**Consumerprice Index of Basel**

****

**CAS MAIN - Deep Learning: Beate Sick, Oliver Dürr**

**Constantin Sluka, Christian Moser**

**April 2022**

**Data, Motivation & Objective**

**Data found on data.bs.ch**

* Time series over 10+ years
* month-on-month price increase (mmpi) «Monatsteuerung»,   
  year-on-year price increase (yypi) «Jahresteuerung»,   
  month and year
* 400+ products, divided into
* 12 main categories

**Inspired by our banknote exercise**

* Are mmpi and yypi sufficient to explain (classify) the products into the main categories using Deep Learning?
* Spoiler: probably not, because some categories are correlated

**The Dataset**

* From 01.01.2009 until 31.12.2019
* 430 products
* 12 main categories (max. 97 prod in ‘Nahrungsmittel’, min. 9 prod in ‘Unterricht’)

Input features:

* Monatsteuerung, Jahresteuerung (z-transformed), Month (cos-transformed)

Label:

* Hauptgruppe (one-hot encoded)

****

**Train - Validation - Test data**

Randomly shuffle products: 300 (70%, train), 2 x 65 (15%, validation, test)

→ All index data of a product belong to the same data subset

**Data Cleaning**

* Remove missing data and historic products, resolve duplicated main categories
* Remove discontinued products (index change: -100%)
* Feature transformation

**Our Baseline**

Let’s first use “classic” classifiers:

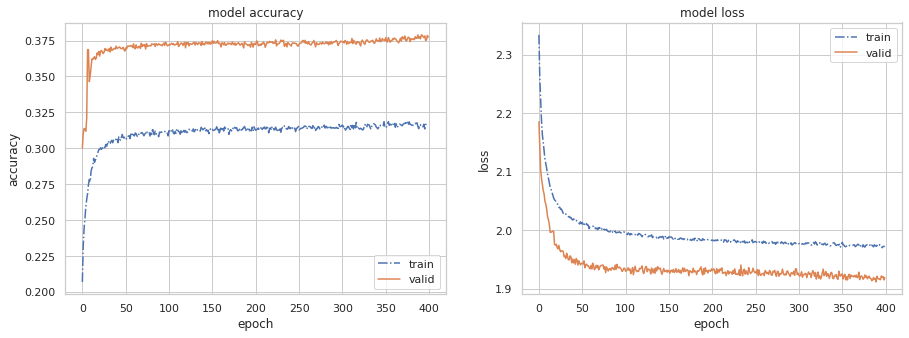
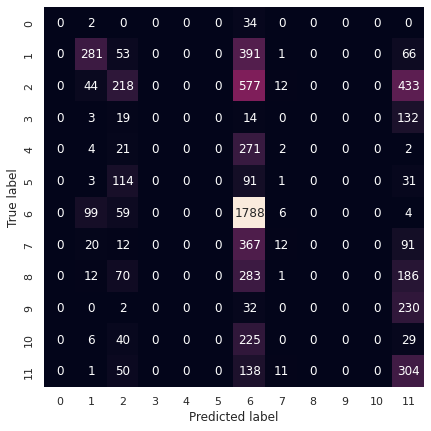
| **Method** | **Accuracy on validation set** |
| --- | --- |
| LogisticRegression (using scikit-learn) | 28.7% |
| KNeighborsClassifier (using scikit-learn) | 29.5% (k=10), 35.1% (k=50) |
| Random | 1/12 ~ 8.3% |

Accuracy of Baseline classifiers

**1st Attempt: predict all 12 classes**

Model : sequential Model, 1 hidden layers + Dropout (0.3), Sigmoid-activated, softmax- output-layer of size 12  
loss: categorical\_crossentropy, optimizer: adam, metrics: accuracy

Model performance Confusion Matrix

That’s not what we wanted. Still better than baseline, but…

Note: acc\_valid > acc\_train. Maybe because DropOut is turned off in the validation run?

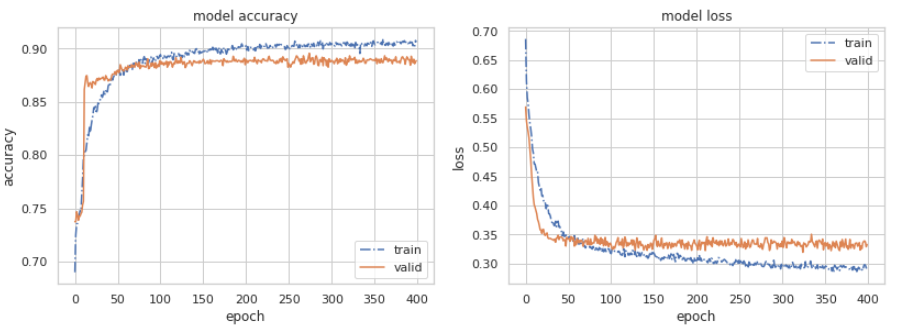
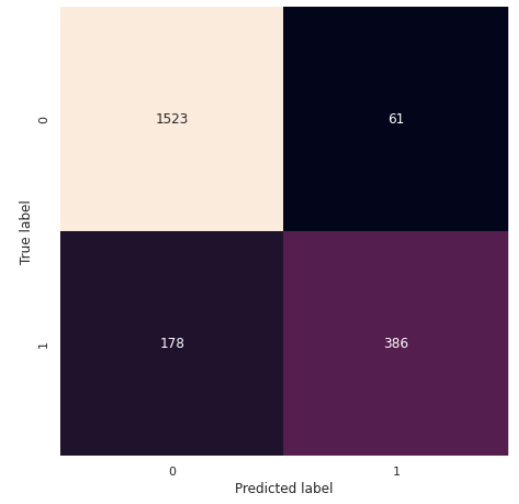
**So, What went wrong?** Probably too much correlation & overlap between classes

**2nd Attempt: predict only 2 classes**

Domain knowledge: how about Food vs. Energy?

Model: sequential Model, 2 (smaller) hidden layers + Dropout (0.3), ReLU-activated,   
softmax- output-layer of size 2, loss: categorical\_crossentropy, optimizer: adam  
metrics: accuracy

Model performance Confusion Matrix

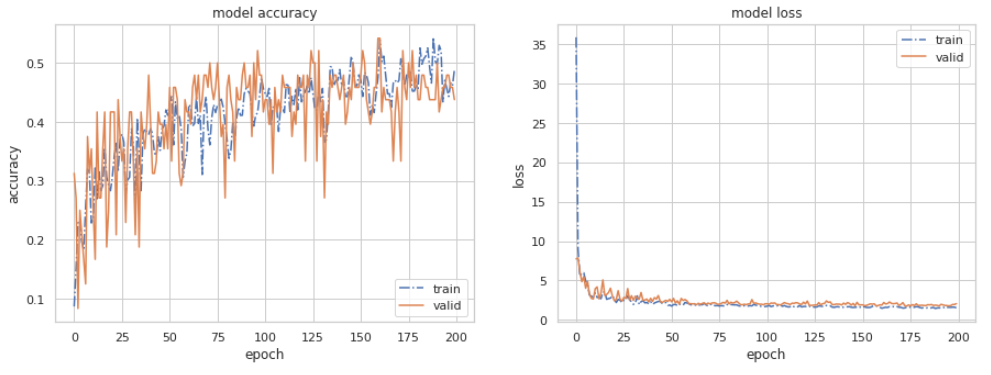
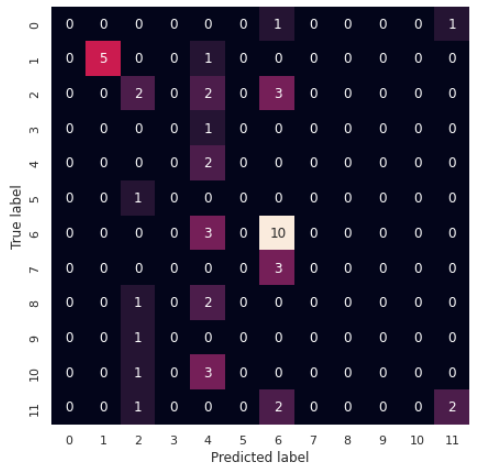
Better than baseline (kNN (k=40) ~ 85%)

**3rd Attempt: Let’s switch axes**

use all data of the time series as input features (p=132) (but then… N = 315 obs only)

Model: sequential Model, 1 hidden layer, No Dropout, ReLU-activated,   
softmax- output-layer of size 12, loss: categorical\_crossentropy, optimizer: adam  
metrics: accuracy

Model performance Confusion Matrix

**That ain’t any better…**

The trained models were either as wiggly as the one depicted, or settled to a stable situation where every observation was then predicted the same label.

Conclusion: Curse of dimensionality, not enough data by a mile.

**Lessons learned, Summary & Ideas to go from here**

An early mishap we finally detected was that by splitting the data before one-hot-encoding using the pd.factorize() method we had introduced different labels for the same class in training and validation sets (very bad, sort = True helps 😉).

The data eluded us quite a bit – or we were a bit too optimistic about what a clever Neural Network could possibly return on input data which is unbalanced and correlated.

So, careful selection of ‘what can be predicted’ is mandatory.

Also clearly visible: having enough data is absolutely key.

While we were working with the data, ideas of other interesting applications with the same data came to our minds: as we’re working with time series, we could use the index prices and make predictions for the future. We’d then employ a 1D Convolutional NN with time dilitation to try to predict possible seasonal trends.

**Acknowledgements & References**

Thanks to: The Tensorchiefs (<https://tensorchiefs.github.io>)

Dataset: <https://data.bs.ch/explore/dataset/100003/information/>

Data Food vs Energy

