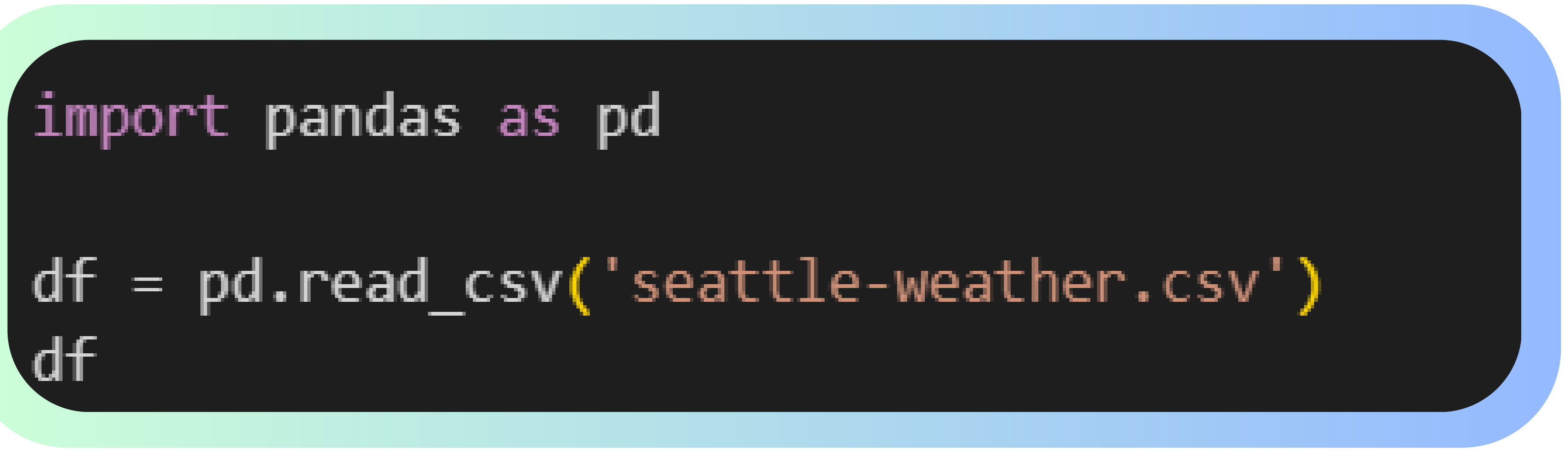
Machine Learning
Classification

By Muhammad Ridho S.

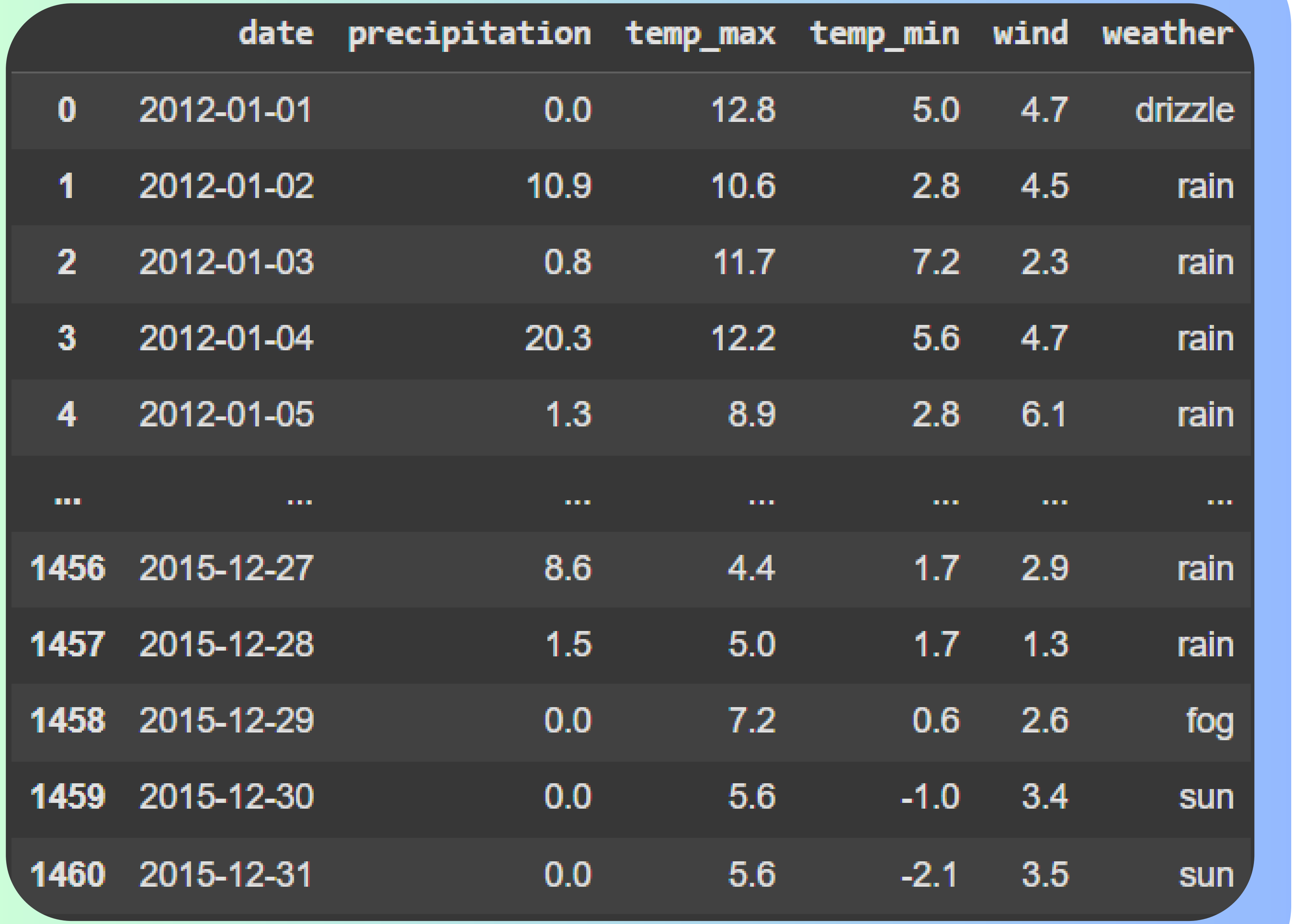
Tools

pandas
colab
NumPy
kaggle

Dataset



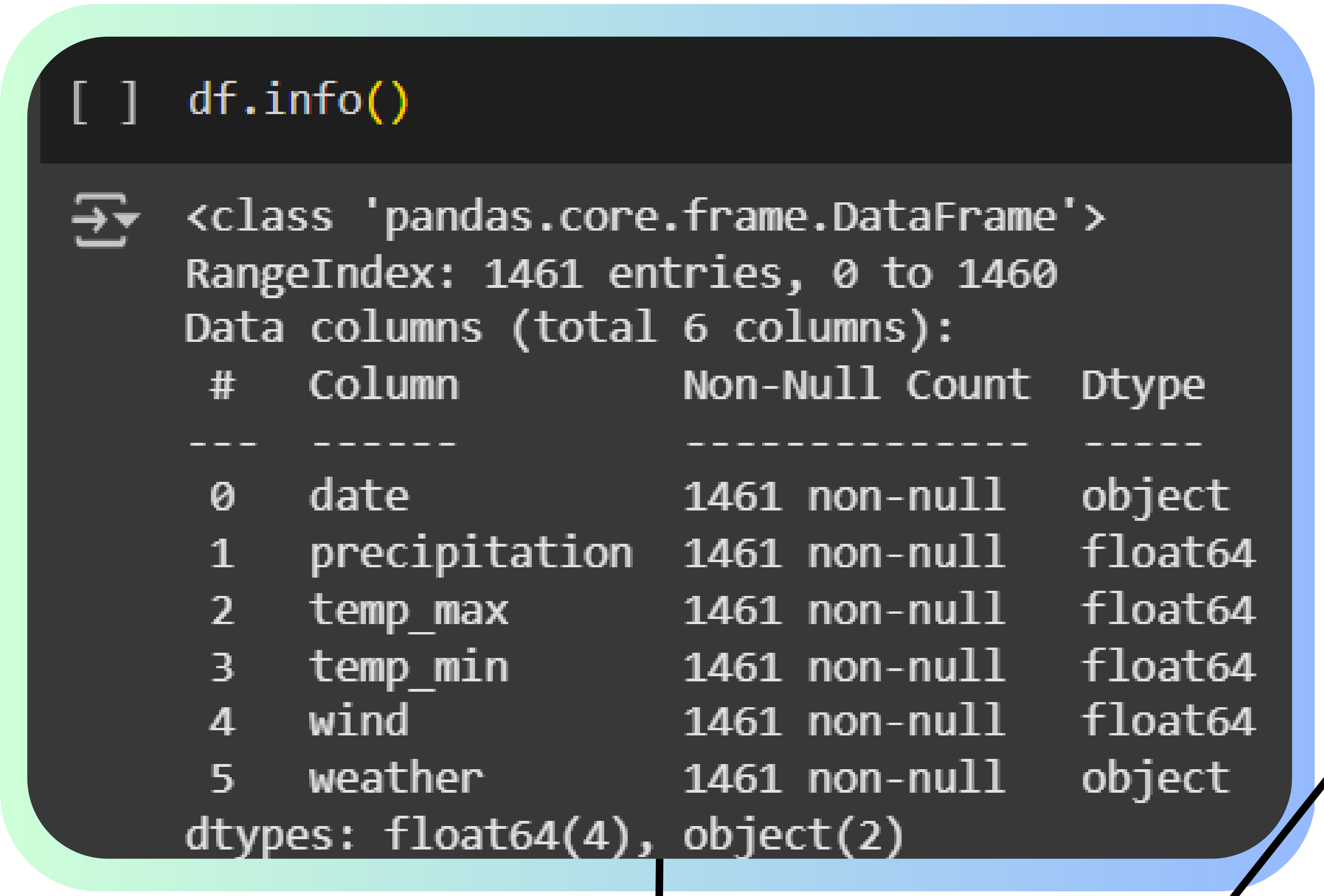
```
import pandas as pd

df = pd.read_csv('seattle-weather.csv')
df
```



	date	precipitation	temp_max	temp_min	wind	weather
0	2012-01-01	0.0	12.8	5.0	4.7	drizzle
1	2012-01-02	10.9	10.6	2.8	4.5	rain
2	2012-01-03	0.8	11.7	7.2	2.3	rain
3	2012-01-04	20.3	12.2	5.6	4.7	rain
4	2012-01-05	1.3	8.9	2.8	6.1	rain
...	...	...	...	...	...	...
1456	2015-12-27	8.6	4.4	1.7	2.9	rain
1457	2015-12-28	1.5	5.0	1.7	1.3	rain
1458	2015-12-29	0.0	7.2	0.6	2.6	fog
1459	2015-12-30	0.0	5.6	-1.0	3.4	sun
1460	2015-12-31	0.0	5.6	-2.1	3.5	sun

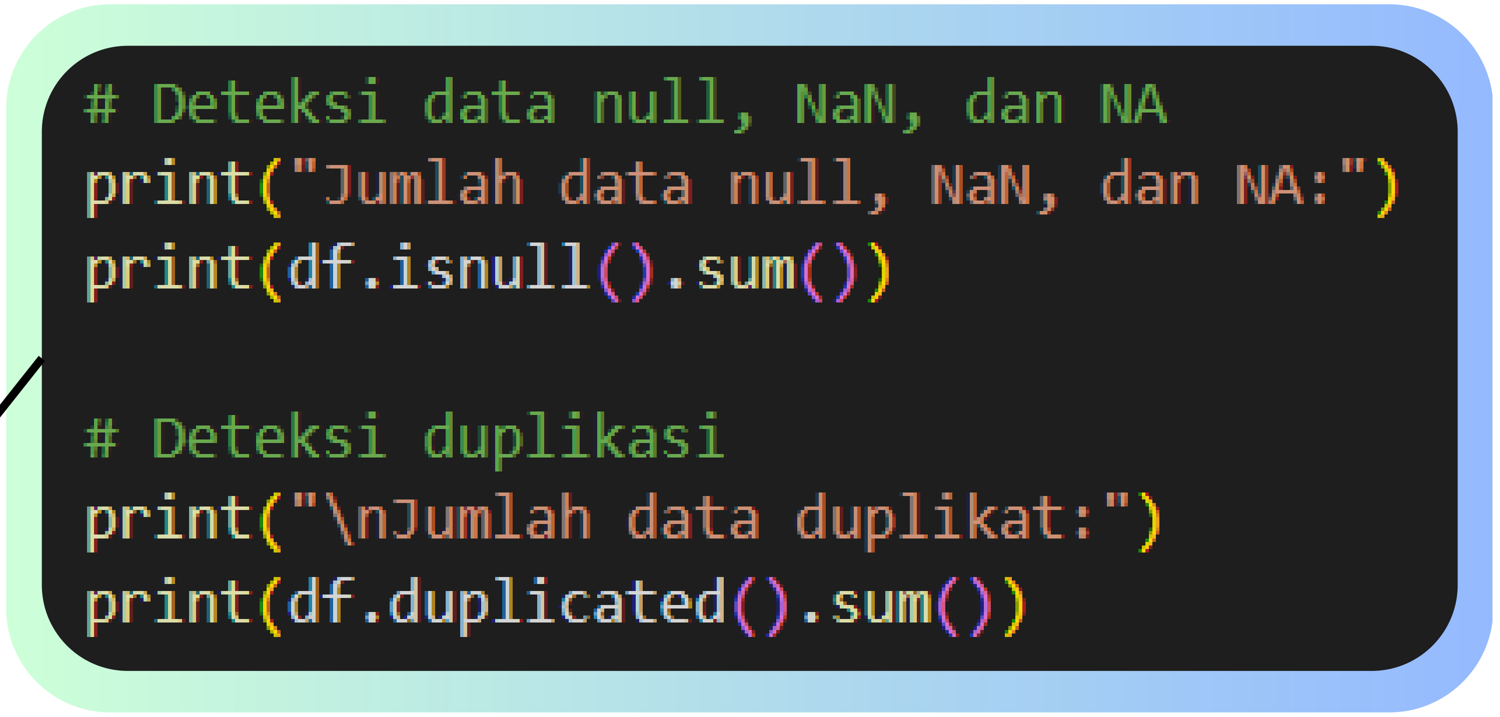
This dataset contains weather data for a specific date and is intended for weather prediction. The table includes the following columns: date, precipitation, max temperature, min temperature, wind, and weather conditions.

Exploratory Data Analysis

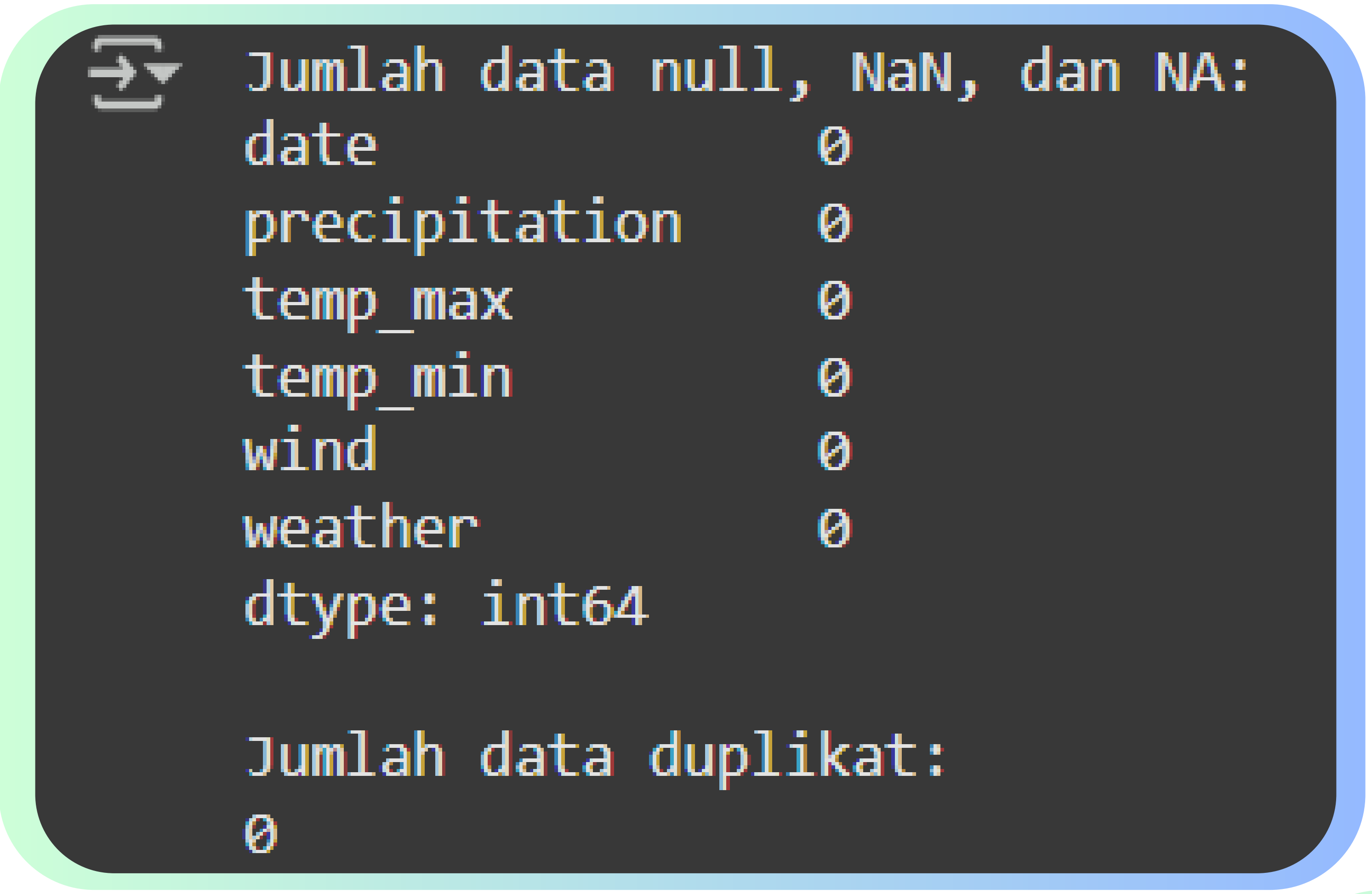


```
[ ]  df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1461 entries, 0 to 1460
Data columns (total 6 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   date           1461 non-null   object 
 1   precipitation  1461 non-null   float64
 2   temp_max       1461 non-null   float64
 3   temp_min       1461 non-null   float64
 4   wind           1461 non-null   float64
 5   weather        1461 non-null   object 
dtypes: float64(4), object(2)
```

Checking data type



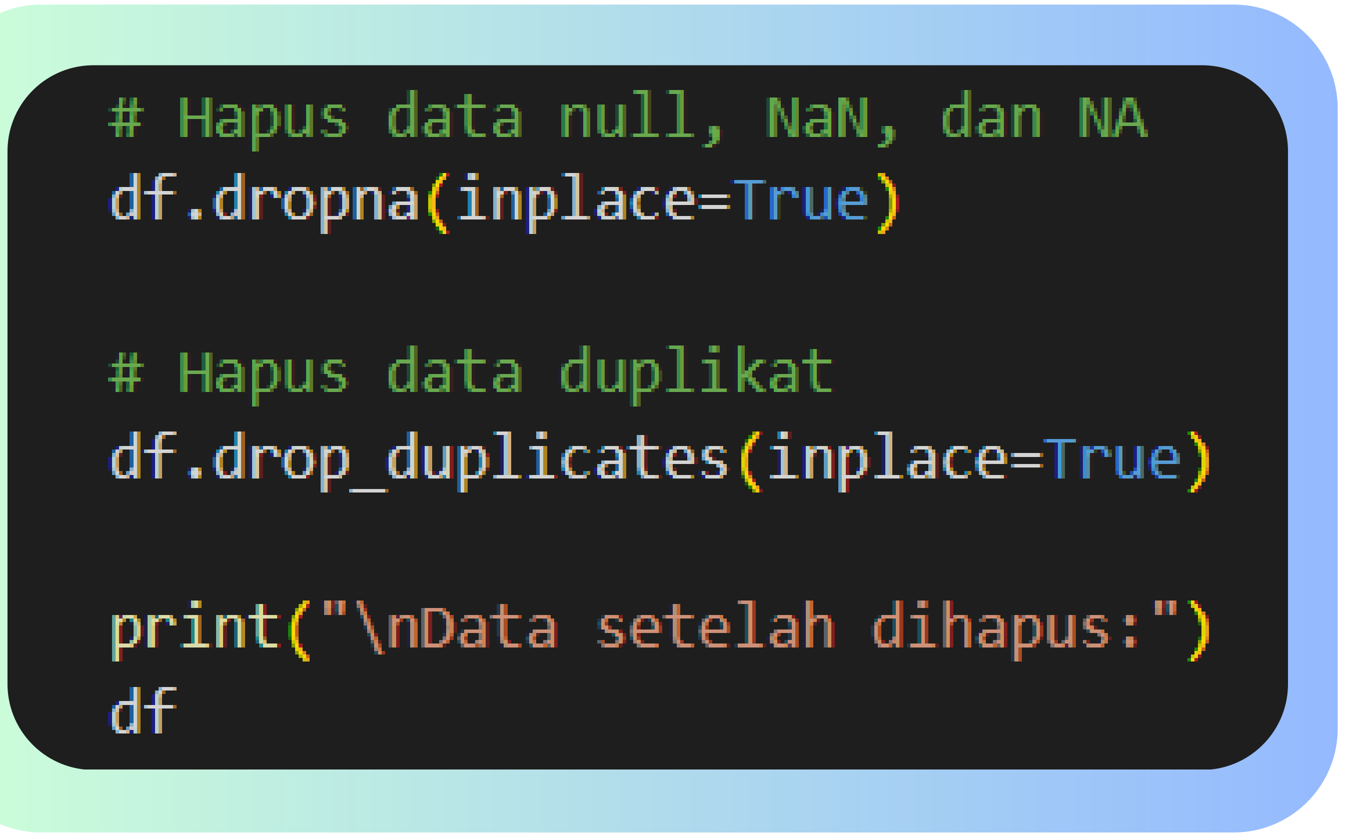
```
# Deteksi data null, NaN, dan NA
print("Jumlah data null, NaN, dan NA:")
print(df.isnull().sum())

# Deteksi duplikasi
print("\nJumlah data duplikat:")
print(df.duplicated().sum())
```



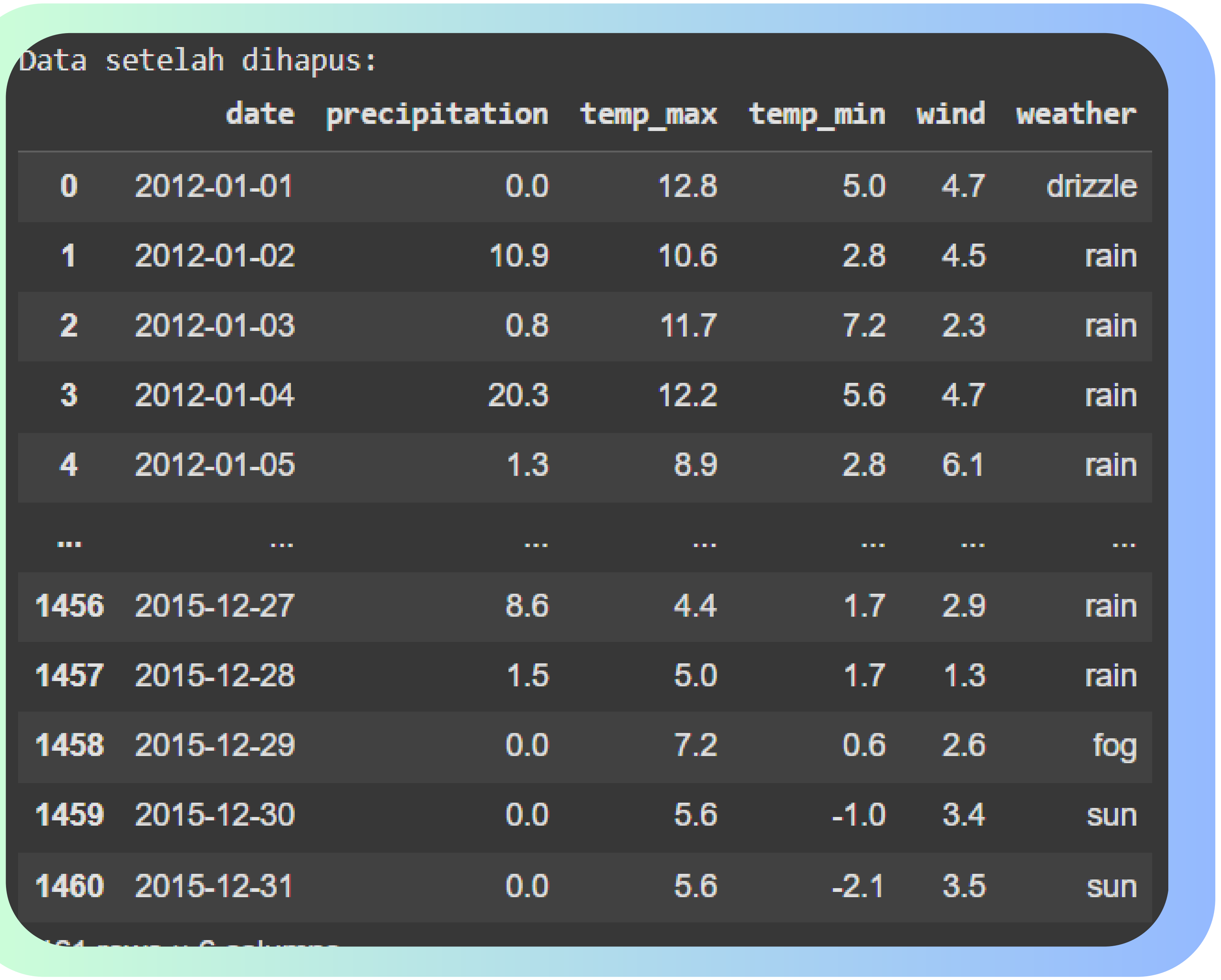
```
Jumlah data null, NaN, dan NA:
date             0
precipitation    0
temp_max         0
temp_min         0
wind             0
weather          0
dtype: int64

Jumlah data duplikat:
0
```

Checking Null data and Duplicate data

Data Cleaning



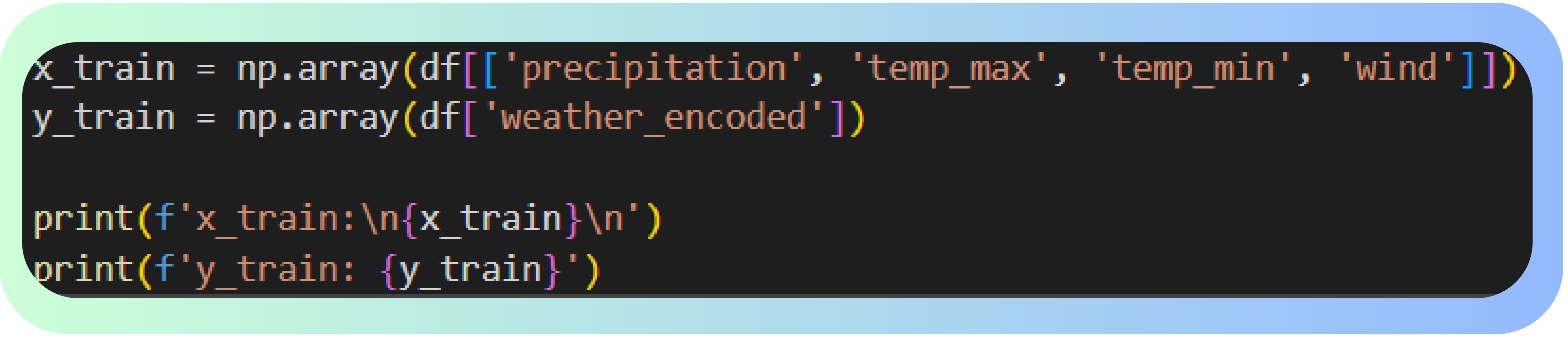
```
# Hapus data null, NaN, dan NA
df.dropna(inplace=True)

# Hapus data duplikat
df.drop_duplicates(inplace=True)

print("\nData setelah dihapus:")
df
```



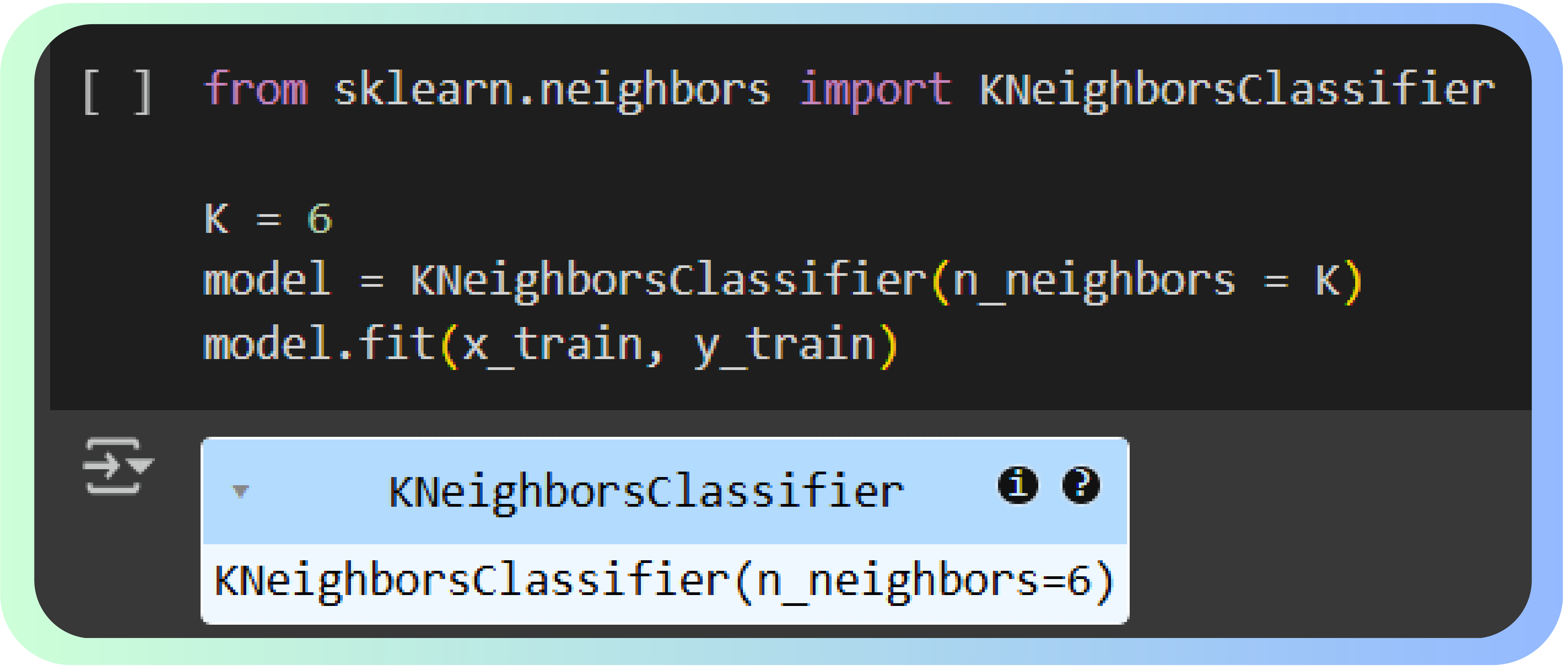
```
Data setelah dihapus:
            date  precipitation  temp_max  temp_min  wind  weather
0     2012-01-01            0.0      12.8       5.0   4.7  drizzle
1     2012-01-02           10.9      10.6       2.8   4.5     rain
2     2012-01-03            0.8      11.7       7.2   2.3     rain
3     2012-01-04           20.3      12.2       5.6   4.7     rain
4     2012-01-05            1.3       8.9       2.8   6.1     rain
...          ...            ...       ...       ...   ...      ...
1456  2015-12-27            8.6       4.4       1.7   2.9     rain
1457  2015-12-28            1.5       5.0       1.7   1.3     rain
1458  2015-12-29            0.0       7.2       0.6   2.6      fog
1459  2015-12-30            0.0       5.6      -1.0   3.4      sun
1460  2015-12-31            0.0       5.6      -2.1   3.5      sun
```

This script is useful for cleaning null and duplicate data to prevent errors.

Machine Learning KNN



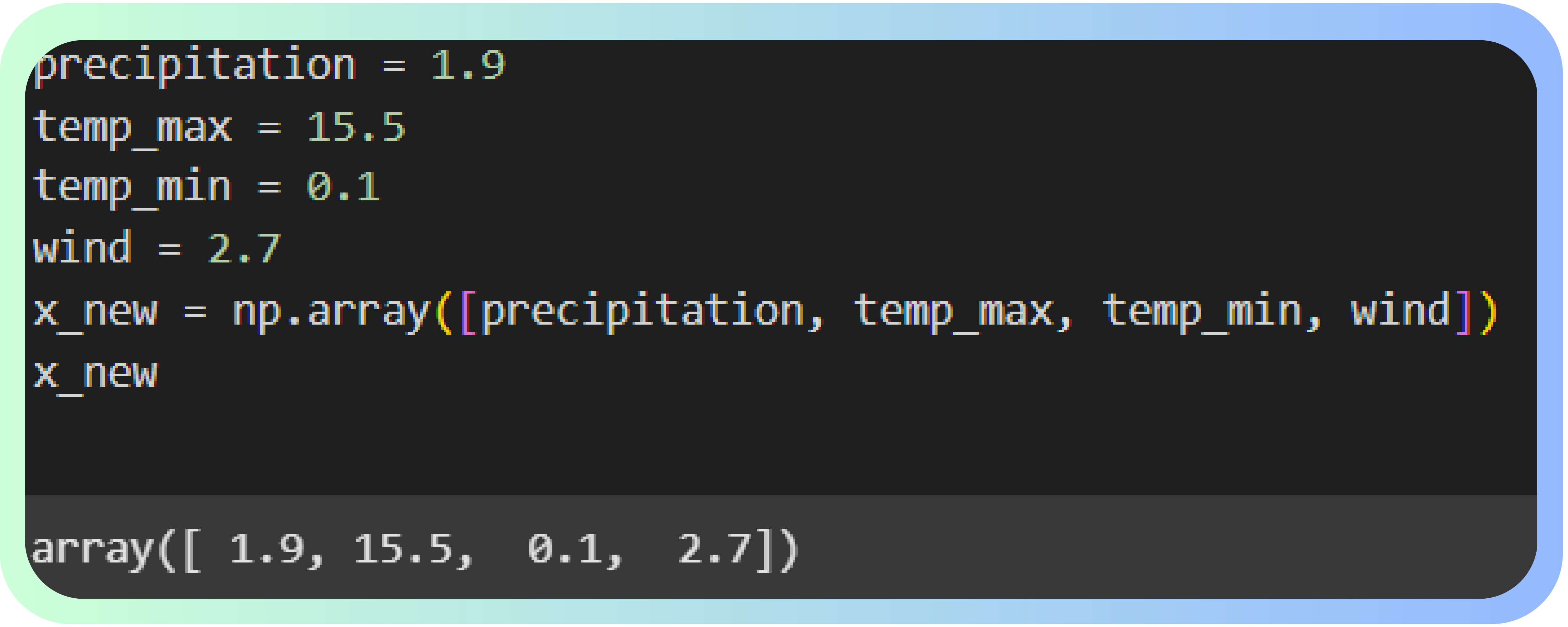
```python
x_train = np.array(df[['precipitation', 'temp_max', 'temp_min', 'wind']])
y_train = np.array(df['weather_encoded'])

print(f'x_train:\n{x_train}\n')
print(f'y_train: {y_train}')
```



```python
from sklearn.neighbors import KNeighborsClassifier

K = 6
model = KNeighborsClassifier(n_neighbors = K)
model.fit(x_train, y_train)
```

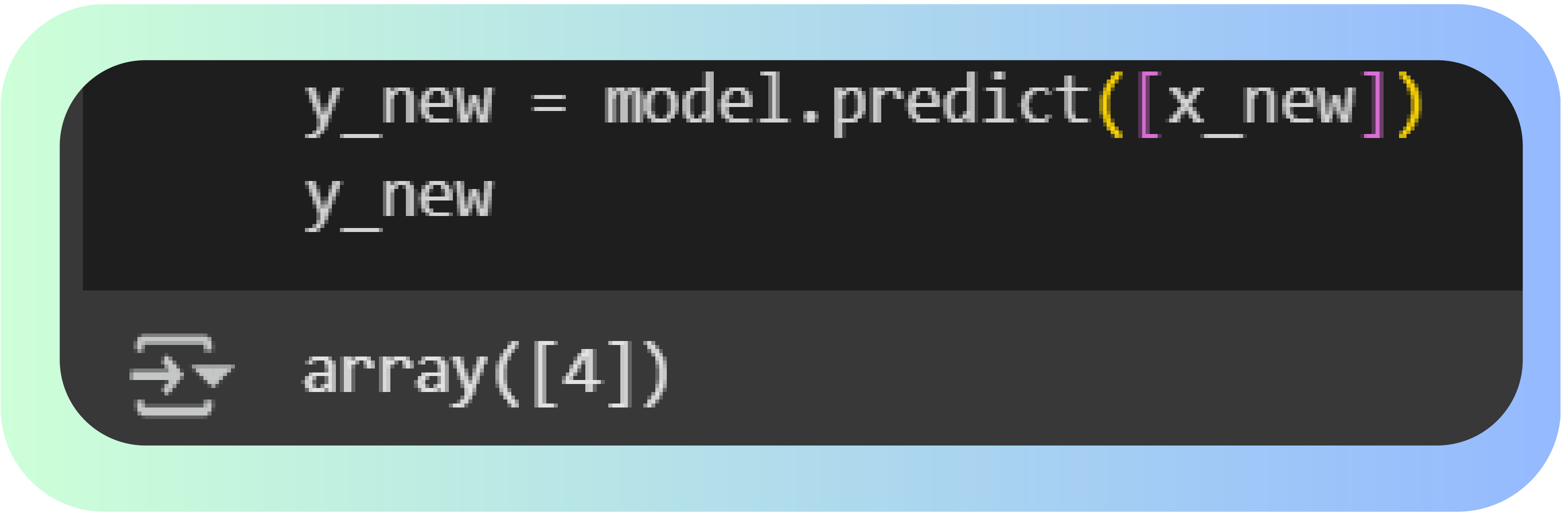
```
KNeighborsClassifier
KNeighborsClassifier(n_neighbors=6)
```

Machine Learning KNN



```
precipitation = 1.9
temp_max = 15.5
temp_min = 0.1
wind = 2.7
x_new = np.array([precipitation, temp_max, temp_min, wind])
x_new
```

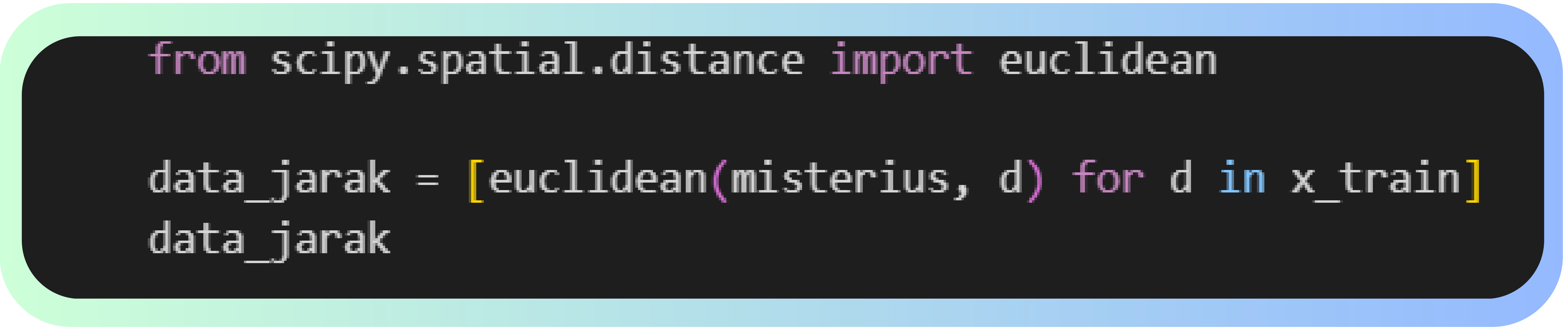
```
array([ 1.9, 15.5,  0.1,  2.7])
```



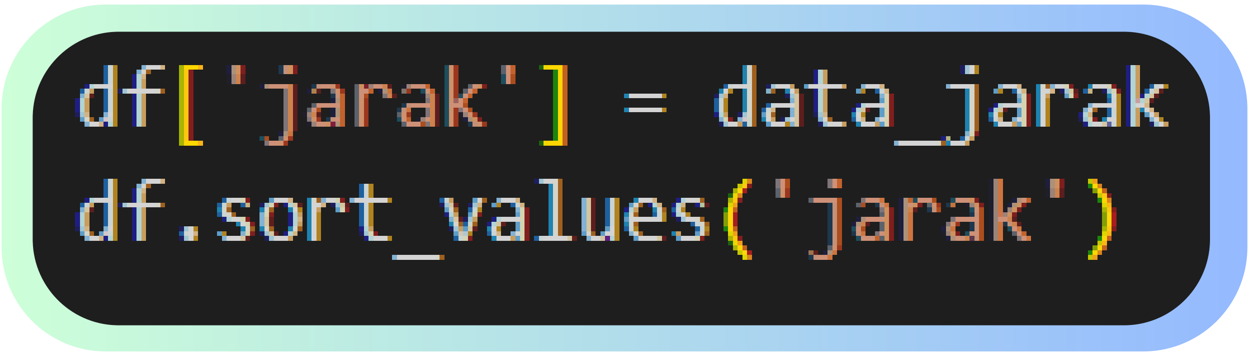
```
y_new = model.predict([x_new])
y_new
```

```
array([4])
```

Machine Learning KNN

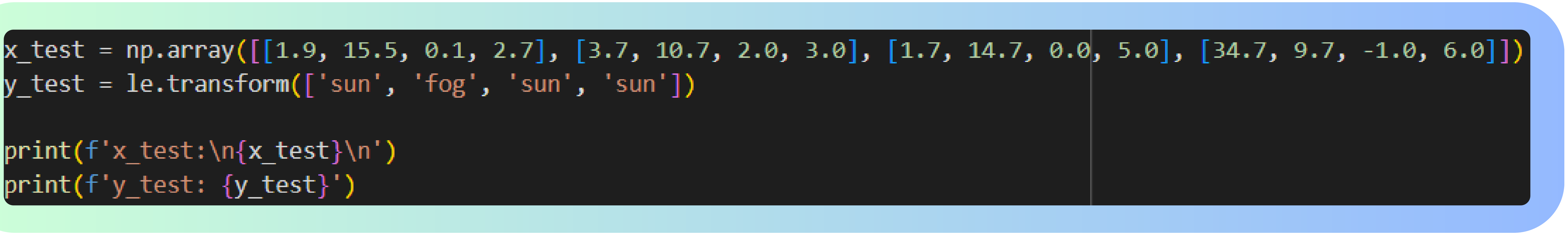


```
from scipy.spatial.distance import euclidean

data_jarak = [euclidean(misterius, d) for d in x_train]
data_jarak
```

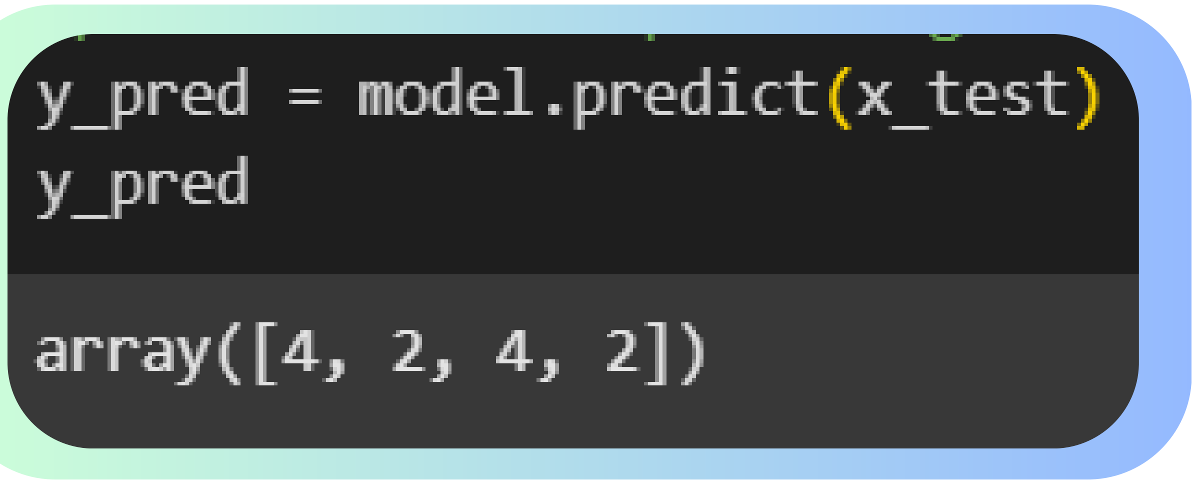


```
df['jarak'] = data_jarak
df.sort_values('jarak')
```

Machine Learning KNN

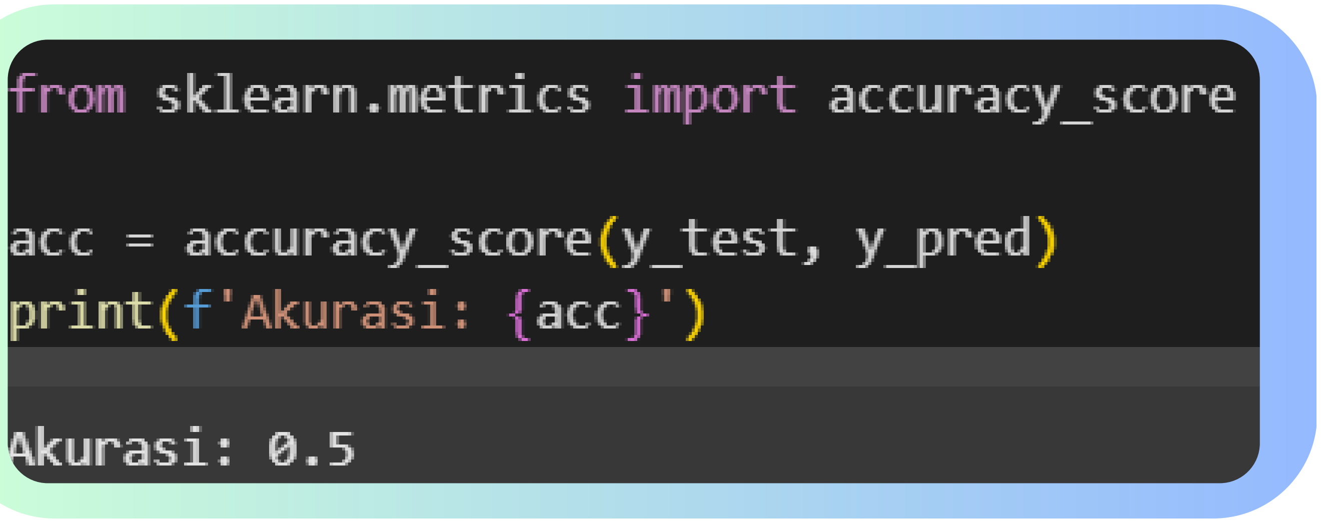


```
x_test = np.array([[1.9, 15.5, 0.1, 2.7], [3.7, 10.7, 2.0, 3.0], [1.7, 14.7, 0.0, 5.0], [34.7, 9.7, -1.0, 6.0]])
y_test = le.transform(['sun', 'fog', 'sun', 'sun'])

print(f'x_test:\n{x_test}\n')
print(f'y_test: {y_test}')
```



```
y_pred = model.predict(x_test)
y_pred
```

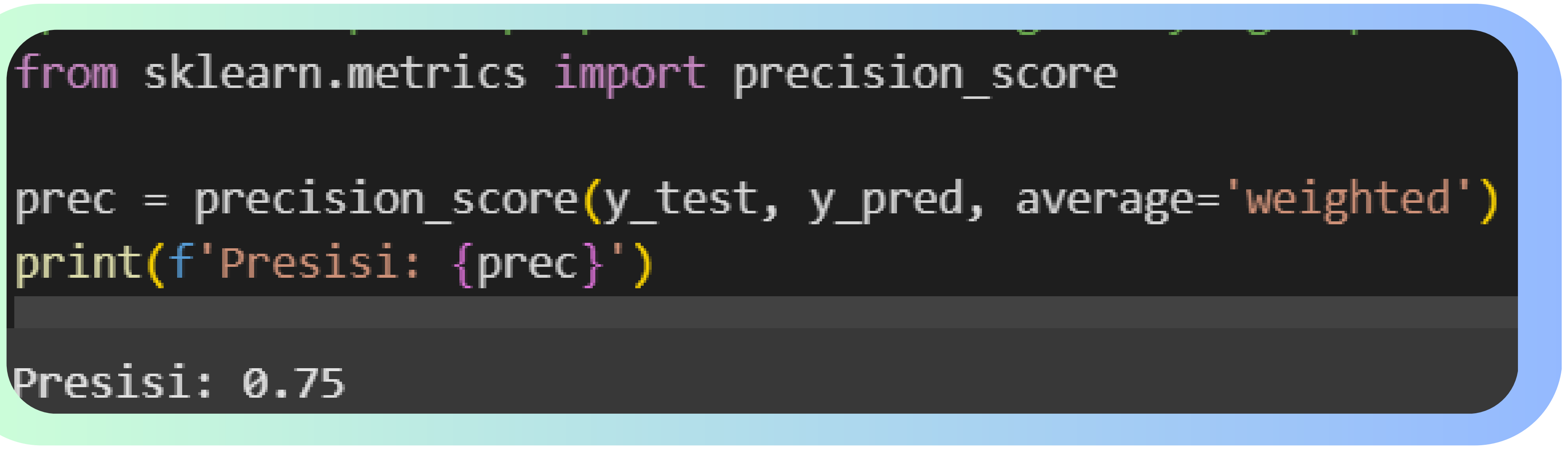
```
array([4, 2, 4, 2])
```



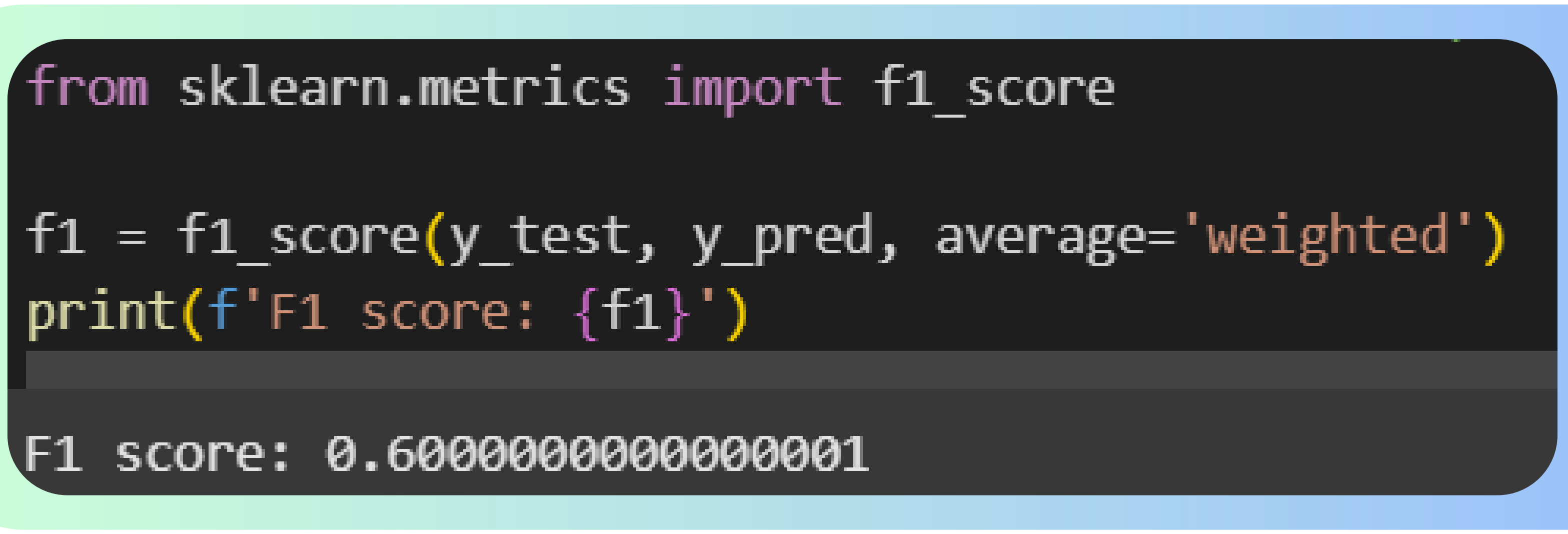
```
from sklearn.metrics import accuracy_score

acc = accuracy_score(y_test, y_pred)
print(f'Akurasi: {acc}')
```

```
Akurasi: 0.5
```

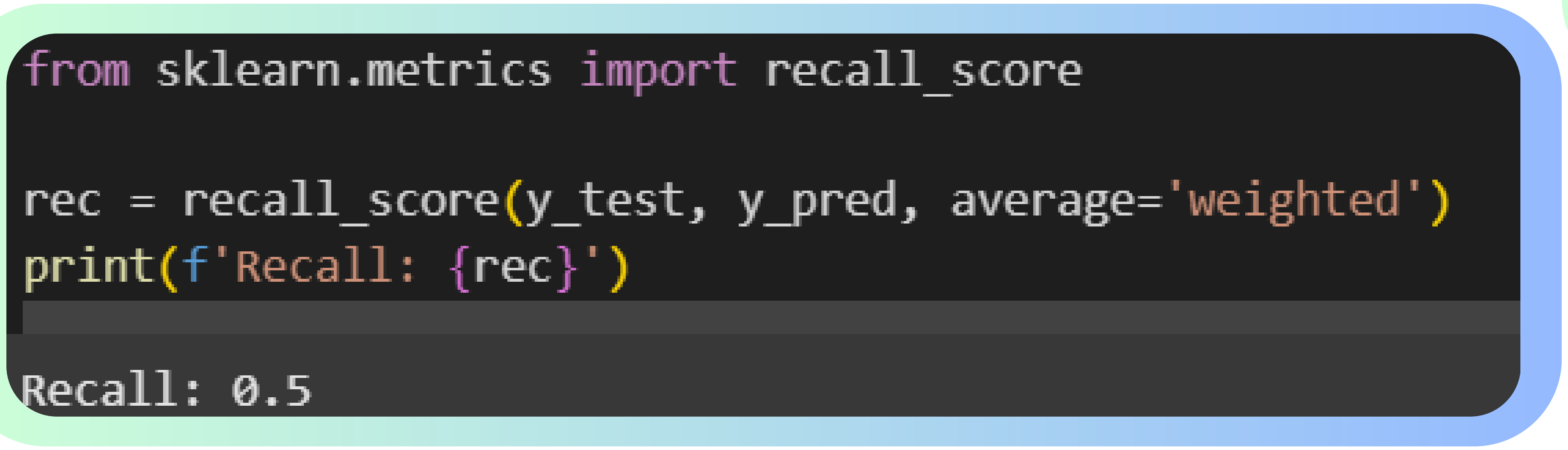
Machine Learning KNN



```
from sklearn.metrics import precision_score

prec = precision_score(y_test, y_pred, average='weighted')
print(f'Presisi: {prec}')

Presisi: 0.75
```

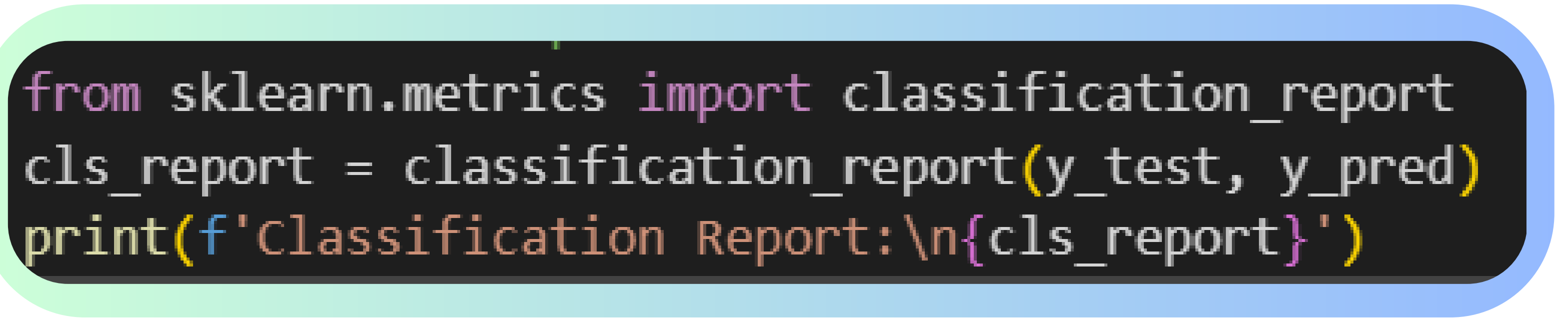


```
from sklearn.metrics import f1_score

f1 = f1_score(y_test, y_pred, average='weighted')
print(f'F1 score: {f1}')

F1 score: 0.6000000000000001
```

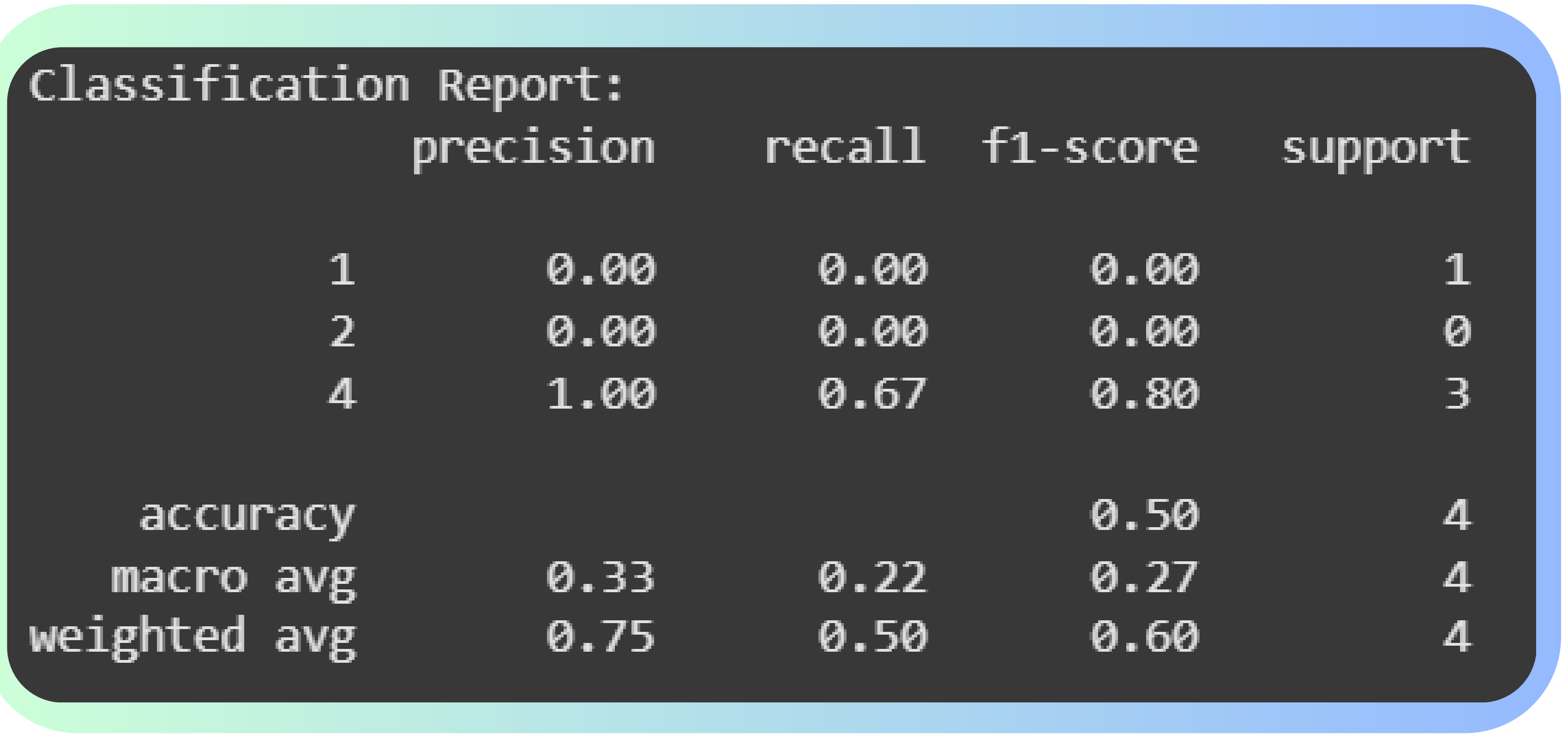


```
from sklearn.metrics import recall_score

rec = recall_score(y_test, y_pred, average='weighted')
print(f'Recall: {rec}')

Recall: 0.5
```

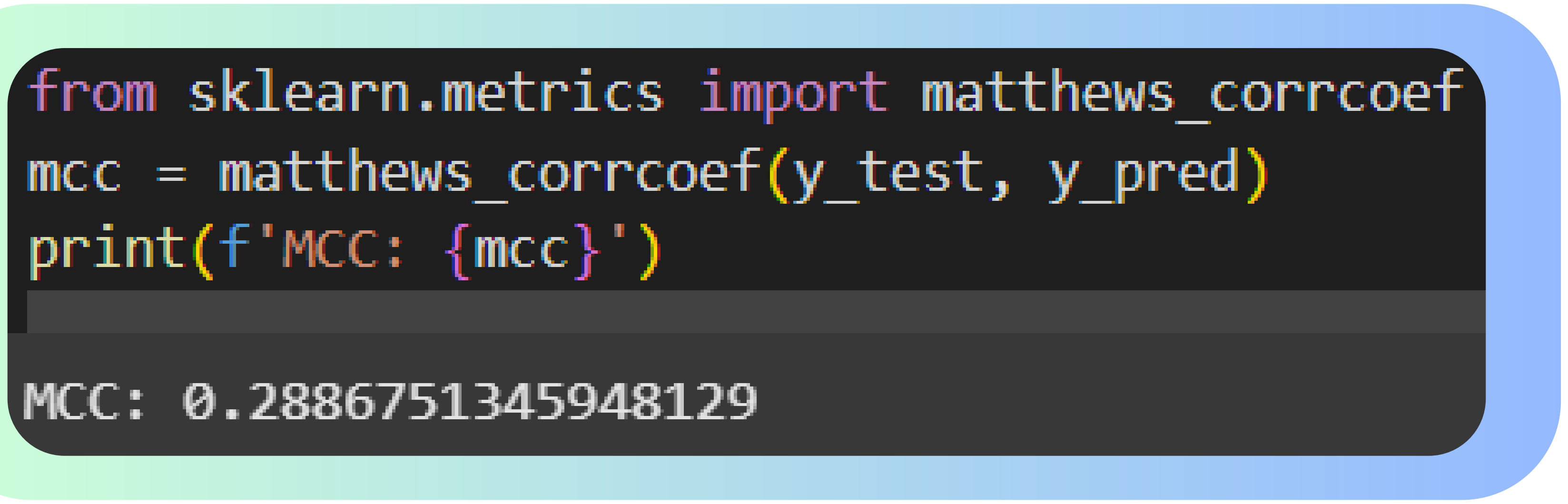
Machine Learning KNN



```
from sklearn.metrics import classification_report
cls_report = classification_report(y_test, y_pred)
print(f'Classification Report:\n{cls_report}')
```



```
Classification Report:
              precision    recall  f1-score   support

           1       0.00      0.00      0.00         1
           2       0.00      0.00      0.00         0
           4       1.00      0.67      0.80         3

    accuracy                           0.50         4
   macro avg       0.33      0.22      0.27         4
weighted avg       0.75      0.50      0.60         4
```



```
from sklearn.metrics import matthews_corrcoef
mcc = matthews_corrcoef(y_test, y_pred)
print(f'MCC: {mcc}')
```

```
MCC: 0.2886751345948129
```

THANKYOU!