Fáza 2

Michaela Gubovská, Jakub Hajdu

V tejto fáze zealizujeme predspracovanie údajov pre strojové učenie.

In [1]:
```
import pandas as pd
import matplotlib.pyplot as plt
import scipy.stats as stats
import numpy as np
import category_encoders as ce
from datetime import datetime
from sklearn.impute import SimpleImputer, KNNImputer
from copy import deepcopy

filename_p = "data/profiles.csv"
dfp = pd.read_csv(filename_p, sep='\t')

filename_l = "data/labor.csv"
dfl = pd.read_csv(filename_l, sep='\t')
```

Pôvodné neočistené dáta vyzerajú napríklad nasledovne:

In [2]:
```
dfp.head()
```

Out[2]:

	Unnamed: 0	job	address	blood_group	ssn	birthdate	residence	race	nan
0	0	Quarry manager	6269 Kelsey Cove\nJessicahaven, KY 76299	B+	818-49-6074	1925-12-31	30070 Anderson Branch Suite 371\nHoffmanberg, ...	White	Kimber Frazi
1	1	Financial risk analyst	72241 Luna Divide Suite 877\nMelindachester, I...	B+	617-52-9894	03/21/1912, 00:00:00	61551 Williams Shoal Apt. 506\nLunaborough, VA...	Black	Timotl Gutierr M
2	2	Waste management officer	6554 Nicole Lodge\nNorth Tanner, PA 59584	B+	690-88-6942	1960-11-19	77305 Palmer Valleys Suite 424\nJennifertown, ...	Black	Elair Ellic
3	3	Publishing rights manager	847 Taylor Court Apt. 547\nGutierrezfort, SD 1...	O+	767-13-0171	1978/11/14	62021 Schwartz Roads\nJaimeville, SD 64413	Black	Richa Anders
4	4	Medical sales representative	83815 Richard Causeway Suite 275\nLemouth, WV ...	B+	412-57-1910	03/14/1995, 00:00:00	Unit 4872 Box 1158\nDPO AA 51410	white	Jennif Ba

In [3]:
```
dfl.head()
```

Out[3]:

	Unnamed: 0	leukocyty	ssn	name	smoker	hemoglobin	trombocyty	indicator	alt	relationship	wei
0	0	5.90289	513-95-7625	Andrew Jacobs	no	7.54279	5.83096	1.0	17.60670	widowed	8.09

	Unnamed: 0	leukocyty	ssn	name	smoker	hemoglobin	trombocyty	indicator	alt	relationship	wei
1	1	5.56403	025-71-2115	Ian Harrison	Y	7.87747	NaN	1.0	17.78037	single	73.58
2	2	6.24057	824-63-0108	Matthew Williams	no	4.72650	7.83234	1.0	25.25152	single	129.45
3	3	5.48374	157-32-2908	Charles Chavez	no	5.43079	5.36911	1.0	18.32802	divoced	18.61
4	4	6.04784	545-96-1267	Allen Chung MD	yes	8.85943	6.76682	1.0	11.03841	divoced	78.83

Ako prvé vykonáme základné úpravy dát z prvej fázy (EDA). Konkrétne očistíme tabuľky od nepotrebného stĺpca "Unnamed", zjednotíme hodnoty vo vybraných stĺpcoch (yes/no a pod.), upravíme stĺpec "birthdate" na iba rok narodenia a odstránime duplikáty z druhej tabuľky, nakoľko z EDA vieme, že v prvej sa duplikáty nenachádzajú.

In [4]:
```
dfp.drop('Unnamed: 0', axis=1, inplace=True)
dfp['race'] = dfp['race'].str.replace('white','White')
dfp['race'] = dfp['race'].str.replace('black','Black')
dfp['race'] = dfp['race'].str.replace('blsck','Black')
dfp['birthdate'] = (pd.to_datetime(dfp.birthdate)).dt.year


dfl.drop('Unnamed: 0', axis=1, inplace=True)
dfl = dfl.drop_duplicates()
dfl['smoker'] = dfl['smoker'].str.replace('N','no')
dfl['smoker'] = dfl['smoker'].str.replace('Y','yes')
```

Po týchto úpravách už vyzerajú naše dáta takto:

In [5]:
```
dfp.head()
```

Out[5]:

	job	address	blood_group	ssn	birthdate	residence	race	name	sex
0	Quarry manager	6269 Kelsey Cove\nJessicahaven, KY 76299	B+	818-49-6074	1925	30070 Anderson Branch Suite 371\nHoffmanberg, ...	White	Kimberly Frazier	F
1	Financial risk analyst	72241 Luna Divide Suite 877\nMelindachester, I...	B+	617-52-9894	1912	61551 Williams Shoal Apt. 506\nLunaborough, VA...	Black	Timothy Gutierrez MD	M
2	Waste management officer	6554 Nicole Lodge\nNorth Tanner, PA 59584	B+	690-88-6942	1960	77305 Palmer Valleys Suite 424\nJennifertown, ...	Black	Elaine Elliott	F
3	Publishing rights manager	847 Taylor Court Apt. 547\nGutierrezfort, SD 1...	O+	767-13-0171	1978	62021 Schwartz Roads\nJaimeville, SD 64413	Black	Richard Anderson	M
4	Medical sales representative	83815 Richard Causeway Suite 275\nLemouth, WV ...	B+	412-57-1910	1995	Unit 4872 Box 1158\nDPO AA 51410	White	Jennifer Bass	F

In [6]:
```
dfl.head()
```

Out[6]:

	leukocyty	ssn	name	smoker	hemoglobin	trombocyty	indicator	alt	relationship	weight	a
0	5.90289	513-95-7625	Andrew Jacobs	no	7.54279	5.83096	1.0	17.60670	widowed	8.09544	42.9287
1	5.56403	025-71-2115	Ian Harrison	yes	7.87747	NaN	1.0	17.78037	single	73.58725	50.2550
2	6.24057	824-63-0108	Matthew Williams	no	4.72650	7.83234	1.0	25.25152	single	129.45079	25.1594
3	5.48374	157-32-2908	Charles Chavez	no	5.43079	5.36911	1.0	18.32802	divoced	18.61698	45.0209
4	6.04784	545-96-1267	Allen Chung MD	yes	8.85943	6.76682	1.0	11.03841	divoced	78.83355	59.0639

Pre ďalšie úpravy dát si tieto 2 tabuľky spojíme do jednej.

In [7]:
```
df = pd.merge(dfp, dfl, how="left", on=["ssn"])
df.head()
```

Out[7]:

	job	address	blood_group	ssn	birthdate	residence	race	name_x	sex	leukocyty
0	Quarry manager	6269 Kelsey Cove\nJessicahaven, KY 76299	B+	818-49-6074	1925	30070 Anderson Branch Suite 371\nHoffmanberg, ...	White	Kimberly Frazier	F	6.35996
1	Quarry manager	6269 Kelsey Cove\nJessicahaven, KY 76299	B+	818-49-6074	1925	30070 Anderson Branch Suite 371\nHoffmanberg, ...	White	Kimberly Frazier	F	6.11726
2	Quarry manager	6269 Kelsey Cove\nJessicahaven, KY 76299	B+	818-49-6074	1925	30070 Anderson Branch Suite 371\nHoffmanberg, ...	White	Kimberly Frazier	F	6.65582
3	Quarry manager	6269 Kelsey Cove\nJessicahaven, KY 76299	B+	818-49-6074	1925	30070 Anderson Branch Suite 371\nHoffmanberg, ...	White	Kimberly Frazier	F	6.77521
4	Financial risk analyst	72241 Luna Divide Suite 877\nMelindachester, I...	B+	617-52-9894	1912	61551 Williams Shoal Apt. 506\nLunaborough, VA...	Black	Timothy Gutierrez MD	M	5.99290

5 rows × 25 columns

Odstránime stĺpce, ktoré ďalej nebudeme používať - stĺpce ssn, mená, adresy, zamestnanie, a rodinný stav. Tieto stĺpce sú nám pre ďalšiu fázu strojového učenia zbytočné, nakoľko podľa nás nemajú vplyv na indikátor.

Odstránime aj stĺpce "blood_group" a "race", pretože sme zistili, že nemajú vplyv na indikátor, keďže pri každej

skupine je pomer indikátora 0 a 1 približne rovnaký.

In [8]:
```
df.groupby(['blood_group', 'indicator']).size()
```

Out[8]:
```
blood_group  indicator
A+           0.0          469
             1.0          842
A-           0.0          422
             1.0          776
AB+          0.0          447
             1.0          809
AB-          0.0          438
             1.0          782
B+           0.0          438
             1.0          783
B-           0.0          406
             1.0          758
O+           0.0          490
             1.0          839
O-           0.0          413
             1.0          806
dtype: int64
```

In [9]:
```
df.groupby(['race', 'indicator']).size()
```

Out[9]:
```
race      indicator
Asian     0.0           370
          1.0           650
Black     0.0          1018
          1.0          1764
Hawaiian  0.0           177
          1.0           303
Indian    0.0           160
          1.0           317
White     0.0          1798
          1.0          3361
dtype: int64
```

In [10]:
```
df.drop('name_x', axis=1, inplace=True)
df.drop('name_y', axis=1, inplace=True)
df.drop('job', axis=1, inplace=True)
df.drop('address', axis=1, inplace=True)
df.drop('residence', axis=1, inplace=True)
df.drop('relationship', axis=1, inplace=True)
df.drop('blood_group', axis=1, inplace=True)
df.drop('race', axis=1, inplace=True)
df.drop('ssn', axis=1, inplace=True)
df.head()
```

Out[10]:

	birthdate	sex	leukocyty	smoker	hemoglobin	trombocyty	indicator	alt	weight	ast	alp
0	1925	F	6.35996	no	7.05602	5.71857	1.0	16.37812	51.36819	43.29070	70.74271
1	1925	F	6.11726	yes	6.47482	6.54765	1.0	16.43658	29.84194	28.92169	82.95045
2	1925	F	6.65582	no	9.75669	8.89793	1.0	15.94319	122.71392	46.08027	23.47025
3	1925	F	6.77521	no	6.89806	6.73572	1.0	15.15328	81.49556	32.56414	80.44580
4	1912	M	5.99290	yes	8.93612	5.38672	1.0	11.70433	65.37642	35.27976	22.40511

1. Integrácia a čistenie dát

Zakódovanie nečíselných atribútov

Ako prvé si zakódujeme všetky nečíselné hodnoty, nakoľko pre ďalšiu fázu strojového učenia musí byť každé pozorovanie opísané jedným riadkom s iba numerickými hodnotami. Na zakódovanie použijeme One Hot encoder, ktorý pre daný stĺpec zakóduje hodnoty ako 0 alebo 1 (true or false) a vytvorí nové stĺpce, ktorých počet je rovný počtu rôznych kódovaných hodnôt.

In [11]:
```
# create an object of the OneHotEncoder
ce_OHE = ce.OneHotEncoder(cols=['sex', 'smoker'])

# fit and transform and you will get the encoded data
df = ce_OHE.fit_transform(df)
df = df.rename(columns={"sex_1": "sex_f", "sex_2": "sex_m", "smoker_1": "smoker_no", "smok
df.head()
```

Out[11]:

	birthdate	sex_f	sex_m	leukocyty	smoker_no	smoker_yes	hemoglobin	trombocyty	indicator	alt	wei
0	1925	1	0	6.35996	1	0	7.05602	5.71857	1.0	16.37812	51.36
1	1925	1	0	6.11726	0	1	6.47482	6.54765	1.0	16.43658	29.84
2	1925	1	0	6.65582	1	0	9.75669	8.89793	1.0	15.94319	122.71
3	1925	1	0	6.77521	1	0	6.89806	6.73572	1.0	15.15328	81.49
4	1912	0	1	5.99290	0	1	8.93612	5.38672	1.0	11.70433	65.37

Ďalším dôležitým krokom je identifikácia outlierov a chýbajúcich hodnôt. V tejto časti si ukážeme viaceré spôsoby na úpravu vychýlených a chýbajúcich hodnôt. Tento krok je dôležitý pre ďalšiu fázu machine learningu, aby sme mali pre vybraný algoritmus vhodný model.

Chýbajúce hodnoty

Ako prvé sa pozrieme, koľko záznamov v našej tabuľke obsahuje aspoň jednu chýbajúcu (NaN) hodnotu. Toto vieme zistiť pomocou funkcie .dropna(), ktorá vráti dataframe očistený od záznamov obsahujúcich aspoň jednu chýbajúcu hodnotu.

In [12]:
```
len(df) - len(df.dropna())
```

Out[12]:
```
324
```

Vidíme, že celkový počet záznamov s nejakou chýbajucou hodnotou je 324, čo predstavuje približne 3.2% záznamov. Nakoľko je táto hodnota menšia ako 5% (podľa prednášky odporúčaná hranica po ktorú je možné odstrániť takéto záznamy z datasetu), problém s chýbajúcimi hodnotami by sme mohli jednoducho vyriešiť odstránením záznamov s aspoň jednou chýbajúcou hodnotou príkazom `df = df.dropna()` .

Iným postupom je nahradenie chýbajúcich hodnôt. Pozrieme sa, v ktorých stĺpcoch sa nachádzajú chýbajúce hodnoty.

Vidíme, že sa nachádzajú iba v stĺpcoch s číselnými hodnotami, čo nám vyhovuje, nakoľko v ďalších krokoch budeme používať SimpleImputer, ktorý vie pre chýbajúcu hodnotu podľa zvolenej stratégie vypočítať jej novú hodnotu na základe ostatných číselných hodnôt v danom stĺpci.

In [13]:
```
df.isnull().sum()
```

Out[13]:
```
birthdate      0
```

```
sex_f           0
sex_m           0
leukocyty      30
smoker_no       0
smoker_yes      0
hemoglobin     30
trombocyty     30
indicator       0
alt            30
weight          0
ast            30
alp            30
hematokrit     30
hbver          30
etytr          30
er-cv          30
erytrocyty     30
dtype: int64
```

In [14]:
```
nan_columns = df.columns[df.isna().any()].tolist()
nan_columns
```

Out[14]:
```
['leukocyty',
 'hemoglobin',
 'trombocyty',
 'alt',
 'ast',
 'alp',
 'hematokrit',
 'hbver',
 'etytr',
 'er-cv',
 'erytrocyty']
```

Pre neskoršie porovnávanie nahradených hodnôt rôznymi stratégiami si zapamätáme indexy záznamov, ktoré majú v stĺpci "leukocyty" chýbajúcu hodnotu.

In [15]:
```
nan_values_leukocyty = df[(df['leukocyty'].isna() == True)].index.values.astype(int)
nan_values_leukocyty
```

Out[15]:
```
array([ 364,  830, 1097, 1437, 1733, 2224, 2440, 2764, 2998, 3047, 3678,
       3855, 4477, 4615, 4989, 5032, 5121, 5220, 6018, 6148, 6185, 6665,
       6758, 7275, 7508, 7509, 8059, 8325, 8430, 9071])
```

Nahradenie chýbajúcich hodnôt mediánom

In [16]:
```
# imputer
temp_median = deepcopy(df)
imp_median = SimpleImputer(strategy='median', missing_values=np.nan)
imp_median = imp_median.fit(temp_median[nan_columns])
temp_median[nan_columns] = imp_median.transform(temp_median[nan_columns])

print("Medián neupravených hodnôt leukocytov: ", df['leukocyty'].median())
temp_median.loc[nan_values_leukocyty].head()
```

```
Medián neupravených hodnôt leukocytov:  5.96288
```

Out[16]:

	birthdate	sex_f	sex_m	leukocyty	smoker_no	smoker_yes	hemoglobin	trombocyty	indicator	alt	
364	1961	0	1	5.96288	0	1	8.66276	5.42962	1.0	14.19480	14
830	1989	1	0	5.96288	1	0	6.56560	6.41618	0.0	20.10001	3
1097	1908	0	1	5.96288	0	1	8.68357	6.37006	0.0	14.98156	9

	birthdate	sex_f	sex_m	leukocyty	smoker_no	smoker_yes	hemoglobin	trombocyty	indicator	alt	
1437	2001	1	0	5.96288	1	0	5.97156	5.93306	0.0	21.02052	12
1733	1978	1	0	5.96288	0	1	5.77229	7.56224	1.0	16.03531	4

Nahradili sme chýbajúce hodnoty mediánom hodnôt v danom stĺpci. V príklade tabuľky vyššie môžeme vidieť nahradené chýbajúce hodnoty leukocytov hodnotou mediána stĺpca "leukocyty".

Pre kontrolu si môžeme pozrieť, či sa v našej tabuľke už naozaj v žiadnom stĺpci nenachádzajú chýbajúce hodnoty.

In [17]:
```
temp_median.isnull().sum()
```

Out[17]:
```
birthdate      0
sex_f          0
sex_m          0
leukocyty      0
smoker_no      0
smoker_yes     0
hemoglobin     0
trombocyty     0
indicator      0
alt            0
weight         0
ast            0
alp            0
hematokrit     0
hbver          0
etytr          0
er-cv          0
erytrocyty     0
dtype: int64
```

Nahradenie chýbajúcich hodnôt priemerom

In [18]:
```
# imputer
temp_mean = deepcopy(df)
imp_mean = SimpleImputer(strategy='mean', missing_values=np.nan)
imp_mean = imp_mean.fit(temp_mean[nan_columns])
temp_mean[nan_columns] = imp_mean.transform(temp_mean[nan_columns])

print("Priemer neupravených hodnôt leukocytov: ", df['leukocyty'].mean())
temp_mean.loc[nan_values_leukocyty].head()
```

```
Priemer neupravených hodnôt leukocytov:  5.963460102144014
```

Out[18]:

	birthdate	sex_f	sex_m	leukocyty	smoker_no	smoker_yes	hemoglobin	trombocyty	indicator	alt	
364	1961	0	1	5.96346	0	1	8.66276	5.42962	1.0	14.19480	14
830	1989	1	0	5.96346	1	0	6.56560	6.41618	0.0	20.10001	3
1097	1908	0	1	5.96346	0	1	8.68357	6.37006	0.0	14.98156	9
1437	2001	1	0	5.96346	1	0	5.97156	5.93306	0.0	21.02052	12
1733	1978	1	0	5.96346	0	1	5.77229	7.56224	1.0	16.03531	4

Nahradili sme chýbajúce hodnoty priemerom hodnôt v danom stĺpci. V príklade tabuľky vyššie môžeme vidieť nahradené chýbajúce hodnoty leukocytov priemernou hodnotou stĺpca "leukocyty".

Pre kontrolu si môžeme pozrieť, či sa v našej tabuľke už naozaj v žiadnom stĺpci nenachádzajú chýbajúce hodnoty.

In [19]:
```
temp_mean.isnull().sum()
```

Out[19]:
```
birthdate     0
sex_f         0
sex_m         0
leukocyty     0
smoker_no     0
smoker_yes    0
hemoglobin    0
trombocyty    0
indicator     0
alt           0
weight        0
ast           0
alp           0
hematokrit    0
hbver         0
etytr         0
er-cv         0
erytrocyty    0
dtype: int64
```

Nahradenie chýbajúcich hodnôt pomerom ku korelovanému atribútu

Nahradenie chýbajúcich hodnôt pomocou kNN algoritmu

Ako posledné sa pozrieme na stratégiu nahradenia chýbajúcich hodnôt pomocou kNN algoritmu. Pre každú chýbajúcu hodnotu sa zvolí 5 najbližších (najpodobnejších) záznamov s nie-NaN hodnotami v danom stĺpci, spriemeruje ich a túto hodnotu priradí. Túto stratégiu aplikujeme na celý dataframe, nakoľko ju považujeme za najpresnejší spôsob nahradenia chýbajúcej hodnoty, keďže berie pri výpočte novej hodnoty do úvahy podobné záznamy.

In [20]:
```
# imputer
imp_knn = KNNImputer(n_neighbors=5, weights='uniform', metric='nan_euclidean')
imp_knn.fit(df[nan_columns])
df[nan_columns] = imp_knn.transform(df[nan_columns])

df.loc[nan_values_leukocyty].head()
```

Out[20]:

	birthdate	sex_f	sex_m	leukocyty	smoker_no	smoker_yes	hemoglobin	trombocyty	indicator	alt	
364	1961	0	1	5.595214	0	1	8.66276	5.42962	1.0	14.19480	14
830	1989	1	0	6.402176	1	0	6.56560	6.41618	0.0	20.10001	3
1097	1908	0	1	6.142742	0	1	8.68357	6.37006	0.0	14.98156	9
1437	2001	1	0	5.961904	1	0	5.97156	5.93306	0.0	21.02052	12
1733	1978	1	0	5.734782	0	1	5.77229	7.56224	1.0	16.03531	4

Nahradili sme chýbajúce hodnoty hodnotami získanými kNN algoritmom podľa 5 najpodobnejších záznamov. V príklade tabuľky vyššie môžeme vidieť nahradené chýbajúce hodnoty leukocytov. Hodnoty sa líšia, keďže pre každý záznam sa počítajú na základe iných piatich susedov. Pre kontrolu si môžeme pozrieť, či sa v našej tabuľke už naozaj v žiadnom stĺpci nenachádzajú chýbajúce hodnoty.

In [21]:
```
df.isnull().sum()
```

Out[21]:
```
birthdate      0
sex_f          0
sex_m          0
leukocyty      0
smoker_no      0
smoker_yes     0
hemoglobin     0
trombocyty     0
indicator      0
alt            0
weight         0
ast            0
alp            0
hematokrit     0
hbver          0
etytr          0
er-cv          0
erytrocyty     0
dtype: int64
```

In [22]:
```
df.weight.describe()
```

Out[22]:
```
count    9918.000000
mean       70.675972
std        35.164592
min       -50.180020
25%        46.796368
50%        70.870875
75%        94.500495
max       204.381010
Name: weight, dtype: float64
```

Vychýlené hodnoty

In [23]:
```
##funkcia na detekciu outlierov
def identify_outliers(a):
    lower = a.quantile(0.25) - 1.5 * stats.iqr(a)
    upper = a.quantile(0.75) + 1.5 * stats.iqr(a)

    return a[(a > upper) | (a < lower)]
```

Odstránenie vychýlených pozorovaní

Ako prvé si identifikujeme vychýlené hodnoty naprieč stĺpcami v našom dataframe. Tieto nájdené hodnoty následne odstránime.

In [24]:
```
df.head()
```

Out[24]:

	birthdate	sex_f	sex_m	leukocyty	smoker_no	smoker_yes	hemoglobin	trombocyty	indicator	alt	wei
0	1925	1	0	6.35996	1	0	7.05602	5.71857	1.0	16.37812	51.36
1	1925	1	0	6.11726	0	1	6.47482	6.54765	1.0	16.43658	29.84
2	1925	1	0	6.65582	1	0	9.75669	8.89793	1.0	15.94319	122.71
3	1925	1	0	6.77521	1	0	6.89806	6.73572	1.0	15.15328	81.49
4	1912	0	1	5.99290	0	1	8.93612	5.38672	1.0	11.70433	65.37

In [25]:
```
temp_df = deepcopy(df)
```

```
outliers = []
d = {}
for col in nan_columns + ['birthdate', 'weight']:
    x = identify_outliers(temp_df[col]).index.values.astype(int)
    d[col] = x
    for element in x:
        if element not in outliers:
            outliers.append(element)

print(len(outliers))
```

```
734
```

Nakoľko sme zistili, že počet záznamov patriacich medzi outlierov aspoň jedného atribútu je 734, čo pri celkovom počte pozorovaní predstavuje približne 7.4%. Nakoľko sa podľa prednášok neodporúča odstraňovať vychýlené hodnoty, ktorých počet presahuje 5% z celkového počtu záznamov, nepovažujeme za správne odstránenie záznamov obsahujúcich tieto vychýlené hodnoty. V prípade, že by ich počet bol <= 5% všetkých pozorovaní, odstránili by sme ich pomocou `temp_df.drop(outliers, inplace=True)`

Nahradenie vychýlenej hodnoty hraničnými hodnotami rozdelenia

Pre zachovanie počtu záznamov nahradíme vychýlené hodnoty hraničnými hodnotami daného stĺpca. Konkrétne outlierov za maximom nahradíme hodnotou 95. percentilu a outlierov za minimom nahradíme hodnotou 5. percentilu.

In [26]:
```
##funkcia na detekciu outlierov, vrati oddelene zoznamy indexov prilis vysokych a prilis n
def identify_outliers_low_up(a):
    lower = a.quantile(0.25) - 1.5 * stats.iqr(a)
    upper = a.quantile(0.75) + 1.5 * stats.iqr(a)

    return a[(a < lower)].index.values.astype(int), a[(a > upper)].index.values.astype(int
```

In [27]:
```
##nahradenie hodnot outlierov hodnotami 5. a 95. percentilom rozlozenia
for col in nan_columns + ['birthdate', 'weight']:
    #print('\n', col) # nazov stlpca pre kontrolny vypis
    low, up = identify_outliers_low_up(df[col])
    #if (len(low) > 0): print('low[0]: ', df.loc[low[0], col]) ##prvy nizky outlier pre ko
    #if (len(up) > 0): print('up[0]: ', df.loc[up[0], col]) ##prvy vysoky outlier pre konu
    #print('0.05: ', df[col].quantile(0.05)) ##hodnota 5. percentilu, ktorou sa nahradzaju
    #print('0.95: ', df[col].quantile(0.95)) ##hodnota 95. percentilu, ktorou sa nahradzaj
    df.loc[low, col] = df[col].quantile(0.05)
    df.loc[up, col] = df[col].quantile(0.95)
    #if (len(low) > 0): print('low[0]: ', df.loc[low[0], col]) ##prvy nizky outlier po nah
    #if (len(up) > 0): print('up[0]: ', df.loc[up[0], col]) ##prvy vysoky outlier po nahra
```

2. Realizácia predspracovania dát

Naše stribúty dát si teraz pretranformujeme na vhodný formát pre ML. Použijeme slgoritmy MinMaxScaling a Power Transformer.

Rozdelenie na trénovaciu a testovaciu sadu

Ako prvý krok prerozdelíme náš dataset na trénovaciu a testovaciu podmnožinu v pomere 75:25. Sada X obsahuje všetky atribúty, ktorých vplyv na indikátor budeme sledovať. Sada Y obsahuje iba atribút indikátor.

In [28]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(df.loc[:, df.columns != 'indicator'],
X_train.head()
```

Out[28]:

	birthdate	sex_f	sex_m	leukocyty	smoker_no	smoker_yes	hemoglobin	trombocyty	alt	weight	
9062	1955	1	0	4.97532	1	0	9.40978	5.54273	21.59492	28.09623	52
4193	2005	0	1	6.67900	0	1	5.62701	4.83411	16.79028	54.31164	63
9551	2016	1	0	5.49486	1	0	8.00685	5.52883	10.57991	36.64095	41
7644	1924	0	1	5.05813	1	0	7.00242	7.17623	10.35175	127.44237	58
9022	1988	0	1	7.96354	0	1	8.95593	6.95014	13.84913	21.88924	55

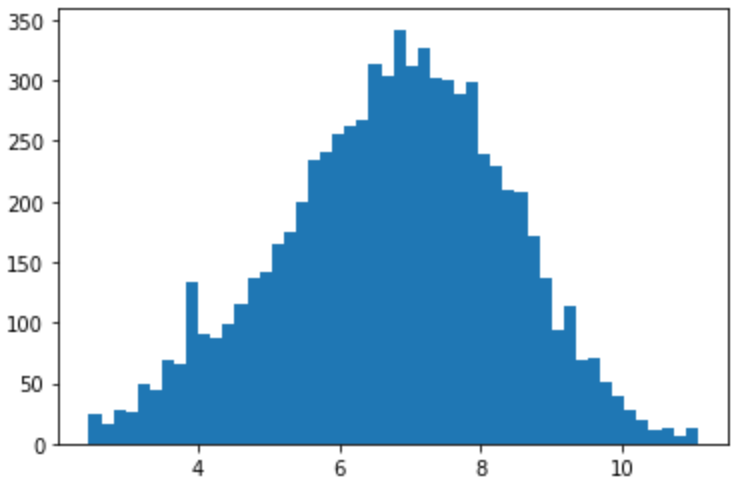
Pre potreby poslednej časti tejto fázy (Pipeline) si spravíme kópiu ešte netransformovanej sady X.

In [29]:
```
X_train_pipe = deepcopy(X_train)
X_test_pipe = deepcopy(X_test)
```

Môžeme sa pozrieť na aktuálne rozloženie hodnôt atribútu hemoglobínu. Vidíme, že rozdelenie nie je normálne a takisto rozsah hodnôt sa nenachádza v rozmedzí 0 až 1.

In [30]:
```
plt.hist(X_train['hemoglobin'], bins=50)
```

Out[30]:
```
(array([ 25.,  17.,  28.,  26.,  49.,  45.,  69.,  65., 133.,  90.,  88.,
         99., 116., 136., 142., 165., 175., 200., 234., 241., 256., 263.,
        267., 313., 304., 342., 312., 326., 302., 301., 289., 298., 239.,
        229., 210., 208., 171., 137.,  94., 114.,  69.,  71.,  51.,  39.,
         27.,  19.,  12.,  13.,   6.,  13.]),
 array([ 2.45362  ,  2.6259208,  2.7982216,  2.9705224,  3.1428232,
         3.315124 ,  3.4874248,  3.6597256,  3.8320264,  4.0043272,
         4.176628 ,  4.3489288,  4.5212296,  4.6935304,  4.8658312,
         5.038132 ,  5.2104328,  5.3827336,  5.5550344,  5.7273352,
         5.899636 ,  6.0719368,  6.2442376,  6.4165384,  6.5888392,
         6.76114  ,  6.9334408,  7.1057416,  7.2780424,  7.4503432,
         7.622644 ,  7.7949448,  7.9672456,  8.1395464,  8.3118472,
         8.484148 ,  8.6564488,  8.8287496,  9.0010504,  9.1733512,
         9.345652 ,  9.5179528,  9.6902536,  9.8625544, 10.0348552,
        10.207156 , 10.3794568, 10.5517576, 10.7240584, 10.8963592,
        11.06866  ]),
 <BarContainer object of 50 artists>)
```



Transformers

Dáta si transformujeme pomocou Power Transformera aby sme dosiahli rozdelenie hodnôt atribútov čo najbližšie normálnemu. Netransformujeme si však atribúty, ktoré nepotrebujeme transformovať - rok narodenia, pohlavie, fajčiar/nefajčiar.

In [31]:
```
from sklearn.compose import ColumnTransformer
from sklearn.preprocessing import PowerTransformer
transform_columns = nan_columns + ['weight']
passthrough_columns = ['birthdate', 'sex_f', 'sex_m', 'smoker_no', 'smoker_yes']

ct = ColumnTransformer(transformers=[("PT", PowerTransformer(method='yeo-johnson', standa
X_train = pd.DataFrame(ct.fit_transform(X_train), columns=transform_columns + passthrough_
X_train.head()
```

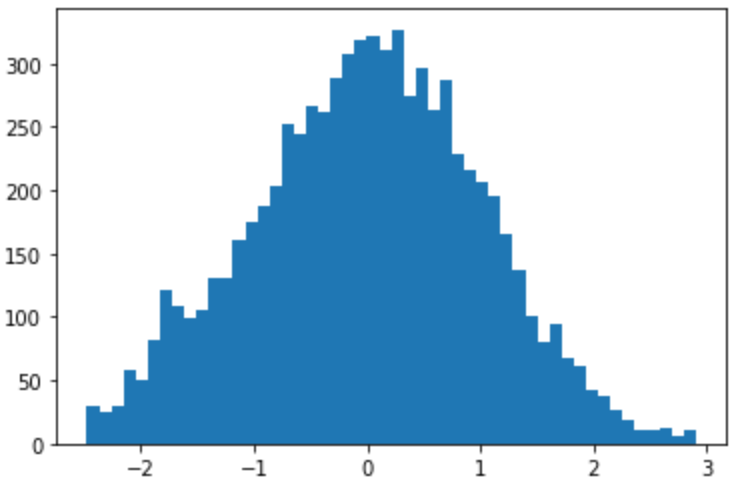
Out[31]:

	leukocyty	hemoglobin	trombocyty	alt	ast	alp	hematokrit	hbver	etytr	er-cv
0	-1.009417	1.726808	-0.477095	1.552396	0.366963	-1.654967	-0.152605	-0.032211	-0.523598	-0.775183
1	0.736221	-0.728332	-1.208990	0.456606	1.283866	0.888232	0.669843	1.292918	-1.282838	-0.223430
2	-0.474324	0.777152	-0.491436	-1.067433	-0.470361	-0.652889	-0.039163	-0.549442	-2.311967	0.197629
3	-0.923951	0.124492	1.204476	-1.126514	0.882733	0.872593	0.560526	-0.694443	0.016614	0.462467
4	2.037109	1.414969	0.972151	-0.246977	0.610174	-1.379821	-0.339389	-1.204800	-1.847407	-1.020153

Keď sa po transformácii pozrieme na hodnoty atribútu hemoglobín, vidíme, že sa rozdelenie viditeľne zmenilo na normálne.

In [32]:
```
plt.hist(X_train['hemoglobin'], bins=50)
```

Out[32]:
```
(array([ 29.,  25.,  29.,  58.,  51.,  82., 121., 108.,  99., 105., 130.,
        131., 160., 175., 187., 204., 252., 245., 267., 262., 289., 307.,
        319., 322., 310., 327., 274., 296., 264., 287., 228., 216., 206.,
        195., 165., 137., 100.,  80.,  94.,  67.,  62.,  43.,  37.,  26.,
         18.,  10.,  11.,  12.,   6.,  10.]),
 array([-2.47602157e+00, -2.36846588e+00, -2.26091019e+00, -2.15335450e+00,
        -2.04579881e+00, -1.93824312e+00, -1.83068743e+00, -1.72313173e+00,
        -1.61557604e+00, -1.50802035e+00, -1.40046466e+00, -1.29290897e+00,
        -1.18535328e+00, -1.07779759e+00, -9.70241897e-01, -8.62686206e-01,
        -7.55130515e-01, -6.47574824e-01, -5.40019133e-01, -4.32463441e-01,
        -3.24907750e-01, -2.17352059e-01, -1.09796368e-01, -2.24067709e-03,
         1.05315014e-01,  2.12870705e-01,  3.20426396e-01,  4.27982087e-01,
         5.35537778e-01,  6.43093470e-01,  7.50649161e-01,  8.58204852e-01,
         9.65760543e-01,  1.07331623e+00,  1.18087192e+00,  1.28842762e+00,
         1.39598331e+00,  1.50353900e+00,  1.61109469e+00,  1.71865038e+00,
         1.82620607e+00,  1.93376176e+00,  2.04131745e+00,  2.14887314e+00,
         2.25642884e+00,  2.36398453e+00,  2.47154022e+00,  2.57909591e+00,
         2.68665160e+00,  2.79420729e+00,  2.90176298e+00]),
 <BarContainer object of 50 artists>)
```



Scaling

Ďalšiu techniku, ktorú aplikujeme je MinMaxScaling. Táto technika nám zabezpečí transformáciu hodnôt atribútov do rozmedzia od 0 po 1.

In [33]:
```
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler()

X_train = pd.DataFrame(scaler.fit_transform(X_train), columns=X_train.columns)
X_train.head()
```

Out[33]:

	leukocyty	hemoglobin	trombocyty	alt	ast	alp	hematokrit	hbver	etytr	er-cv	eryt
0	0.316511	0.781517	0.417987	0.824124	0.577061	0.107337	0.489452	0.458152	0.400023	0.365514	0.4
1	0.639761	0.324983	0.284371	0.625973	0.741888	0.768415	0.632758	0.680407	0.259519	0.464994	0.5
2	0.415597	0.604928	0.415369	0.350382	0.426540	0.367817	0.509218	0.371400	0.069071	0.540910	0.4
3	0.332337	0.483566	0.724977	0.339698	0.669778	0.764350	0.613710	0.347080	0.499993	0.588659	0.4
4	0.880653	0.723530	0.682564	0.498744	0.620782	0.178858	0.456906	0.261482	0.155041	0.321346	0.4

In [34]:
```
X_train.describe()
```

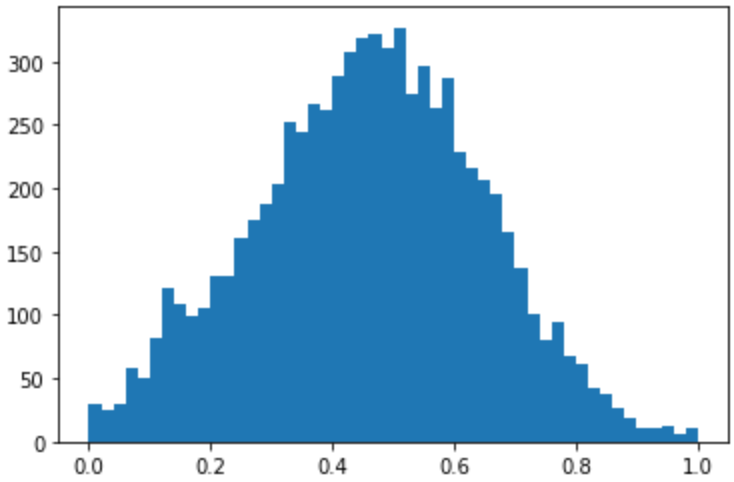
Out[34]:

	leukocyty	hemoglobin	trombocyty	alt	ast	alp	hematokrit	hbver	
count	7438.000000	7438.000000	7438.000000	7438.000000	7438.000000	7438.000000	7438.000000	7438.000000	743
mean	0.503431	0.460417	0.505086	0.543405	0.511094	0.537529	0.516042	0.463554	
std	0.185188	0.185963	0.182574	0.180842	0.179777	0.259957	0.174255	0.167734	
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	
25%	0.379050	0.333894	0.379043	0.421613	0.388955	0.301763	0.383690	0.330819	
50%	0.503659	0.465135	0.506626	0.542307	0.513223	0.588896	0.524337	0.472023	
75%	0.630550	0.589485	0.629275	0.665311	0.634871	0.775774	0.647155	0.595491	
max	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	

Po aplikovaní scalingu vidíme napríklad na hodnotách atribútu hemoglobín zmenu v rozmedzí nadobúdaných hodnôt na rozpätie 0 až 1.

In [35]:
```
plt.hist(X_train['hemoglobin'], bins=50)
```

Out[35]:
```
(array([ 29.,  25.,  29.,  58.,  51.,  82., 121., 108.,  99., 105., 130.,
        131., 160., 175., 187., 204., 252., 245., 267., 262., 289., 307.,
        319., 322., 310., 327., 274., 296., 264., 287., 228., 216., 206.,
        195., 165., 137., 100.,  80.,  94.,  67.,  62.,  43.,  37.,  26.,
         18.,  10.,  11.,  12.,   6.,  10.]),
 array([0.  , 0.02, 0.04, 0.06, 0.08, 0.1 , 0.12, 0.14, 0.16, 0.18, 0.2 ,
        0.22, 0.24, 0.26, 0.28, 0.3 , 0.32, 0.34, 0.36, 0.38, 0.4 , 0.42,
        0.44, 0.46, 0.48, 0.5 , 0.52, 0.54, 0.56, 0.58, 0.6 , 0.62, 0.64,
        0.66, 0.68, 0.7 , 0.72, 0.74, 0.76, 0.78, 0.8 , 0.82, 0.84, 0.86,
        0.88, 0.9 , 0.92, 0.94, 0.96, 0.98, 1.  ]),
 <BarContainer object of 50 artists>)
```



3. Výber atribútov pre strojové učenie

In [36]:
```
from sklearn.feature_selection import SelectKBest
from sklearn.feature_selection import mutual_info_regression
```

Pre výber najvhodnejších atribútov pre strojové učenie sme použili Mutual Information algoritmus, ktorý vyhodnotí závislosti medzi jednotlivými atribútmi. Čím vyššie číslo vyjde, tým je závislosť medzi danými atribútmi vyššia. My sme zisťovali závislosť indikátora od jednotlivých atribútov.

Pre väčšiu presnosť sme spustili algoritmus 5x a výsledky spriemerovali.

In [37]:
```
selector = SelectKBest(mutual_info_regression, k=5)
n = 5 ##overene s vyssim poctom testov, aktualne nastavene nizsie pre rychlejsi beh
scores = pd.Series((0 for col in X_train.columns), index=X_train.columns, dtype=float)
for i in range(n):
    selector.fit(X_train, y_train)
    mi_values = pd.Series(selector.scores_, index=X_train.columns)
    for col in mi_values.index:
        scores[col] += mi_values[col] / n
scores = scores.sort_values(ascending=False)
scores
```

Out[37]:
```
hematokrit    0.080988
hemoglobin    0.076894
er-cv         0.012194
erytrocyty    0.009673
hbver         0.009578
alt           0.008468
sex_m         0.007282
smoker_no     0.007151
alp           0.006276
trombocyty    0.005526
smoker_yes    0.005115
birthdate     0.003372
sex_f         0.002352
ast           0.002313
leukocyty     0.000000
weight        0.000000
etytr         0.000000
dtype: float64
```

Po zoradení všetkých skóre atribútov od najväčšieho po najmenšie sme zistili, že výrazný vplyv na indikátor majú práve atribúty hemoglobín a hematokrit. Ostatné atribúty majú nižší vplyv na indikátor a zároveň sa výška ich

vplyvu mierne odlišuje pri každom spustení algoritmu. Preto sme sa rozhodli vybrať pre strojové učenie prvých 5 najvplyvnejších atribútov. Hodnoty vplyvu sme si zoradili zostupne a vybrali prvých 5, ktorými sú:

In [38]:
```
scores.index[0:5] ## 5 best
```

Out[38]:
```
Index(['hematokrit', 'hemoglobin', 'er-cv', 'erytrocyty', 'hbver'], dtype='object')
```

Tu môžeme vidieť rovnaký výsledok (hemoglobín a hematokrit sú top 2 atribúty s najväčším vplyvom) použitím iného prístupu a to iba striktného vybratia k=5 najlepších atribútov.

In [39]:
```
X_best = SelectKBest(mutual_info_regression, k=5).fit_transform(X_train, y_train)
pd.DataFrame(X_best)
```

Out[39]:

	0	1	2	3	4
0	0.781517	0.489452	0.365514	0.493230	1.0
1	0.324983	0.632758	0.464994	0.562196	0.0
2	0.604928	0.509218	0.540910	0.470804	1.0
3	0.483566	0.613710	0.588659	0.497569	1.0
4	0.723530	0.456906	0.321346	0.477563	0.0
...	...	...	...	...	...
7433	0.666367	0.599123	0.750780	0.116027	1.0
7434	0.003092	0.795075	0.494317	0.229950	0.0
7435	0.620453	0.481418	0.577534	0.300379	1.0
7436	0.363165	0.344020	0.536503	0.393597	1.0
7437	0.442578	0.375136	0.408227	0.519224	1.0

7438 rows × 5 columns

4. Replikovateľnosť predspracovania

Náš doterajší kód predspracovania údajov pre strojové učenie sme v tejto časti upravili pomocou pipeline tak, aby ho bolo možné bez ďalších úprav znovu použiť na predspracovanie testovacej množiny.

In [40]:
```
from sklearn.pipeline import Pipeline
from sklearn.base import TransformerMixin
from sklearn.compose import ColumnTransformer
from sklearn.preprocessing import PowerTransformer
from sklearn.feature_selection import SelectKBest
from sklearn.feature_selection import mutual_info_regression
```

In [41]:
```
class Transformer(TransformerMixin):
    def __init__(self, transform_columns, passthrough_columns):
        self.transformer = ColumnTransformer(transformers=[("PT", PowerTransformer(method=
        self.columns = transform_columns + passthrough_columns

    def fit(self, X_pipe, y=None, **fit_params):
        self.transformer.fit(X_pipe)
        return self
```

```
    def transform(self, X_pipe, **transform_params):
        X_pipe = pd.DataFrame(self.transformer.transform(X_pipe), columns=self.columns)
        return X_pipe
```

In [42]:
```
class Scaler(TransformerMixin):
    def __init__(self):
        self.scaler = MinMaxScaler()

    def fit(self, X_pipe, y=None, **fit_params):
        self.scaler.fit(X_pipe)
        return self

    def transform(self, X_pipe, **transform_params):
        return pd.DataFrame(self.scaler.transform(X_pipe), columns=X_pipe.columns)
```

In [43]:
```
class FeatureSelector(TransformerMixin):
    def __init__(self):
        self.selector = SelectKBest(mutual_info_regression, k=5)

    def fit(self, X_pipe, y_pipe, **fit_params):
        self.y_ = y_pipe
        return self

    def transform(self, X_pipe, **transform_params):
        selector.fit(X_pipe, self.y_)
        scores = pd.Series((0 for col in X_train.columns), index=X_pipe.columns, dtype=flo
        mi_values = pd.Series(selector.scores_, index=X_pipe.columns)
        for col in mi_values.index:
            scores[col] = mi_values[col]
        scores = scores.sort_values(ascending=False)
        # zmenime poradie stlpcov X_pipe podla scores Series, aby mal rovnake poradie stlp
        X_pipe = X_pipe[scores.index]
        return X_pipe
```

In [44]:
```
transform_columns = ['leukocyty', 'hemoglobin', 'trombocyty', 'alt', 'ast', 'alp', 'hematc
passthrough_columns = ['birthdate', 'sex_f', 'sex_m', 'smoker_no', 'smoker_yes']

ppl = Pipeline([
    ('power-transformer', Transformer(transform_columns, passthrough_columns)),
    ('minmax-scaler', Scaler()),
    ('feature-selector', FeatureSelector())
])
```

In [45]:
```
ppl_transformed_train = ppl.fit_transform(X_train_pipe, y_train)
ppl_transformed_test = ppl.fit_transform(X_test_pipe, y_test)
```

Spracujeme trénovaciu a testovaciu sadu X pomocou nami vytvorenej pipeline, pričom výsledné dataframes obsahujú hodnoty atribútov po transformácii na normálne rozdelenie, po scalingu do rozsahu <0, 1>, a pričom poradie stĺpcov korešponduje so zoradením atribútov podľa dôležitosti hodnotenej pomocou Mutual information.

In [46]:
```
ppl_transformed_train.head()
```

Out[46]:

	hematokrit	hemoglobin	er-cv	erytrocyty	hbver	alt	sex_f	alp	trombocyty	ast	sex_m
0	0.489452	0.781517	0.365514	0.493230	0.458152	0.824124	1.0	0.107337	0.417987	0.577061	0.0

	hematokrit	hemoglobin	er-cv	erytrocyty	hbver	alt	sex_f	alp	trombocyty	ast	sex_m
1	0.632758	0.324983	0.464994	0.562196	0.680407	0.625973	0.0	0.768415	0.284371	0.741888	1.0
2	0.509218	0.604928	0.540910	0.470804	0.371400	0.350382	1.0	0.367817	0.415369	0.426540	0.0
3	0.613710	0.483566	0.588659	0.497569	0.347080	0.339698	0.0	0.764350	0.724977	0.669778	1.0
4	0.456906	0.723530	0.321346	0.477563	0.261482	0.498744	0.0	0.178858	0.682564	0.620782	1.0

In [47]:
```
ppl_transformed_test.head()
```

Out[47]:

	hemoglobin	hematokrit	sex_m	erytrocyty	smoker_no	weight	trombocyty	etytr	alp	alt	leul
0	0.318940	0.364483	0.0	0.381989	0.0	0.707070	0.271435	0.442809	0.709487	0.500720	0.4
1	0.529430	0.493597	1.0	0.789102	0.0	0.123772	0.764386	0.429507	0.574181	0.580669	0.4
2	0.385789	0.512664	1.0	0.539253	1.0	0.641928	0.506023	0.482059	0.777975	0.635137	0.3
3	0.791257	0.515158	1.0	0.741048	1.0	0.847144	0.388077	0.890119	0.069697	0.782829	0.5
4	0.751983	0.491068	1.0	0.463318	1.0	0.695270	0.554563	0.480709	0.109337	0.369444	0.4

Vidíme, že prvých 5 stĺpcov s najvyšším skóre sa medzi trénovacou a testovacou sadou odlišuje. Potrebujeme však, aby obe sady obsahovali rovnakých 5 stĺpcov. Preto si túto množinu určíme z testovacej sady a realizujeme výber tej istej podmnožiny atribútov nad oboma sadami. Vybrané najvyššie hodnotené atribúty si môžme vypísať:

In [48]:
```
selected_columns = ppl_transformed_train.columns[0:5]
print(selected_columns)
ppl_transformed_train = ppl_transformed_train[selected_columns]
ppl_transformed_test = ppl_transformed_test[selected_columns]
```

```
Index(['hematokrit', 'hemoglobin', 'er-cv', 'erytrocyty', 'hbver'], dtype='object')
```

Výsledná trénovacia a testovacia X sada vyzerá nasledovne:

In [49]:
```
ppl_transformed_train.head()
```

Out[49]:

	hematokrit	hemoglobin	er-cv	erytrocyty	hbver
0	0.489452	0.781517	0.365514	0.493230	0.458152
1	0.632758	0.324983	0.464994	0.562196	0.680407
2	0.509218	0.604928	0.540910	0.470804	0.371400
3	0.613710	0.483566	0.588659	0.497569	0.347080
4	0.456906	0.723530	0.321346	0.477563	0.261482

In [50]:
```
ppl_transformed_test.head()
```

Out[50]:

	hematokrit	hemoglobin	er-cv	erytrocyty	hbver
0	0.364483	0.318940	0.732350	0.381989	0.500125
1	0.493597	0.529430	0.387655	0.789102	0.664415
2	0.512664	0.385789	0.677845	0.539253	0.764952

	hematokrit	hemoglobin	er-cv	erytrocyty	hbver
3	0.515158	0.791257	0.314309	0.741048	0.246863
4	0.491068	0.751983	0.645236	0.463318	0.550533

Na záver uložíme trénovacie a testovacie sady hodnôt sledovaných atribútov a taktiež príslušné hodnoty indikátora do jednotlivých súborov.

In [51]:
```
ppl_transformed_train.to_csv("data/X_train.csv", sep=',', index=False)
ppl_transformed_test.to_csv("data/X_test.csv", sep=',', index=False)
y_train.to_csv("data/y_train.csv", sep=',', index=False)
y_test.to_csv("data/y_test.csv", sep=',', index=False)
```

In [ ]:
```
```