In [1]:

```
%matplotlib inline
```

In [2]:

```
import matplotlib
from matplotlib import pyplot as plt
print(matplotlib.__version__)
```

```
3.5.3
```

In [3]:

```
import pandas as pd
print(pd.__version__)
```

```
1.4.3
```

In [4]:

```
import numpy as np
np.__version__
```

Out[4]:

```
'1.22.4'
```

In [5]:

```
import sklearn
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.dummy import DummyClassifier
from sklearn import preprocessing
from sklearn.cluster import KMeans
print(sklearn.__version__)
```

```
1.1.2
```

In [6]:

```
import catboost
from catboost import CatBoostClassifier
print(catboost.__version__)
```

```
1.0.6
```

In [7]:

```
url = "https://raw.githubusercontent.com/caravanuden/cardio/master/cardio_train.csv"
df = pd.read_csv(url, sep=';')
df.head()
```

Out[7]:

	id	age	gender	height	weight	ap_hi	ap_lo	cholesterol	gluc	smoke	alco	active	cardio
0	0	18393	2	168	62.0	110	80	1	1	0	0	1	0
1	1	20228	1	156	85.0	140	90	3	1	0	0	1	1
2	2	18857	1	165	64.0	130	70	3	1	0	0	0	1
3	3	17623	2	169	82.0	150	100	1	1	0	0	1	1
4	4	17474	1	156	56.0	100	60	1	1	0	0	0	0

In [8]:

```
X = df.drop('cardio', axis=1)
y = df.cardio
X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=20221013, test_size=0.25)
X_train.shape[0], X_test.shape[0]
```

Out[8]:

```
(52500, 17500)
```

In [9]:

```
model = RandomForestClassifier(n_estimators=15, random_state=20221013)
model.fit(X_train, y_train);
```

In [10]:

```
h = model.predict(X_test)
print(f"RF Accuracy: {(h == y_test).mean()}")
```

```
RF Accuracy: 0.7070857142857143
```

In [11]:

```
# https://scikit-learn.org/stable/modules/generated/sklearn.dummy.DummyClassifier.html
model_dummy = DummyClassifier(strategy="constant", constant=1)
model_dummy.fit(X_train, y_train)
h_dummy = model_dummy.predict(X_test)
print(f"Constant Accuracy: {(h_dummy == y_test).mean()}")
```

```
Constant Accuracy: 0.5010857142857142
```

In [12]:

```
model_random = DummyClassifier(strategy="uniform", random_state=78)
model_random.fit(X_train, y_train)
h_random = model_random.predict(X_test)
print(f"Random Accuracy: {(h_random == y_test).mean()}")
```

```
Random Accuracy: 0.5130285714285714
```

In [13]:

```
def simple_rule(d):
    return (d.age >= 365*55)
```

In [14]:

```
h_simple = simple_rule(X_test)
print(f"Simple Rule Accuracy: {(h_simple == y_test).mean()}")
```

```
Simple Rule Accuracy: 0.5958857142857142
```

In [15]:

```
model2 = CatBoostClassifier(random_state=20221013, verbose = 0)
model2.fit(X_train, y_train);
```

In [16]:

```
h_catboost = model2.predict(X_test)
print(f"CatBoost Accuracy: {(h_catboost == y_test).mean()}")
```

```
CatBoost Accuracy: 0.7346285714285714
```

In [17]:

```
# Давайте сделаем всех старше на 5 лет
X_aged = X_test.assign(age = lambda x: x.age + 365*5)
h_aged = model.predict(X_aged)
print(f"RF_Target drift: {h_aged.mean() - h.mean()}")
```

```
RF_Target drift: 0.07754285714285719
```

In [18]:

```
h_aged2 = model2.predict(X_aged)
print(f"CB_Target drift: {h_aged2.mean() - h.mean()}")
```

```
CB_Target drift: 0.07211428571428574
```

In [19]:

```
h_simple_aged = simple_rule(X_aged)
print(f"Simple Rule Target Drift: {h_simple_aged.mean() - h_simple.mean()}")
```

```
Simple Rule Target Drift: 0.2503428571428571
```

In [20]:

```
h_dummy_aged = model_dummy.predict(X_aged)
print(f"Dummy Target Drift: {h_dummy_aged.mean() - h_dummy.mean()}")
```

```
Dummy Target Drift: 0.0
```

In [21]:

```
idx_man = (X_test.gender == 2)
```

In [22]:

```
def slice_accuracy(h, y, idx):
    acc = dict()
    for lbl in np.unique(idx):
        acc[lbl] =  (h[idx == lbl] == y[idx == lbl]).mean() - (h[idx != lbl] == y[idx != lbl]).mean()
    return acc
```

In [23]:

```
# Точность RF модели для женщин выше на 0,17%
slice_accuracy(h, y_test, idx_man)
```

Out[23]:

```
{False: 0.0017739203519019364, True: -0.0017739203519019364}
```

In [24]:

```
# Точность CB модели для женщин выше на 0,53%
slice_accuracy(h_catboost, y_test, idx_man)
```

Out[24]:

```
{False: 0.005326591335765629, True: -0.005326591335765629}
```

In [25]:

```
scaled_X_test = preprocessing.StandardScaler().fit_transform(X_test)
```

In [26]:

```
kmeans = KMeans(n_clusters=4, random_state=20221013).fit(scaled_X_test)
```

In [27]:

```
# Для uheggs 2 RF-модель работает хуже на 7%
slice_accuracy(h, y_test, kmeans.labels_)
```

Out[27]:

```
{0: 0.008310089084933359,
 1: -0.010667841868729488,
 2: 0.07161491099566786,
 3: -0.00942936433783892}
```

In [28]:

```
# Похоже, это те, кто пьет алкоголь
X_test.loc[kmeans.labels_ == 2]
```

Out[28]:

	id	age	gender	height	weight	ap_hi	ap_lo	cholesterol	gluc	smoke	alco	active
17587	25120	19030	2	170	76.0	120	70	1	1	1	1	1
7197	10253	18253	2	175	71.0	130	80	1	1	0	1	1
15540	22207	23356	2	179	63.0	110	80	2	2	1	1	1
158	213	16028	1	157	69.0	120	80	1	1	0	1	1
38430	54868	19137	1	160	70.0	130	90	1	1	1	1	1
...	...	...	...	...	...	...	...	...	...	...	...	...
56100	80040	21979	2	169	68.0	120	80	1	3	0	1	1
28183	40288	21628	2	181	100.0	125	70	1	1	0	1	1
22359	31939	20547	2	170	86.0	120	80	1	1	0	1	1
40784	58279	16721	1	165	94.0	120	60	1	1	0	1	1
58293	83179	20426	2	160	62.0	120	80	1	1	1	1	1

911 rows × 12 columns

In [29]:

```
# Для тех, кто пьет алкоголь, CB-модель работает хуже на 4.8%
slice_accuracy(h_catboost, y_test, kmeans.labels_)
```

Out[29]:

```
{0: 0.008898763189759573,
 1: -0.011082390186475943,
 2: 0.04834939158961549,
 3: -0.004774455087559137}
```

In [ ]: