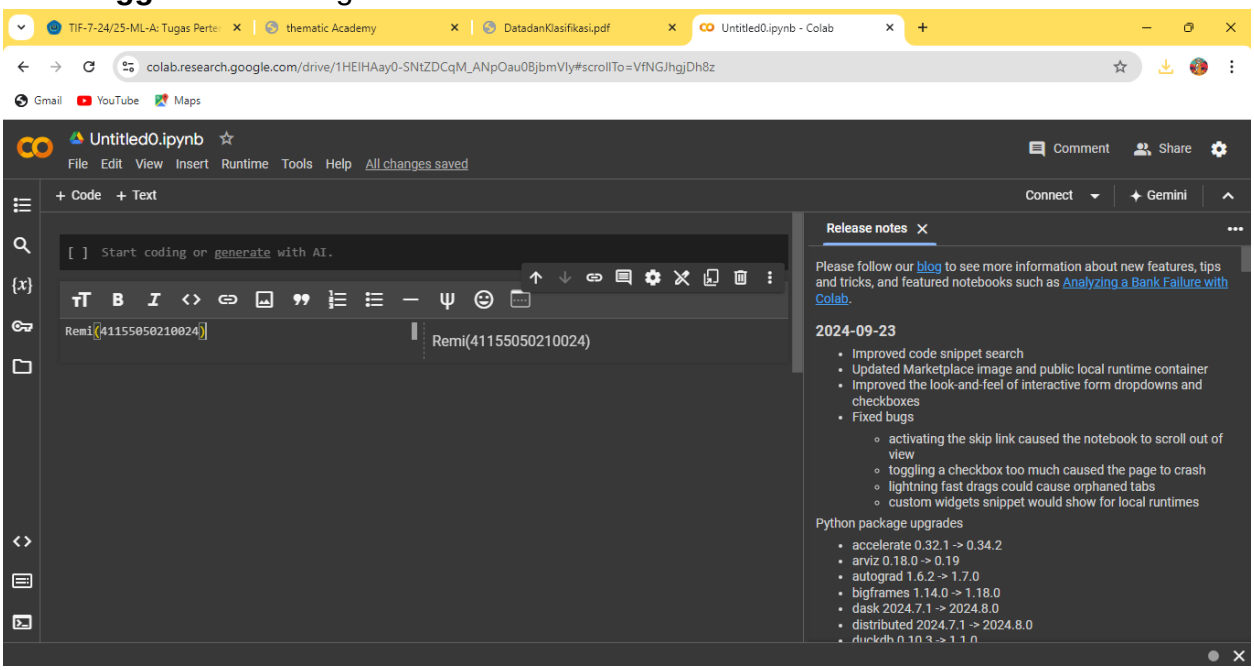
TUGAS MACHINE LEARNING 1

Nama : Remi Maulani H

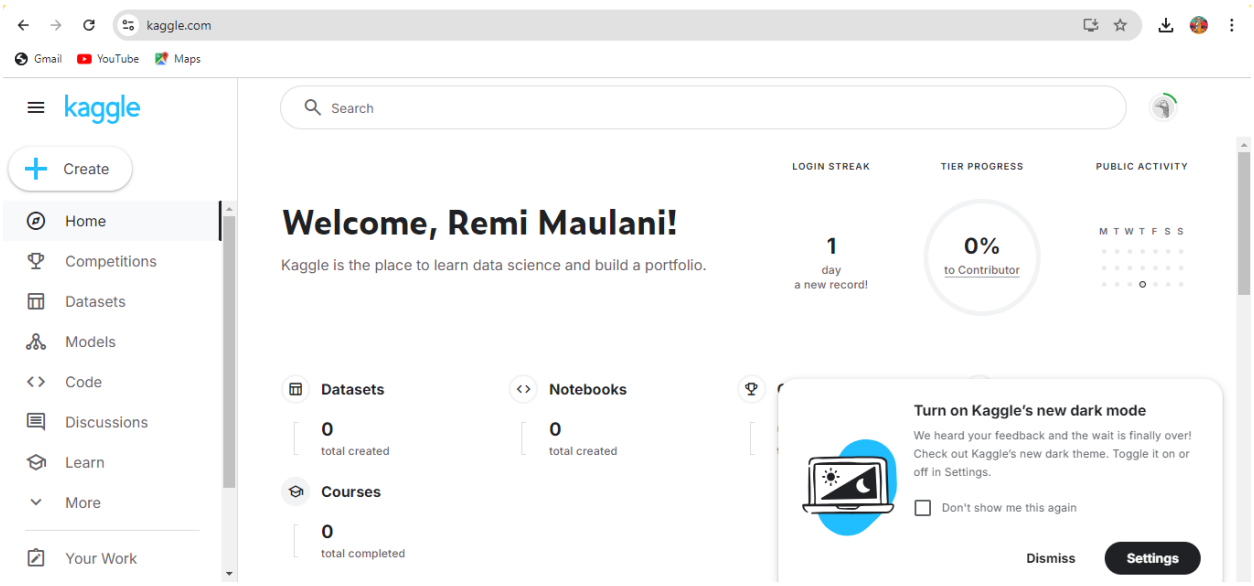
Npm : 41155050210024

Kelas : INF-A1

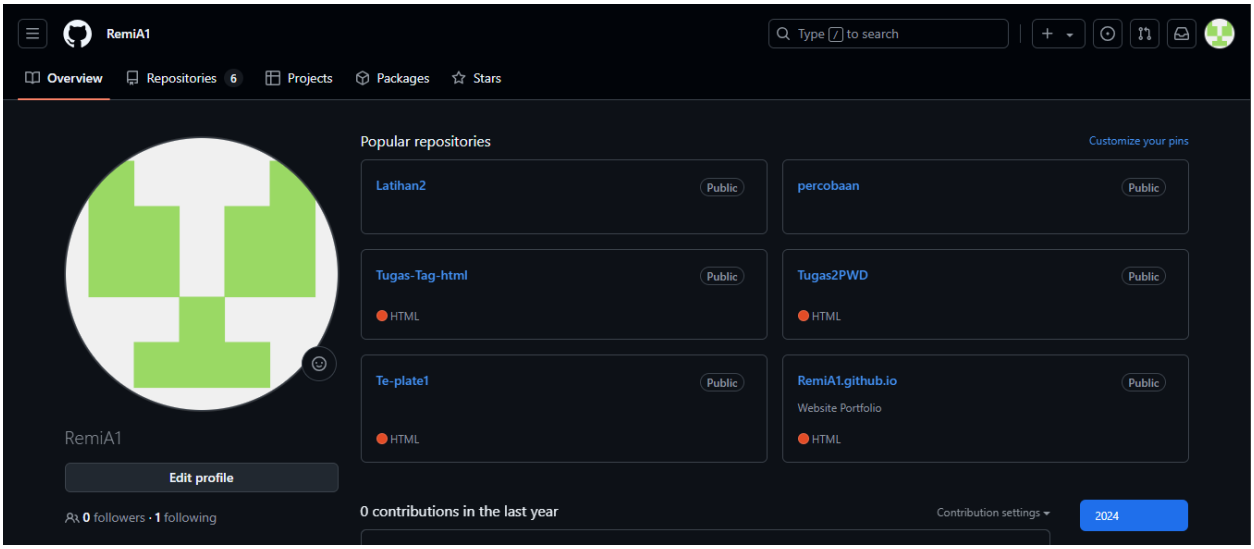
1. **Menggunakan** Google Collab



2. **Membuat akun kaggle**

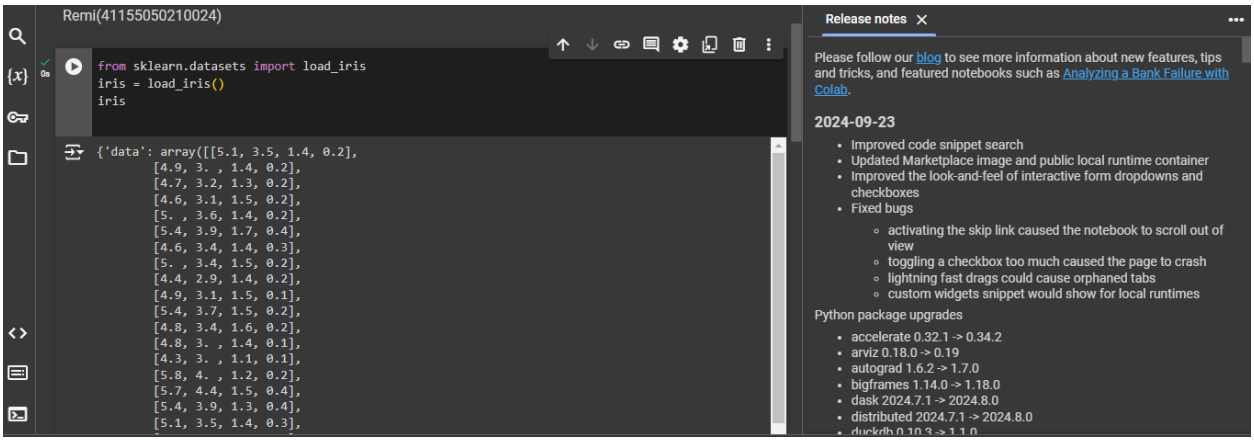


3. Akun GitHub



5. Melakukan Praktek Youtube 1

- 5.1 Load sample dataset



- 5.2 Metadata | Deskripsi dari sample dataset

```
print(iris.DESCR)
```

```
: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 (MARSHALL%PLU@io.arc.nasa.gov)
:Date: July, 1988

The famous Iris database, first used by Sir R.A. Fisher. The dataset is taken
from Fisher's paper. Note that it's the same as in R, but not as in the UCI
Machine Learning Repository, which has two wrong data points.

This is perhaps the best known database to be found in the
pattern recognition literature.  Fisher's paper is a classic in the field and
is referenced frequently to this day.  (See Duda & Hart, for example.)  The
data set contains 3 classes of 50 instances each, where each class refers to a
type of iris plant.  One class is linearly separable from the other 2; the
latter are NOT linearly separable from each other.

.. dropdown:: References

  - Fisher, R.A. "The use of multiple measurements in taxonomic problems"
    Annual Eugenics, 7, Part II, 179-188 (1936); also in "Contributions to
    Mathematical Statistics" (John Wiley, NY, 1950).
  - Duda, R.O., & Hart, P.E. (1973) Pattern Classification and Scene Analysis.
    (Q327.D83) John Wiley & Sons.  ISBN 0-471-22361-1.  See page 218.
  - Dasarathy, B.V. (1980) "Nosing Around the Neighborhood: A New System
    Structure and Classification Rule for Recognition in Partially Exposed
    Environments".  IEEE Transactions on Pattern Analysis and Machine
```

- 5.3 Explanatory & Response Variables | Features & Target

```
[18] X = iris.data
     X.shape
     #X
```

```
(150, 4)
```

```
y = iris.target
y.shape
#y
```

```
(150,)
```

```
+ Code  + Text

X = iris.data
#X.shape
X
```

```
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],
       [5.7, 3.8, 1.7, 0.3],
       [5.1, 3.8, 1.5, 0.3],
       [5.4, 3.4, 1.7, 0.2],
```

```
[21] y = iris.target
     #y.shape
     y
```

```
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, 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, 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, 2, 2])
```

- 5.4 Feature & Target Names

```
+ Code  + Text

[28] feature_names = iris.feature_names
     feature_names
```

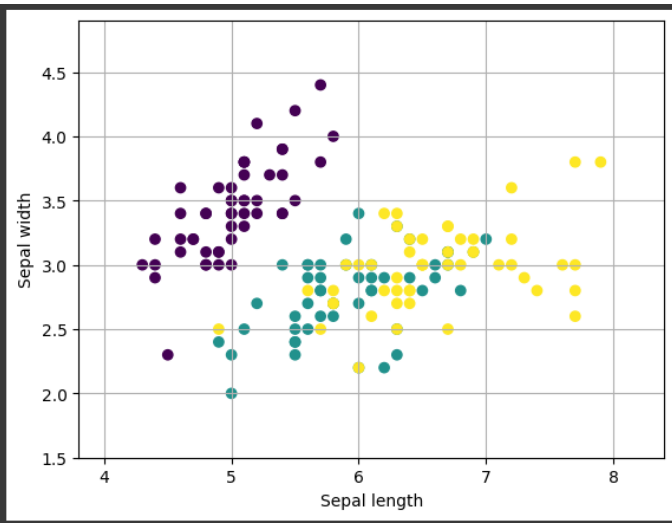
```
['sepal length (cm)',
 'sepal width (cm)',
 'petal length (cm)',
 'petal width (cm)']
```

```
target_names = iris.target_names
target_names
```

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

- 5.5 Visualisasi Data

```python
import matplotlib.pyplot as plt

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.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

```python
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, 4)
X test: (45, 4)
y train: (105,)
y test: (45,)
```

- 5.7 Load sample dataset sebagai Pandas Data Frame

```
iris = load_iris(as_frame=True)

iris_features_df = iris.data
iris_features_df
```

	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

0s completed at 11:35 PM

	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

6.0. Melakukan praktek dari Youtube 2

- 6.1. Persiapan dataset | Loading & splitting dataset

```
[35] from sklearn.datasets import load_iris

     iris = load_iris()
     X = iris.data
     y = iris.target
```

```
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)
```

- 6.2. Training model Machine Learning

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

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

- 6.3. Evaluasi model Machine Learning

```
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
```

- 6.4. Pemanfaatan trained model machine learning

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

preds = model.predict(data_baru)
preds
```

```
array([1, 2])
```

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

```
Hasil Prediksi : ['versicolor', 'virginica']
```

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

```
[45] import joblib

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

```
['iris_classifier_knn.joblib']
```

```
production_model = joblib.load('iris_classifier_knn.joblib')
```

7.0. Melakukan praktek dari Youtube 3

- 7.1. Persiapan sample dataset

```
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
```

```
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.shape
```

```
(4, 3)
```

- 7.2. Teknik data preprocessing 1: binarisation

```
+ Code  + Text

[51] sample_data

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]])

preprocessor = preprocessing.Binarizer(threshold=0.5)
binarised_data = preprocessor.transform(sample_data)
binarised_data

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

- 7.3. Teknik data preprocessing 2: scaling

```
+ Code  + Text

[53] 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]])

preprocessor = preprocessing.MinMaxScaler(feature_range=(0, 1))
preprocessor.fit(sample_data)
scaled_data = preprocessor.transform(sample_data)
scaled_data

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

scaled_data = preprocessor.fit_transform(sample_data)
scaled_data

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: normalization

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

```
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]])
```

```
l1_normalised_data = preprocessing.normalize(sample_data, norm='l2')
l1_normalised_data
```

```
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 ]])
```