# 1 Nadstavba - všeobecný model pre logistickú regresiu pomocou kvázinewtonovských alebo gradientných metód

Ako nadstavbu sme si zvolili implementovať všeobecný model na binárnu klasifikáciu, teda nie viazaný len na dáta o solventnosti. V neskorších projektoch by teda malo byť možné používať náš kód ako balík, z ktorého sa importuje trieda na binárnu klasifikáciu pomocou:

```
from logistic_regression import LogisticRegression
```

Následne, pomocou metódy `LogisticRegression.fit()` sa zistí vektor koeficientov $x$. Táto metóda očakáva trénovaciu maticu nezávislých vektorov `u`, vektor binárnych dát `v`, označenie metódy, ktorá má byť použitá na minimalizáciu účelovej funkcie (buď `"BFGS"`, `"DFP"`, `"Cauchy"` alebo `"Grad-Const"`) a v prípade použitia BFGS alebo DFP metódy očakáva aj požadovanú dĺžku kroku (buď `"optimal"` alebo `"suboptimal"`). Príklad použitia môže vyzerať takto:

```
log_reg = LogisticRegression()
log_reg.fit(u=u_train, v=v_train, method="DFP", step="optimal")
```

Vektor predikovaných $v^{i'}$ sa získa pomocou `LogisticRegression.predict()`, ktorá očakáva testovaciu maticu `u_test`, s rovnakým počtom stĺpcov, ako matica `u` (musí tomuto volaniu však predchádzať volanie metódy `fit()`):

```
log_reg.predict(u_test)
```

Konvergenčný graf sa generuje pomocou `LogisticRegression.visualize()`, ktorý očakáva ako argument inštanciu triedy `matplotlib.pyplot.Axes`:

```
import matplotlib.pyplot as plt
ax = plt.gca()
log_reg.visualize(ax)
```

Práca s touto triedou aj s minimalizačnými metódami je potrebne popísaná v dokumentácii v samotnom kóde.

Príklad použitia tohto modulu sa dá spustiť v terminálovom okne, ktoré je otvorené v priečinku `source` jedným z príkazov (podľa operačného systému, resp. nainštalovaného nástoja `make`):

```
python -m logistic_regression
python3 -m logistic_regression
make log_reg
```