day10-pandas-data-cleaning

February 3, 2024

Data Clanning –by Punith V T

[39]:
```
import pandas as pd

data = { "A" : [10,20,30,40,50,None],
       "B" : [None,"Bangaluru","Tumkur","chennai","Mangaluru","Badami"]}

df = pd.DataFrame(data)
df
```

[39]:

	A	B
0	10.0	None
1	20.0	Bangaluru
2	30.0	Tumkur
3	40.0	chennai
4	50.0	Mangaluru
5	NaN	Badami

1. Handling Missing Values:

Dropping rows or columns with missing values:

[40]:
```
# filling missing value of A with the mean of the columns
df['A'].fillna(df['A'].mean(), inplace=True)
df
```

```
/tmp/ipykernel_33/3767303133.py:2: FutureWarning: A value is trying to be set on
a copy of a DataFrame or Series through chained assignment using an inplace
method.
The behavior will change in pandas 3.0. This inplace method will never work
because the intermediate object on which we are setting values always behaves as
a copy.

For example, when doing 'df[col].method(value, inplace=True)', try using
'df.method({col: value}, inplace=True)' or df[col] = df[col].method(value)
instead, to perform the operation inplace on the original object.


  df['A'].fillna(df['A'].mean(), inplace=True)
```

[40]:
```
      A          B
0  10.0       None
1  20.0  Bangaluru
2  30.0     Tumkur
3  40.0    chennai
4  50.0  Mangaluru
5  30.0     Badami
```

[44]:
```
cleanDF = df.dropna()
cleanDF
```

[44]:
```
      A          B
1  20.0  Bangaluru
2  30.0     Tumkur
3  40.0    chennai
4  50.0  Mangaluru
5  30.0     Badami
```

This code will create a new DataFrame by removing columns with any missing values

[45]:
```
cleandf = cleanDF.dropna(axis=1)
cleandf
```

[45]:
```
      A          B
1  20.0  Bangaluru
2  30.0     Tumkur
3  40.0    chennai
4  50.0  Mangaluru
5  30.0     Badami
```

[46]:
```
cleandf = df.dropna(axis=1)
print(cleandf)
```

```
      A
0  10.0
1  20.0
2  30.0
3  40.0
4  50.0
5  30.0
```

2. Removing Duplicates:

Removing duplicate rows

[47]:
```
x = df.drop_duplicates()
x
```

[47]:

	A	B
0	10.0	None
1	20.0	Bangaluru
2	30.0	Tumkur
3	40.0	chennai
4	50.0	Mangaluru
5	30.0	Badami

[57]:
```
# Sample data
data = {'A' : [10,20,30,40,50]}
df = pd.DataFrame(data)

df
```

[57]:

	A
0	10
1	20
2	30
3	40
4	50

3. Data Type Conversion:

Converting data types:

[58]:
```
df["A"]= df["A"].astype(int)
df
```

[58]:

	A
0	10
1	20
2	30
3	40
4	50

4.String Cleaning:

[66]:
```
data = {'A': [1, 2, 3, 4, 5],
        'B': [' apple ', 'banana', 'cherry ', 'date', ' elderberry ']}
df = pd.DataFrame(data)
df
```

[66]:

	A	B
0	1	apple
1	2	banana
2	3	cherry
3	4	date
4	5	elderberry

the str.strip() method to remove leading and trailing whitespaces

[68]:
```
df["B"]= df["B"].str.strip()
df
```

[68]:

	A	B
0	1	apple
1	2	banana
2	3	cherry
3	4	date
4	5	elderberry

[71]:
```
# Convert 'B' column to uppercase
df["B"] = df["B"].str.upper()
df
```

[71]:

	A	B
0	1	APPLE
1	2	BANANA
2	3	CHERRY
3	4	DATE
4	5	ELDERBERRY

6. Removing Irrelevant Columns:

[73]:
```
data = {'A': [1, 2, 3, 4, 5],
        'B': ['apple', 'banana', 'cherry', 'date', 'chocolate'],
        'C': [10, 20, 30, 40, 50]}
df = pd.DataFrame(data)
```

[81]:
```
#removing C column from dataframe

df.drop("C", axis=1, inplace=True)
df
```

[81]:

	A	B
0	1	apple
1	2	banana
2	3	cherry
3	4	date
4	5	chocolate

Data transformation

[88]:
```
data = {'A': [10, 20, 30, 40, 50]}
df = pd.DataFrame(data)
```

[89]:
```
def double(x):
    return x+x

df["A+A"] = df["A"].apply(double)
df
```

[89]:

	A	A+A
0	10	20
1	20	40
2	30	60
3	40	80
4	50	100

[92]:
```
# map()
data = {'Category': ['A', 'B', 'A', 'C', 'B']}
df = pd.DataFrame(data)

category_mapping = {'A': 1, 'B': 2, 'C': 3}
df
```

[92]:

	Category
0	A
1	B
2	A
3	C
4	B

[94]:
```
df["Category_Num"] =df["Category"].map(category_mapping)
df
```

[94]:

	Category	Category_Num
0	A	1
1	B	2
2	A	1
3	C	3
4	B	2

[99]:
```
# applymap()
data = {'A': [1, 2, 3],
        'B': [4, 5, 6]}
df = pd.DataFrame(data)

def square(x):
    return x ** 2

df = df.applymap(square) #we can also use map()
print(df)
```

```
   A   B
0  1  16
1  4  25
2  9  36
```

```
/tmp/ipykernel_33/2438595464.py:9: FutureWarning: DataFrame.applymap has been
deprecated. Use DataFrame.map instead.
  df = df.applymap(square) #we can also use map()
```

[ ]: