Data Pre-processing case study

Pandas - Cleaning Empty Cells

Empty Cells

Empty cells can potentially give you a wrong result when you analyze data.

Remove Rows

One way to deal with empty cells is to remove rows that contain empty cells.

This is usually OK, since data sets can be very big, and removing a few rows will not have a big impact on the result.

Example

Return a new Data Frame with no empty cells:

```
import pandas as pd
df = pd.read_csv('data.csv')
new_df = df.dropna()
print(new_df.to_string())
```

Data Pre-processing case study

If you want to change the original DataFrame, use the inplace = True argument:
Example
Remove all rows with NULL values:

```
import pandas as pd
df = pd.read_csv('data.csv')
df.dropna(inplace = True)
print(df.to_string())
```

Data Pre-processing case study

Replace Empty Values
Another way of dealing with empty cells is to insert a *new* value instead.
This way you do not have to delete entire rows just because of some empty cells.
The fillna() method allows us to replace empty cells with a value:
Example
Replace NULL values with the number 130:

```
import pandas as pd
df = pd.read_csv('data.csv')
df.fillna(130, inplace = True)
print(df.to_string())
```

Data Pre-processing case study

Replace Using Mean, Median, or Mode
A common way to replace empty cells, is to calculate the mean, median or mode value of the column.
Pandas uses the mean() median() and mode() methods to calculate the respective values for a specified column:
Example
Calculate the MEAN, and replace any empty values with it:

```
import pandas as pd
df = pd.read_csv('data.csv')
x = df["Calories"].mean()
df["Calories"].fillna(x, inplace = True)
print(df.to_string())
```

Data Pre-processing case study

Removing Rows

Another way of handling wrong data is to remove the rows that contains wrong data.

This way you do not have to find out what to replace them with, and there is a good chance you do not need them to do your analyses.

Example

Delete rows where "Duration" is higher than 120:

```
import pandas as pd
df = pd.read_csv('data.csv')
for x in df.index:
  if df.loc[x, "Duration"] > 120:
    df.drop(x, inplace = True)
print(df.to_string())
```

Data Pre-processing case study

To discover duplicates, we can use the duplicated() method.
The duplicated() method returns a Boolean values for each row:
Example
Returns True for every row that is a duplicate, othwerwise False:

```
import pandas as pd
df = pd.read_csv('data.csv')
print(df.duplicated())
```