Course
CS-E4002
Course materials
Your points
Code Vault

This course has already ended.

« Erroneous Data Course materials Filtering Data »
CS-E4002 / Structure and Cleaning / Preparing Data for Analysis

Preparing Data for Analysis

How to correct the issues previously mentioned: cleaning the dataset.

1. Correcting encodings:

One way to correct encoding issues is to use regular expressions to change the format of the values that don't fit the desired conventions, one column at a time. Panda's string methods can be very useful to make those kind of corrections. Here is a full inventory of these functions.

```
#To replace all the spaces by underscores:
df.columns = df.columns.str.replace(" ", "_")

#To have only lower cases:
df.columns.str.lower()
```

These same operations can be performed on subsets of DataFrames, including individual columns. Selecting a single column of a DataFrame can be done by inserting the name of the column (as a string) in square brackets after the DataFrame. This gives us a Pandas Series, which is a data type similar to a DataFrame, but it is a strictly one-dimensional series of key-value pairs.

```
#To replace all the spaces by underscores in column 'b':
df['b'] = df['b'].str.replace(" ", "_")
```

The applymap() method allows to apply a function to all the elements of a DataFrame.

```
#To square all elements in the DataFrame:
df.applymap(np.square)
```

2. Removing duplicates:

Duplicates can be removed by using the drop_duplicates() method on the dataframe with the right parameters.

```
#To drop the lines that have the same value in the column name:
df = df.drop_duplicates(subset="name")
```

3. Converting data to the right Python data type:

Once everything is fairly clean, pandas has a convert_dtype() method, which can automatically recognise the type of the data contained in a column and cast it to the proper python datatype.

```
#To first check the types:
>>> df.dtypes
a      int32
b     object
c     object
d    float64
dtype: object

#To convert and get the best types possible:
new_df = df.convert_dtypes()

#To check the types afterwards:
>>> dfn.dtypes
a      Int32
b     string
c    boolean
d      Int64
dtype: object
```

Alternatively, if we want to convert specifically to a numeric data type or to datetime, we can use the functions pd.to_numeric() or pd.to_datetime(). For instance, if we want to do this to a specific column of a DataFrame, we can run the given function with the column (as a Pandas Series) as a parameter, and then assign the result back to the column.

```
df['a'] = pd.to_numeric(df['a'])
```

If some of the values in the column cannot be converted, we can use the parameter **errors='coerce'** to set these values as **NaN**.

```
df['a'] = pd.to_numeric(df['a'], errors='coerce')
```

« Erroneous Data Course materials Filtering Data »