Data Science 01: Příprava a čištění dat (Data Preparation / Cleaning)

Získávání dat (Data Acquisition)

Dataset TimeLaps, vzniklý na základě časosběrného monitorování robotických zdicích prací, uložen ve formátu CSV

In [1]:
```
# Instalace potřebných knihoven
#%pip install pandas
#%pip install numpy
```

In [2]:
```
# Import potřebných knihoven
import pandas as pd
import numpy as np
```

In [3]:
```
# Soubor je načten a přiřazen do proměnné ‚df‘
other_path = '../../data/raw/timelaps.csv'
df = pd.read_csv(other_path, header=0)
```

In [4]:
```
# Zobrazení prvních 5 řádků datasetu
print('Prvních 5 řádků datového rámce')
df.head(5)
```

```
Prvních 5 řádků datového rámce
```

Out[4]:

	ID	TYPE	START	CAL	WALL	END
0	1	CORNER	13:52:18	13:52:24	13:52:42	13:52:58
1	2	HALF	13:52:58	13:53:02	13:53:18	13:53:36
2	3	BASIC	13:53:36	13:55:20	13:55:44	13:56:00
3	4	BASIC	13:56:00	13:56:06	13:56:20	13:56:36
4	5	BASIC	13:56:36	13:56:40	13:56:54	13:58:10

Přidání nebo změna názvů sloupců

In [5]:
```
# Tvorba názvů sloupců
headers = ['id', 'type_brick', 'time_start', 'time_verif', 'time_dest', 'time_end']
print('Sloupce\n', headers)
```

```
Sloupce
 ['id', 'type_brick', 'time_start', 'time_verif', 'time_dest', 'time_end']
```

In [6]:
```
# Nahrazení názvů sloupců a následná kontrola datového rámce
df.columns = headers
df.head()
```

Out[6]:

	id	type_brick	time_start	time_verif	time_dest	time_end
0	1	CORNER	13:52:18	13:52:24	13:52:42	13:52:58
1	2	HALF	13:52:58	13:53:02	13:53:18	13:53:36
2	3	BASIC	13:53:36	13:55:20	13:55:44	13:56:00
3	4	BASIC	13:56:00	13:56:06	13:56:20	13:56:36
4	5	BASIC	13:56:36	13:56:40	13:56:54	13:58:10

Analýza chybějících hodnot v datové sadě

In [7]:
```
# Nahrazení symbolu „?“ chybějící hodnotou (NaN)
df = df.replace('?', np.nan)
df.head()
```

Out[7]:

	id	type_brick	time_start	time_verif	time_dest	time_end
0	1	CORNER	13:52:18	13:52:24	13:52:42	13:52:58
1	2	HALF	13:52:58	13:53:02	13:53:18	13:53:36
2	3	BASIC	13:53:36	13:55:20	13:55:44	13:56:00
3	4	BASIC	13:56:00	13:56:06	13:56:20	13:56:36
4	5	BASIC	13:56:36	13:56:40	13:56:54	13:58:10

In [8]:
```
# Logická hodnota ‚True‘ indikuje přítomnost chybějící hodnoty, zatímco ‚False‘ označuje její nep
missing_data = df.isnull()
missing_data.head(5)
```

Out[8]:

	id	type_brick	time_start	time_verif	time_dest	time_end
0	False	False	False	False	False	False
1	False	False	False	False	False	False
2	False	False	False	False	False	False
3	False	False	False	False	False	False
4	False	False	False	False	False	False

In [9]:
```
# Výpočet počtu chybějících hodnot v jednotlivých sloupcích datového rámce
for column in missing_data.columns.values.tolist():
    print(f'{missing_data[column].value_counts()}\n')
```

```
id
False    136
Name: count, dtype: int64

type_brick
False    136
Name: count, dtype: int64

time_start
False    136
Name: count, dtype: int64

time_verif
False    136
Name: count, dtype: int64

time_dest
False    136
Name: count, dtype: int64

time_end
False    136
Name: count, dtype: int64
```

Práce s chybějícími daty

Jak pracovat s chybějícími daty?

1. Odstranění dat:
   a. Odstranění celého řádku
   b. Odstranění celého sloupce
2. Nahrazení dat:
   a. Nahrazení průměrnou hodnotou
   b. Nahrazení nejčastější hodnotou (frekvencí)
   c. Nahrazení na základě jiných funkcí

Výpočty a následné úpravy dat

1. Change "type_brick" data to category values:
   a. BASIC -> 1
   b. CORNER -> 2
   c. HALF -> 3
   d. END -> 4
2. Trasfeer values:
   a. time_start (text: HH:MM:SS) -> time_start(int: SS)
   b. time_verif (text: HH:MM:SS) -> time_verif(int: SS)
   c. time_dest (text: HH:MM:SS) -> time_dest(int: SS)
   d. time_end (text: HH:MM:SS) -> time_end(int: SS)
3. Calculate time cyclus:
   a. time_verif = time_verif - time_start
   b. time_dest = time_dest - time_start

c. time_end = time_end - time_start

d. time_start = 0

In [10]:
```
# Konverze časové hodnoty
def time_convert(x):
    h,m,s = map(int,x.split(':'))
    return (h*60+m)*60+s
```

In [11]:
```
df['time_start_sec'] = df.time_start.apply(time_convert)
df['time_verif_sec'] = df.time_verif.apply(time_convert)
df['time_dest_sec'] = df.time_dest.apply(time_convert)
df['time_end_sec'] = df.time_end.apply(time_convert)
df.head()
```

Out[11]:

	id	type_brick	time_start	time_verif	time_dest	time_end	time_start_sec	time_verif_sec	time_dest_sec
0	1	CORNER	13:52:18	13:52:24	13:52:42	13:52:58	49938	49944	49962
1	2	HALF	13:52:58	13:53:02	13:53:18	13:53:36	49978	49982	49998
2	3	BASIC	13:53:36	13:55:20	13:55:44	13:56:00	50016	50120	50144
3	4	BASIC	13:56:00	13:56:06	13:56:20	13:56:36	50160	50166	50180
4	5	BASIC	13:56:36	13:56:40	13:56:54	13:58:10	50196	50200	50214

Kontrola a úprava formátu dat

In [12]:
```
# Kontrola datového typu
df.dtypes
```

Out[12]:
```
id                int64
type_brick       object
time_start       object
time_verif       object
time_dest        object
time_end         object
time_start_sec    int64
time_verif_sec    int64
time_dest_sec     int64
time_end_sec      int64
dtype: object
```

In [13]:
```
# úprava formátu dat: Categories -> Int
df['type'] = df['type_brick'].replace(['CORNER', 'HALF', 'BASIC', 'END'], [2, 3, 1, 4]).infer_obj
df.head()
```

Out[13]:

	id	type_brick	time_start	time_verif	time_dest	time_end	time_start_sec	time_verif_sec	time_dest_sec
0	1	CORNER	13:52:18	13:52:24	13:52:42	13:52:58	49938	49944	49962
1	2	HALF	13:52:58	13:53:02	13:53:18	13:53:36	49978	49982	49998
2	3	BASIC	13:53:36	13:55:20	13:55:44	13:56:00	50016	50120	50144
3	4	BASIC	13:56:00	13:56:06	13:56:20	13:56:36	50160	50166	50180
4	5	BASIC	13:56:36	13:56:40	13:56:54	13:58:10	50196	50200	50214

Výpočet doby pracovního cyklu

In [14]:
```
df['start_to_verif'] = df['time_verif_sec'] - df['time_start_sec']
df['verif_to_dest'] = df['time_dest_sec'] - df['time_verif_sec']
df['dest_to_end'] = df['time_end_sec'] - df['time_dest_sec']
df['total_time'] = df['time_end_sec'] - df['time_start_sec']
df.head()
```

Out[14]:

	id	type_brick	time_start	time_verif	time_dest	time_end	time_start_sec	time_verif_sec	time_dest_sec
0	1	CORNER	13:52:18	13:52:24	13:52:42	13:52:58	49938	49944	49962
1	2	HALF	13:52:58	13:53:02	13:53:18	13:53:36	49978	49982	49998
2	3	BASIC	13:53:36	13:55:20	13:55:44	13:56:00	50016	50120	50144
3	4	BASIC	13:56:00	13:56:06	13:56:20	13:56:36	50160	50166	50180
4	5	BASIC	13:56:36	13:56:40	13:56:54	13:58:10	50196	50200	50214

Identifikace a odstranění odlehlých hodnot

In [15]:
```
# Nastavení horní a dolní meze pro odlehlé hodnoty
low_limit = 0.10
hi_limit =  0.90

q_low = df['total_time'].quantile(low_limit)
q_hi  = df['total_time'].quantile(hi_limit)

df = df[(df['total_time'] < q_hi) & (df['total_time'] > q_low)]
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 102 entries, 0 to 135
Data columns (total 15 columns):
 #   Column          Non-Null Count  Dtype 
---  ------          --------------  ----- 
 0   id              102 non-null    int64 
 1   type_brick      102 non-null    object
 2   time_start      102 non-null    object
 3   time_verif      102 non-null    object
 4   time_dest       102 non-null    object
 5   time_end        102 non-null    object
 6   time_start_sec  102 non-null    int64 
 7   time_verif_sec  102 non-null    int64 
 8   time_dest_sec   102 non-null    int64 
 9   time_end_sec    102 non-null    int64 
 10  type            102 non-null    int64 
 11  start_to_verif  102 non-null    int64 
 12  verif_to_dest   102 non-null    int64 
 13  dest_to_end     102 non-null    int64 
 14  total_time      102 non-null    int64 
dtypes: int64(10), object(5)
memory usage: 12.8+ KB
```

Export datové sady do formátu CSV

In [16]:
```
df.to_csv("../../data/raw/clean_timelaps.csv", index=False)
```

Read/Save Other Data Formats

Data Formate	Read	Save
csv	`pd.read_csv()`	`df.to_csv()`
json	`pd.read_json()`	`df.to_json()`
excel	`pd.read_excel()`	`df.to_excel()`
hdf	`pd.read_hdf()`	`df.to_hdf()`
sql	`pd.read_sql()`	`df.to_sql()`

Autor / Organizace / Datum

Vjačeslav Usmanov, ČVUT v Praze, Fakulta stavební

Přehled změn

Datum (YYYY-MM-DD)	Verze	Autor změny	Popis změny
2026-01-18	1.1	Vjačeslav Usmanov	added DS_01_Data_Cleaning.ipnyb
2026-02-10	1.2	Vjačeslav Usmanov	changed DS_01_Data_Cleaning.ipnyb