Data Science - A practical Approach

Lorenz Feyen

Sep 24, 2021

CONTENTS

this is a foreword

pdf version can be found here here.

Part I

1. Introduction

CHAPTER ONE

INTRODUCTION

this is an introduction

Part II

2. Data Preparation

CHAPTER

TWO

DATA PREPARATION

this is an introduction

CHAPTER THREE

INDEXING AND SLICING

```
import pandas as pd
```

```
min_temp_df = pd.read_csv('https://raw.githubusercontent.com/jbrownlee/Datasets/
↪master/daily-min-temperatures.csv')
min_temp_df
```

```
            Date  Temp
0     1981-01-01  20.7
1     1981-01-02  17.9
2     1981-01-03  18.8
3     1981-01-04  14.6
4     1981-01-05  15.8
...          ...   ...
3645  1990-12-27  14.0
3646  1990-12-28  13.6
3647  1990-12-29  13.5
3648  1990-12-30  15.7
3649  1990-12-31  13.0

[3650 rows x 2 columns]
```

```
min_temp_df.Date = pd.to_datetime(min_temp_df.Date)
```

```
min_temp_df = min_temp_df.set_index('Date')
```

```
min_temp_df.loc['1989-06-01':'1989-06-30']
```

```
            Temp
Date
1989-06-01   2.3
1989-06-02   1.4
1989-06-03   2.1
1989-06-04   6.6
1989-06-05   8.9
1989-06-06   7.8
1989-06-07   9.0
1989-06-08  10.3
1989-06-09   7.9
1989-06-10   7.2
1989-06-11   8.6
1989-06-12   8.8
```

(continues on next page)

(continued from previous page)

```
1989-06-13   6.2
1989-06-14   9.5
1989-06-15  10.2
1989-06-16   9.7
1989-06-17  11.2
1989-06-18  10.2
1989-06-19  10.1
1989-06-20   8.1
1989-06-21   6.6
1989-06-22   5.0
1989-06-23   4.7
1989-06-24   5.3
1989-06-25   4.5
1989-06-26   2.3
1989-06-27   1.4
1989-06-28   0.5
1989-06-29   2.4
1989-06-30   8.0
```

```
min_temp_df.loc['1989-06-01':'1989-06-30'].mean()
```

```
Temp    6.56
dtype: float64
```

```
import seaborn as sns
```

```
tip_df = sns.load_dataset('tips')
tip_df.head()
```

```
   total_bill   tip     sex smoker  day    time  size
0       16.99  1.01  Female     No  Sun  Dinner     2
1       10.34  1.66    Male     No  Sun  Dinner     3
2       21.01  3.50    Male     No  Sun  Dinner     3
3       23.68  3.31    Male     No  Sun  Dinner     2
4       24.59  3.61  Female     No  Sun  Dinner     4
```

```
tip_index_df = tip_df.set_index('day')
```

```
tip_index_df.loc['Sun']
```

```
     total_bill   tip     sex smoker    time  size
day
Sun       16.99  1.01  Female     No  Dinner     2
Sun       10.34  1.66    Male     No  Dinner     3
Sun       21.01  3.50    Male     No  Dinner     3
Sun       23.68  3.31    Male     No  Dinner     2
Sun       24.59  3.61  Female     No  Dinner     4
..          ...   ...     ...    ...     ...   ...
Sun       20.90  3.50  Female    Yes  Dinner     3
Sun       30.46  2.00    Male    Yes  Dinner     5
Sun       18.15  3.50  Female    Yes  Dinner     3
Sun       23.10  4.00    Male    Yes  Dinner     3
Sun       15.69  1.50    Male    Yes  Dinner     2
```

(continues on next page)

(continued from previous page)

```
[76 rows x 6 columns]
```

```
tip_index_df = tip_df.set_index(['day','time'])
```

```
tip_index_df.loc[('Thur','Lunch')].tip.mean()
```

```
/tmp/ipykernel_36874/2537502835.py:1: PerformanceWarning: indexing past lexsort depth␣
↪may impact performance.
  tip_index_df.loc[('Thur','Lunch')].tip.mean()
```

```
2.767704918032786
```

```
pd.pivot_table(tip_df, values='total_bill', index='day', columns='time', aggfunc=
↪'median')
```

```
time  Lunch  Dinner
day
Thur  16.00  18.780
Fri   13.42  18.665
Sat     NaN  18.240
Sun     NaN  19.630
```

CHAPTER

FOUR

MISSING DATA

In this notebook we will look at a few datasets where values from columns are missing. It is crucial for data science and machine learning to have a dataset where no values are missing as algorithms are usually not able to handle data with information missing.

For python, we will be using the pandas library to handle our dataset.

```
import pandas as pd
```

4.1 Kamyr digester

The first dataset we will be looking at is taken from a psysical device equiped with numerous sensors, each timepoint (1 hour) these sensors are read out and the data is collected. Let's have a look at the general structure

```
kamyr_df = pd.read_csv('https://openmv.net/file/kamyr-digester.csv')
kamyr_df.head()
```

```
  Observation  Y-Kappa  ChipRate  BF-CMratio  BlowFlow  ChipLevel4   \
0    31-00:00    23.10    16.520     121.717  1177.607      169.805
1    31-01:00    27.60    16.810      79.022  1328.360      341.327
2    31-02:00    23.19    16.709      79.562  1329.407      239.161
3    31-03:00    23.60    16.478      81.011  1334.877      213.527
4    31-04:00    22.90    15.618      93.244  1334.168      243.131

   T-upperExt-2   T-lowerExt-2    UCZAA  WhiteFlow-4    ...   SteamFlow-4   \
0        358.282        329.545  1.443       599.253  ...        67.122
1        351.050        329.067  1.549       537.201  ...        60.012
2        350.022        329.260  1.600       549.611  ...        61.304
3        350.938        331.142  1.604       623.362  ...        68.496
4        351.640        332.709    NaN       638.672  ...        70.022

   Lower-HeatT-3  Upper-HeatT-3   ChipMass-4   WeakLiquorF   BlackFlow-2   \
0        329.432        303.099      175.964      1127.197      1319.039
1        330.823        304.879      163.202       665.975      1297.317
2        329.140        303.383      164.013       677.534      1327.072
3        328.875        302.254      181.487       767.853      1324.461
4        328.352        300.954      183.929       888.448      1343.424

   WeakWashF   SteamHeatF-3   T-Top-Chips-4   SulphidityL-4
0     257.325         54.612         252.077             NaN
1     241.182         46.603         251.406           29.11
2     237.272         51.795         251.335             NaN
```

(continues on next page)

(continued from previous page)

```
3     239.478          54.846         250.312            29.02
4     215.372          54.186         249.916            29.01

[5 rows x 23 columns]
```

Interesting, there seem to be 22 sensor values and 1 timestamp for each record. As mechanical devices are prone to noise and dropouts of sensors we would be foolish to assume no missing values are present.

```
kamyr_df.isna().sum().divide(len(kamyr_df)).round(4)*100
```

```
Observation         0.00
Y-Kappa             0.00
ChipRate            1.33
BF-CMratio          4.65
BlowFlow            4.32
ChipLevel4          0.33
T-upperExt-2        0.33
T-lowerExt-2        0.33
UCZAA               7.97
WhiteFlow-4         0.33
AAWhiteSt-4        46.84
AA-Wood-4           0.33
ChipMoisture-4      0.33
SteamFlow-4         0.33
Lower-HeatT-3       0.33
Upper-HeatT-3       0.33
ChipMass-4          0.33
WeakLiquorF         0.33
BlackFlow-2         0.33
WeakWashF           0.33
SteamHeatF-3        0.33
T-Top-Chips-4       0.33
SulphidityL-4      46.84
dtype: float64
```

As expected, the datapoint 'AAWhiteSt-4' even has 46% of data missing! It seems we only have 300 datapoints and presumably these missing values occur in different records our dataset will be decimated if we just drop all rows with missing values.

```
kamyr_df.shape
```

```
(301, 23)
```

```
kamyr_df.dropna().shape
```

```
(131, 23)
```

As we drop all rows with missing values, we are left with only 131 records. Whilst this might be good enough for some purposes, there are more viable options.

Perhaps we can first remove the column with the most missing values and then drop all remaining

```
kamyr_df.drop(columns=['AAWhiteSt-4 ','SulphidityL-4 ']).dropna().shape
```

```
(263, 21)
```

Significantly better, although we lost the information of 2 sensors we now have a complete dataset with 263 records. For purposes where those 2 sensors are irrelevant this is a viable option, keep in mind that this dataset is still 100% truthful, as we have not imputed any values.

Another option, where we retain all our records would be using the timely nature of our dataset, each record is a measurement with an interval of 1 hour. I have no knowledge of this dataset but one might make the assumption that the interval of 1 hour is taken as the state of the machine does not alter much in 1 hour. Therefore we could do what is called a forward fill, where we fill in the missing values with the same value of the sensor for the previous measurement.

This would solve nearly all nan values as there might be a problem where the first value is missing. This is shown below.

```
kamyr_df.fillna(method='ffill')['SulphidityL-4 ']
```

```
0        NaN
1      29.11
2      29.11
3      29.02
4      29.01
       ...
296    30.43
297    30.29
298    30.47
299    30.47
300    30.46
Name: SulphidityL-4 , Length: 301, dtype: float64
```

Although our dataset is not fully the truth, we can see that little to no changes occur in the sensor and using a forward fill is arguably the most suitable option.

4.2 Travel times

Another dataset from the same source contains a collection of recorded travel times and specific information about the travel itself as e.g.: the day of the week, where they were going, …

```
travel_df = pd.read_csv('https://openmv.net/file/travel-times.csv')
travel_df
```

```
          Date StartTime  DayOfWeek GoingTo  Distance  MaxSpeed  AvgSpeed  \
0     1/6/2012     16:37     Friday    Home     51.29     127.4      78.3
1     1/6/2012     08:20     Friday     GSK     51.63     130.3      81.8
2     1/4/2012     16:17  Wednesday    Home     51.27     127.4      82.0
3     1/4/2012     07:53  Wednesday     GSK     49.17     132.3      74.2
4     1/3/2012     18:57    Tuesday    Home     51.15     136.2      83.4
..         ...       ...        ...     ...       ...       ...       ...
200  7/18/2011     08:09     Monday     GSK     54.52     125.6      49.9
201  7/14/2011     08:03   Thursday     GSK     50.90     123.7      76.2
202  7/13/2011     17:08  Wednesday    Home     51.96     132.6      57.5
203  7/12/2011     17:51    Tuesday    Home     53.28     125.8      61.6
204  7/11/2011     16:56     Monday    Home     51.73     125.0      62.8

     AvgMovingSpeed FuelEconomy  TotalTime  MovingTime Take407All Comments
0              84.8         NaN       39.3        36.3         No      NaN
```

(continues on next page)

(continued from previous page)

```
1              88.9         NaN       37.9        34.9         No      NaN
2              85.8         NaN       37.5        35.9         No      NaN
3              82.9         NaN       39.8        35.6         No      NaN
4              88.1         NaN       36.8        34.8         No      NaN
..              ...         ...        ...         ...        ...      ...
200            82.4        7.89       65.5        39.7         No      NaN
201            95.1        7.89       40.1        32.1        Yes      NaN
202            76.7         NaN       54.2        40.6        Yes      NaN
203            87.6         NaN       51.9        36.5        Yes      NaN
204            92.5         NaN       49.5        33.6        Yes      NaN

[205 rows x 13 columns]
```

we have a total of 205 records and we can already see that the FuelEconomy column seems pretty bad, let's quantify that.

```
travel_df.isna().sum().divide(len(travel_df)).round(4)*100
```

```
Date                  0.00
StartTime             0.00
DayOfWeek             0.00
GoingTo               0.00
Distance              0.00
MaxSpeed              0.00
AvgSpeed              0.00
AvgMovingSpeed        0.00
FuelEconomy           8.29
TotalTime             0.00
MovingTime            0.00
Take407All            0.00
Comments             88.29
dtype: float64
```

In the end, it doesn't seem that bad, but there are comments and nearly none of them are filled in. Which in perspective is understandable. Let's see what the comments look like

```
travel_df[~travel_df.Comments.isna()].Comments
```

```
15                                   Put snow tires on
39                                          Heavy rain
49                                 Huge traffic backup
50      Pumped tires up: check fuel economy improved?
52                                 Backed up at Bronte
54                                 Backed up at Bronte
60                                               Rainy
78                                   Rain, rain, rain
91                                   Rain, rain, rain
92         Accident: backup from Hamilton to 407 ramp
110                                            Raining
132                               Back to school traffic?
133                   Took 407 all the way (to McMaster)
150                               Heavy volume on Derry
156                            Start early to run a batch
158    Accident at 403/highway 6; detour along Dundas
165                                       Detour taken
166                                      Must be Friday
172                               Medium amount of rain
```

(continues on next page)

(continued from previous page)

```
174                                             New tires
182                                  Turn around on Derry
184                                          Empty roads
187                                Police slowdown on 403
189                             Accident blocked 407 exit
Name: Comments, dtype: object
```

As you would expect, these comments are text based. Now imagine we would like to run some Natural Language Processing (NLP) on these, it would be a pain to perform string operations on it when it is riddled with missing values.

Here a simple example where we select all records containing the word 'rain', with no avail.

```
travel_df[travel_df.Comments.str.lower().str.contains('rain')]
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
/tmp/ipykernel_36906/1298831137.py in <module>
----> 1 travel_df[travel_df.Comments.str.lower().str.contains('rain')]

~/git/data-science-practical-approach/venv/lib/python3.8/site-packages/pandas/core/
↪frame.py in __getitem__(self, key)
   3446
   3447         # Do we have a (boolean) 1d indexer?
-> 3448         if com.is_bool_indexer(key):
   3449             return self._getitem_bool_array(key)
   3450

~/git/data-science-practical-approach/venv/lib/python3.8/site-packages/pandas/core/
↪common.py in is_bool_indexer(key)
    137                     # Don't raise on e.g. ["A", "B", np.nan], see
    138                     #  test_loc_getitem_list_of_labels_categoricalindex_with_
↪na
--> 139                     raise ValueError(na_msg)
    140                 return False
    141             return True

ValueError: Cannot mask with non-boolean array containing NA / NaN values
```

The last line of the python error traceback gives us the reason it failed, because there were NaN values present.

Luckily the string variable has more or less it's on 'null' value, being an empty string, this way these operations are still possible, most of the comments will just contain nothing.

```
travel_df.Comments = travel_df.Comments.fillna('')
```

```
travel_df[travel_df.Comments.str.lower().str.contains('rain')]
```

```
           Date StartTime  DayOfWeek GoingTo  Distance  MaxSpeed  AvgSpeed  \
39   11/29/2011     07:23    Tuesday     GSK     51.74     112.2      55.3
60    11/9/2011     16:15  Wednesday    Home     51.28     121.4      65.9
78   10/25/2011     17:24    Tuesday    Home     52.87     123.5      65.1
91   10/12/2011     17:47  Wednesday    Home     51.40     114.4      59.7
110   9/27/2011     07:36    Tuesday     GSK     50.65     128.1      86.3
172    8/9/2011     08:15    Tuesday     GSK     49.08     134.8      60.5

     AvgMovingSpeed FuelEconomy  TotalTime  MovingTime Take407All  \
```

(continues on next page)

(continued from previous page)

```
39             61.0        NaN       56.2        50.9         No
60             71.8       9.35       46.7        42.1         No
78             72.4       8.97       48.7        43.8         No
91             65.8       8.75       51.7        46.9         No
110            88.6       8.31       35.2        34.3        Yes
172            67.2       8.54       48.7        43.8         No

                  Comments
39              Heavy rain
60                   Rainy
78        Rain, rain, rain
91        Rain, rain, rain
110                Raining
172  Medium amount of rain
```

Fixed! now we can use the comments for analysis.

We still have to fix the FuelEconomy, let us take a look at the non NaN values

```
travel_df[~travel_df.FuelEconomy.isna()]
```

```
           Date StartTime  DayOfWeek GoingTo  Distance  MaxSpeed  AvgSpeed  \
6      1/2/2012     17:31     Monday    Home     51.37     123.2      82.9
7      1/2/2012     07:34     Monday     GSK     49.01     128.3      77.5
8    12/23/2011     08:01     Friday     GSK     52.91     130.3      80.9
9    12/22/2011     17:19   Thursday    Home     51.17     122.3      70.6
10   12/22/2011     08:16   Thursday     GSK     49.15     129.4      74.0
..          ...       ...        ...     ...       ...       ...       ...
197   7/20/2011     08:24  Wednesday     GSK     48.50     125.8      75.7
198   7/19/2011     17:17    Tuesday    Home     51.16     126.7      92.2
199   7/19/2011     08:11    Tuesday     GSK     50.96     124.3      82.3
200   7/18/2011     08:09     Monday     GSK     54.52     125.6      49.9
201   7/14/2011     08:03   Thursday     GSK     50.90     123.7      76.2

     AvgMovingSpeed FuelEconomy  TotalTime  MovingTime Take407All Comments
6              87.3           -       37.2        35.3         No
7              85.9           -       37.9        34.3         No
8              88.3        8.89       39.3        36.0         No
9              78.1        8.89       43.5        39.3         No
10             81.4        8.89       39.8        36.2         No
..              ...         ...        ...         ...        ...      ...
197            87.3        7.89       38.5        33.3        Yes
198           102.6        7.89       33.3        29.9        Yes
199            96.4        7.89       37.2        31.7        Yes
200            82.4        7.89       65.5        39.7         No
201            95.1        7.89       40.1        32.1        Yes

[188 rows x 13 columns]
```

It seems that aside NaN values there are also other intruders, a quick check on the data type (Dtype) reveils it is not recognised as a number!

```
travel_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 205 entries, 0 to 204
```

(continues on next page)

(continued from previous page)

```
Data columns (total 13 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   Date            205 non-null    object
 1   StartTime       205 non-null    object
 2   DayOfWeek       205 non-null    object
 3   GoingTo         205 non-null    object
 4   Distance        205 non-null    float64
 5   MaxSpeed        205 non-null    float64
 6   AvgSpeed        205 non-null    float64
 7   AvgMovingSpeed  205 non-null    float64
 8   FuelEconomy     188 non-null    object
 9   TotalTime       205 non-null    float64
 10  MovingTime      205 non-null    float64
 11  Take407All      205 non-null    object
 12  Comments        205 non-null    object
dtypes: float64(6), object(7)
memory usage: 20.9+ KB
```

The column is noted as an object or string type, meaning that these numbers are given as '9.24' instead of 9.24 and numerical operations are not possible. We can cast them to numeric but have to warn pandas to coerce errors, meaning errors will be converted to NaN values. Later we'll handle the NaN's.

```
travel_df.FuelEconomy = pd.to_numeric(travel_df.FuelEconomy, errors='coerce')
travel_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 205 entries, 0 to 204
Data columns (total 13 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   Date            205 non-null    object
 1   StartTime       205 non-null    object
 2   DayOfWeek       205 non-null    object
 3   GoingTo         205 non-null    object
 4   Distance        205 non-null    float64
 5   MaxSpeed        205 non-null    float64
 6   AvgSpeed        205 non-null    float64
 7   AvgMovingSpeed  205 non-null    float64
 8   FuelEconomy     186 non-null    float64
 9   TotalTime       205 non-null    float64
 10  MovingTime      205 non-null    float64
 11  Take407All      205 non-null    object
 12  Comments        205 non-null    object
dtypes: float64(7), object(6)
memory usage: 20.9+ KB
```

Wonderful, now the column is numerical and we can see 2 more missing values have popped up! We could easily drop these 19 records and have a complete dataset.

```
travel_df.dropna()
```

```
           Date StartTime  DayOfWeek GoingTo  Distance  MaxSpeed  AvgSpeed  \
8    12/23/2011     08:01     Friday     GSK     52.91     130.3      80.9
9    12/22/2011     17:19   Thursday    Home     51.17     122.3      70.6
10   12/22/2011     08:16   Thursday     GSK     49.15     129.4      74.0
```

(continues on next page)

(continued from previous page)

```
11   12/21/2011     07:45  Wednesday     GSK     51.77     124.8      71.7
12   12/20/2011     16:05    Tuesday    Home     51.45     130.1      75.2
..          ...       ...        ...     ...       ...       ...       ...
197   7/20/2011     08:24  Wednesday     GSK     48.50     125.8      75.7
198   7/19/2011     17:17    Tuesday    Home     51.16     126.7      92.2
199   7/19/2011     08:11    Tuesday     GSK     50.96     124.3      82.3
200   7/18/2011     08:09     Monday     GSK     54.52     125.6      49.9
201   7/14/2011     08:03   Thursday     GSK     50.90     123.7      76.2

     AvgMovingSpeed  FuelEconomy  TotalTime  MovingTime Take407All Comments
8              88.3         8.89       39.3        36.0         No
9              78.1         8.89       43.5        39.3         No
10             81.4         8.89       39.8        36.2         No
11             78.9         8.89       43.3        39.4         No
12             82.7         8.89       41.1        37.3         No
..              ...          ...        ...         ...        ...      ...
197            87.3         7.89       38.5        33.3        Yes
198           102.6         7.89       33.3        29.9        Yes
199            96.4         7.89       37.2        31.7        Yes
200            82.4         7.89       65.5        39.7         No
201            95.1         7.89       40.1        32.1        Yes

[186 rows x 13 columns]
```

However im leaving them as an excercise for you to apply a technique we will see in the next part

4.3 Material properties

Another dataset from the same source contains the material properties from 30 samples, this time there is not timestamp as the samples are not related in time with each other.

```
material_df = pd.read_csv('http://openmv.net/file/raw-material-properties.csv')
material_df
```

```
    Sample  size1  size2  size3  density1  density2  density3
0   X12558  0.696   2.69   6.38      41.8     17.18      3.90
1   X14728  0.636   2.30   5.14      38.1     12.73      3.89
2   X15468  0.841   2.85   5.20      37.6     13.58      3.98
3   X21364  0.609   2.13   4.62      34.2     11.12      4.02
4   X23671  0.684   2.16   4.87      36.4     12.24      3.92
5   X24055  0.762   2.81   6.36      38.1     13.28      3.89
6   X24905  0.552   2.34   5.03      41.3     16.71      3.86
7   X25917  0.501   2.17   5.09       NaN       NaN       NaN
8   X27871  0.619   2.11   5.13       NaN       NaN       NaN
9   X28690  0.610   2.10   4.18      35.0     12.15      3.86
10  X31385  0.532   2.09   4.93       NaN       NaN       NaN
11  X31813  0.738   2.29   5.47       NaN       NaN       NaN
12  X32807  0.779   2.62   5.59       NaN       NaN       NaN
13  X33943  0.537   2.23   5.41      35.2     11.34      3.99
14  X35035  0.702   2.05   5.10      34.2     10.54      4.02
15  X39223  0.768   2.51   5.09      34.9     12.55      3.90
16  X40503  0.714   2.56   6.03      35.6     12.20      4.02
17  X41400  0.621   2.42   5.10      38.7     14.27      3.98
18  X42988  0.726   2.11   4.69      37.1     13.14      3.98
```

(continues on next page)

(continued from previous page)

```
19  X44749  0.698   2.36   5.40       36.6      12.16       4.01
20  X45295    NaN    NaN    NaN       38.1      13.34       3.89
21  X46965  0.759   2.47   4.83       38.7      14.83       3.89
22  X49666  0.535   2.13   5.23        NaN        NaN        NaN
23  X50678  0.716   2.29   5.45       37.3      13.70       3.92
24  X52894  0.635   2.08   4.94        NaN        NaN        NaN
25  X53925  0.598   2.12   4.69       37.9      13.45       3.78
26  X54254  0.700   2.47   5.22       38.8      14.72       3.92
27  X54272  0.957   2.96   7.37       36.2      13.38       4.20
28  X54394  0.759   2.66   5.36       35.2      12.19       3.98
29  X55408  0.661   2.10   4.27        NaN        NaN        NaN
30  X56952  0.646   2.38   4.51       40.1      15.68       3.86
31  X57095  0.662   2.34   4.71       35.0      12.37       3.90
32  X57128  0.749   2.43   5.16       37.3      13.04       3.92
33  X61870  0.598   2.21   4.90        NaN        NaN        NaN
34  X61888  0.619   2.59   5.81        NaN        NaN        NaN
35  X72736  0.693   2.05   5.02       39.6      15.55       3.94
```

let us quantify the amount of missing data

```
material_df.isna().sum().divide(len(material_df)).round(4)*100
```

```
Sample       0.00
size1        2.78
size2        2.78
size3        2.78
density1    27.78
density2    27.78
density3    27.78
dtype: float64
```

Unfortunately that is a lot of missing data, covered in all records, dropping here seems almost impossible if we want to keep a healthy amount of records.

Here it would be wise to go for a more elaborate method of imputation, I opted for the K-nearest neighbours method, which looks at the K most similar records in the dataset to make an educated guess on what the missing value could be, this because we can assume that records with similar data are also similar over all the properties (columns).

Im using the sklearn library for this, which has more imputation techniques such as MICE. More info can be found here

```
from sklearn.impute import KNNImputer
```

im creating an imputer object and specify that i want to use the 5 most similar records and weigh them by distance from the to imputed record, meaning closer neighbours are more important.

```
imputer = KNNImputer(n_neighbors=5, weights="distance")
```

As the imputer only takes numerical values I had to do some pandas magic and drop the first column, which I then added again. The result is a fully filled dataset, you can recognise the new values as they are not rounded.

```
pd.DataFrame(
    imputer.fit_transform(material_df.drop(columns=['Sample'])),
    columns=material_df.columns.drop('Sample')
)
```

```
       size1     size2     size3   density1   density2  density3
0   0.696000  2.690000  6.380000  41.800000  17.180000  3.900000
1   0.636000  2.300000  5.140000  38.100000  12.730000  3.890000
2   0.841000  2.850000  5.200000  37.600000  13.580000  3.980000
3   0.609000  2.130000  4.620000  34.200000  11.120000  4.020000
4   0.684000  2.160000  4.870000  36.400000  12.240000  3.920000
5   0.762000  2.810000  6.360000  38.100000  13.280000  3.890000
6   0.552000  2.340000  5.030000  41.300000  16.710000  3.860000
7   0.501000  2.170000  5.090000  38.495282  14.029399  3.931180
8   0.619000  2.110000  5.130000  37.405275  13.157346  3.943667
9   0.610000  2.100000  4.180000  35.000000  12.150000  3.860000
10  0.532000  2.090000  4.930000  37.811132  13.646072  3.908364
11  0.738000  2.290000  5.470000  37.088833  13.255412  3.941654
12  0.779000  2.620000  5.590000  36.540567  12.889902  3.970973
13  0.537000  2.230000  5.410000  35.200000  11.340000  3.990000
14  0.702000  2.050000  5.100000  34.200000  10.540000  4.020000
15  0.768000  2.510000  5.090000  34.900000  12.550000  3.900000
16  0.714000  2.560000  6.030000  35.600000  12.200000  4.020000
17  0.621000  2.420000  5.100000  38.700000  14.270000  3.980000
18  0.726000  2.110000  4.690000  37.100000  13.140000  3.980000
19  0.698000  2.360000  5.400000  36.600000  12.160000  4.010000
20  0.733097  2.653959  5.881504  38.100000  13.340000  3.890000
21  0.759000  2.470000  4.830000  38.700000  14.830000  3.890000
22  0.535000  2.130000  5.230000  37.391815  13.089536  3.944335
23  0.716000  2.290000  5.450000  37.300000  13.700000  3.920000
24  0.635000  2.080000  4.940000  37.254724  13.206262  3.933904
25  0.598000  2.120000  4.690000  37.900000  13.450000  3.780000
26  0.700000  2.470000  5.220000  38.800000  14.720000  3.920000
27  0.957000  2.960000  7.370000  36.200000  13.380000  4.200000
28  0.759000  2.660000  5.360000  35.200000  12.190000  3.980000
29  0.661000  2.100000  4.270000  36.172345  12.755632  3.887375
30  0.646000  2.380000  4.510000  40.100000  15.680000  3.860000
31  0.662000  2.340000  4.710000  35.000000  12.370000  3.900000
32  0.749000  2.430000  5.160000  37.300000  13.040000  3.920000
33  0.598000  2.210000  4.900000  37.865882  13.826029  3.887021
34  0.619000  2.590000  5.810000  35.932339  12.318210  3.989911
35  0.693000  2.050000  5.020000  39.600000  15.550000  3.940000
```

This concludes the part of missing values, perhaps you can try yourself and impute the missing values for the FuelEconomy using the SimpleImputer or even the IterativeImputer.

CHAPTER FIVE

CONCATENATION AND DEDUPLICATION

https://s3.amazonaws.com/nyc-tlc/trip+data/yellow_tripdata_2020-01.csv

```
import pandas as pd
```

```
---------------------------------------------------------------------------
ModuleNotFoundError                       Traceback (most recent call last)
/tmp/ipykernel_13304/4080736814.py in <module>
----> 1 import pandas as pd

ModuleNotFoundError: No module named 'pandas'
```

```
df = pd.read_csv('https://s3.amazonaws.com/nyc-tlc/trip+data/yellow_tripdata_2020-01.
↪csv')
```

```
/home/lorenzf/.local/lib/python3.8/site-packages/IPython/core/interactiveshell.
↪py:3441: DtypeWarning: Columns (6) have mixed types.Specify dtype option on import␣
↪or set low_memory=False.
  exec(code_obj, self.user_global_ns, self.user_ns)
```

```
df
```

```
         VendorID tpep_pickup_datetime tpep_dropoff_datetime  passenger_count  \
0             1.0  2020-01-01 00:28:15   2020-01-01 00:33:03              1.0
1             1.0  2020-01-01 00:35:39   2020-01-01 00:43:04              1.0
2             1.0  2020-01-01 00:47:41   2020-01-01 00:53:52              1.0
3             1.0  2020-01-01 00:55:23   2020-01-01 01:00:14              1.0
4             2.0  2020-01-01 00:01:58   2020-01-01 00:04:16              1.0
...           ...                  ...                   ...              ...
6405003       NaN  2020-01-31 22:51:00   2020-01-31 23:22:00              NaN
6405004       NaN  2020-01-31 22:10:00   2020-01-31 23:26:00              NaN
6405005       NaN  2020-01-31 22:50:07   2020-01-31 23:17:57              NaN
6405006       NaN  2020-01-31 22:25:53   2020-01-31 22:48:32              NaN
6405007       NaN  2020-01-31 22:44:00   2020-01-31 23:06:00              NaN

         trip_distance  RatecodeID store_and_fwd_flag  PULocationID  \
0                 1.20         1.0                  N           238
1                 1.20         1.0                  N           239
2                 0.60         1.0                  N           238
3                 0.80         1.0                  N           238
4                 0.00         1.0                  N           193
...                ...         ...                ...           ...
6405003           3.24         NaN                NaN           237
```

(continues on next page)

(continued from previous page)

```
6405004          22.13         NaN              NaN          259
6405005          10.51         NaN              NaN          137
6405006           5.49         NaN              NaN           50
6405007          11.60         NaN              NaN          179

         DOLocationID  payment_type  fare_amount  extra  mta_tax  tip_amount  \
0                 239           1.0         6.00   3.00      0.5        1.47
1                 238           1.0         7.00   3.00      0.5        1.50
2                 238           1.0         6.00   3.00      0.5        1.00
3                 151           1.0         5.50   0.50      0.5        1.36
4                 193           2.0         3.50   0.50      0.5        0.00
...               ...           ...          ...    ...      ...         ...
6405003           234           NaN        17.59   2.75      0.5        0.00
6405004            45           NaN        46.67   2.75      0.5        0.00
6405005           169           NaN        48.85   2.75      0.0        0.00
6405006            42           NaN        27.17   2.75      0.0        0.00
6405007           205           NaN        54.56   2.75      0.5        0.00

         tolls_amount  improvement_surcharge  total_amount  \
0                0.00                    0.3         11.27
1                0.00                    0.3         12.30
2                0.00                    0.3         10.80
3                0.00                    0.3          8.16
4                0.00                    0.3          4.80
...               ...                    ...           ...
6405003          0.00                    0.3         21.14
6405004         12.24                    0.3         62.46
6405005          0.00                    0.3         51.90
6405006          0.00                    0.3         30.22
6405007          0.00                    0.3         58.11

         congestion_surcharge
0                         2.5
1                         2.5
2                         2.5
3                         0.0
4                         0.0
...                       ...
6405003                   0.0
6405004                   0.0
6405005                   0.0
6405006                   0.0
6405007                   0.0

[6405008 rows x 18 columns]
```

CHAPTER

SIX

SOME PRACTICE

Now that you have learned techniques in data preparation, why don't you put them to use in this wonderfully horrifying dataset. Good luck!

```
import os
import json

import pandas as pd
import kaggle
```

```
---------------------------------------------------------------------------
ModuleNotFoundError                       Traceback (most recent call last)
/tmp/ipykernel_36949/2054829274.py in <module>
      3
      4 import pandas as pd
----> 5 import kaggle

ModuleNotFoundError: No module named 'kaggle'
```

```
if not os.path.exists("/root/.kaggle"):
    os.mkdir("/root/.kaggle")

with open('/root/.kaggle/kaggle.json', 'w') as f:
    json.dump(
        {
            "username":"lorenzf",
            "key":"7a44a9e99b27e796177d793a3d85b8cf"
        }
        , f)
```

```
kaggle.api.dataset_download_files(dataset='PromptCloudHQ/us-jobs-on-monstercom', path=
↪'./data', unzip=True)
```

```
df = pd.read_csv('./data/monster_com-job_sample.csv')
```

```
df.head()
```

```
                    country country_code date_added has_expired  \
0  United States of America           US        NaN          No
1  United States of America           US        NaN          No
2  United States of America           US        NaN          No
3  United States of America           US        NaN          No
```

(continues on next page)

(continued from previous page)

```
4  United States of America           US        NaN          No

         job_board                                    job_description  \
0  jobs.monster.com  TeamSoft is seeing an IT Support Specialist to...
1  jobs.monster.com  The Wisconsin State Journal is seeking a flexi...
2  jobs.monster.com  Report this job About the Job DePuy Synthes Co...
3  jobs.monster.com  Why Join Altec? If you're considering a career...
4  jobs.monster.com  Position ID#  76162 # Positions  1 State  CT C...

                                           job_title             job_type  \
0             IT Support Technician Job in Madison   Full Time Employee
1           Business Reporter/Editor Job in Madison            Full Time
2  Johnson & Johnson Family of Companies Job Appl...  Full Time, Employee
3                   Engineer - Quality Job in Dixon            Full Time
4      Shift Supervisor - Part-Time Job in Camphill   Full Time Employee

                                            location  \
0                                 Madison, WI 53702
1                                 Madison, WI 53708
2  DePuy Synthes Companies is a member of Johnson...
3                                        Dixon, CA
4                                     Camphill, PA

                     organization  \
0                             NaN
1         Printing and Publishing
2  Personal and Household Services
3                 Altec Industries
4                           Retail

                                            page_url salary  \
0  http://jobview.monster.com/it-support-technici...    NaN
1  http://jobview.monster.com/business-reporter-e...    NaN
2  http://jobview.monster.com/senior-training-lea...    NaN
3  http://jobview.monster.com/engineer-quality-jo...    NaN
4  http://jobview.monster.com/shift-supervisor-pa...    NaN

                    sector                           uniq_id
0    IT/Software Development  11d599f229a80023d2f40e7c52cd941e
1                        NaN  e4cbb126dabf22159aff90223243ff2a
2                        NaN  839106b353877fa3d896ffb9c1fe01c0
3   Experienced (Non-Manager)  58435fcab804439efdcaa7ecca0fd783
4  Project/Program Management  64d0272dc8496abfd9523a8df63c184c
```

Need some inspiration? perhaps this might work.

Part III

3. Data Preprocessing

CHAPTER SEVEN

DATA PREPROCESSING

this is an introduction

Part IV

4. Data Exploration

CHAPTER

EIGHT

DATA EXPLORATION

this is an introduction

Part V

5. Data Visualisation

CHAPTER

NINE

DATA VISUALISATION

this is an introduction

Part VI

6. Machine Learning

CHAPTER

TEN

MACHINE LEARNING

this is an introduction