

Exploratory Data Science

Dibimbing Digital Skill Fair 39

This code imports the Iris dataset using pandas and displays 150 flower samples with sepal, petal sizes, and species. It's ideal for species classification, pattern analysis, and ML model testing.

```
import pandas as pd

df = pd.read_csv('/content/Iris.csv')
```

[] df

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	1	5.1	3.5	1.4	0.2	Iris-setosa
1	2	4.9	3.0	1.4	0.2	Iris-setosa
2	3	4.7	3.2	1.3	0.2	Iris-setosa
3	4	4.6	3.1	1.5	0.2	Iris-setosa
4	5	5.0	3.6	1.4	0.2	Iris-setosa
...
145	146	6.7	3.0	5.2	2.3	Iris-virginica
146	147	6.3	2.5	5.0	1.9	Iris-virginica
147	148	6.5	3.0	5.2	2.0	Iris-virginica
148	149	6.2	3.4	5.4	2.3	Iris-virginica
149	150	5.9	3.0	5.1	1.8	Iris-virginica

150 rows × 6 columns

This code uses the Pandas library to inspect the structure of the Iris dataset stored in the variable `df`. The method `df.info()` provides a concise summary of the dataset, including:

- The total number of entries (150 rows, indexed from 0 to 149).
- The total number of columns (6 in total).
- The name, count of non-null values, and data type of each column.
- Memory usage of the DataFrame (approximately 7.2 KB).

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 6 columns):
 #   Column              Non-Null Count  Dtype
---  -
 0   Id                  150 non-null   int64
 1   SepalLengthCm       150 non-null   float64
 2   SepalWidthCm        150 non-null   float64
 3   PetalLengthCm       150 non-null   float64
 4   PetalWidthCm        150 non-null   float64
 5   Species              150 non-null   object
dtypes: float64(4), int64(1), object(1)
memory usage: 7.2+ KB
```

The first code `df.isna().sum()` checks for the number of missing values (NaN) in each column of the DataFrame `df`, and the result shows that there are no missing values. The second code `df.describe()` provides descriptive statistics such as mean, standard deviation, minimum, maximum, and quartile values for all numeric columns, offering a quick overview of the data distribution and central tendencies.

```
df.isna().sum()
```

	0
Id	0
SepalLengthCm	0
SepalWidthCm	0
PetalLengthCm	0
PetalWidthCm	0
Species	0

dtype: int64

```
[ ] df.describe()
```

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm
count	150.000000	150.000000	150.000000	150.000000	150.000000
mean	75.500000	5.843333	3.054000	3.758667	1.198667
std	43.445368	0.828066	0.433594	1.764420	0.763161
min	1.000000	4.300000	2.000000	1.000000	0.100000
25%	38.250000	5.100000	2.800000	1.600000	0.300000
50%	75.500000	5.800000	3.000000	4.350000	1.300000
75%	112.750000	6.400000	3.300000	5.100000	1.800000
max	150.000000	7.900000	4.400000	6.900000	2.500000

The code `df['Species'].describe()` provides a descriptive statistical summary for the categorical column `Species` in the DataFrame `df`. The output shows that there are 150 data entries (`count`), with 3 unique categories (`unique`), including Iris species such as `Iris-setosa`, `Iris-versicolor`, and `Iris-virginica`. The most frequent category is `Iris-setosa` (`top`), which appears 50 times (`freq`). This summary helps in understanding the distribution of categories within the column.

```
df['Species'].describe()
```



Species

count	150
unique	3
top	Iris-setosa
freq	50

`dtype: object`

The code iterates through each column in the DataFrame `df` and fills missing values (`NaN`) depending on the data type: for numeric columns, missing values are filled with the mean, while for categorical columns, they are filled with the mode (most frequent value). The purpose is to clean the data by handling missing values before analysis or building machine learning models.

```
# Mengatasi missing value
for column in df.columns:
    if df[column].dtype == 'object':
        # Jika kolom bertipe object, isi dengan mode
        df[column].fillna(df[column].mode()[0], inplace=True)
    else:
        # Jika kolom bertipe numerik, isi dengan mean
        df[column].fillna(df[column].mean(), inplace=True)
```

The code `df.isna().sum()` is used to count the number of missing values (NaN) in each column of the DataFrame `df`. The result shows that there are no missing values in any of the columns, including `Id`, `SepalLengthCm`, `SepalWidthCm`, `PetalLengthCm`, `PetalWidthCm`, and `Species`. The output data type is `int64`, indicating that the number of missing values is represented as integers.

```
df.isna().sum()
```

	0
Id	0
SepalLengthCm	0
SepalWidthCm	0
PetalLengthCm	0
PetalWidthCm	0
Species	0

dtype: int64

The code `df.info()` is used to display general information about the DataFrame `df`, including the number of rows (150), number of columns (6), data types of each column, and whether there are any missing values. The result shows that all columns have 150 non-null entries, with data types being `int64` for `Id`, `float64` for numeric columns such as `SepalLengthCm`, `SepalWidthCm`, `PetalLengthCm`, and `PetalWidthCm`, and `object` for the categorical column `Species`. The total memory usage is approximately 7.2+ KB.

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>  
RangeIndex: 150 entries, 0 to 149  
Data columns (total 6 columns):  
#   Column             Non-Null Count  Dtype  
---  ---             -  
0   Id                 150 non-null   int64  
1   SepalLengthCm      150 non-null   float64  
2   SepalWidthCm       150 non-null   float64  
3   PetalLengthCm      150 non-null   float64  
4   PetalWidthCm       150 non-null   float64  
5   Species            150 non-null   object  
dtypes: float64(4), int64(1), object(1)  
memory usage: 7.2+ KB
```


This code is used to **detect and count duplicate rows** across all columns in the DataFrame `df`. The function `df.duplicated()` identifies rows that have identical values in all columns, and `.sum()` calculates the number of such duplicate rows. The result is printed using the `print` statement, showing that there are no duplicates (**Jumlah data yang duplikat = 0**).

```
# Mengecek apakah ada duplicate di seluruh kolom
check_duplicate = df.duplicated().sum()

print(f"Jumlah data yang duplikat = {check_duplicate}")
```

➞ Jumlah data yang duplikat = 0

#digitalskillfair39



Thank You For Attention!

Dibimbing Digital Skill Fair 39

Rasya Haidarulloh

Data Science