Time-series analysis and prediction for the HICP dataset of Eurostat

Capstone Project Defence
EPFL Extension School

Cornelia Blanke
24/01/2022

Overview of my Presentation

- Idea of the Project
  - ➔ *Origin, Structure and Explanation of the HICP Data*
- Exploratory Data Analysis
  - ➔ *Some selected Findings*
- Machine Learning Task 1
  - ➔ *Analyze the Feature Importance for the Weights dataset*
- Machine Learning Task 2
  - ➔ *Predict year 2019 out of previous years by Machine Learning*

HICP = Harmonized Index of Consumer Prices

Definition

- The Harmonized Index of Consumer Prices (HICP) is an indicator that the member states of EU and EFTA calculate based on a harmonized method and that allows comparing inflation internationally.
- The European Statistical Office **Eurostat** publishes every month the results of all participating countries.

Source: FSO

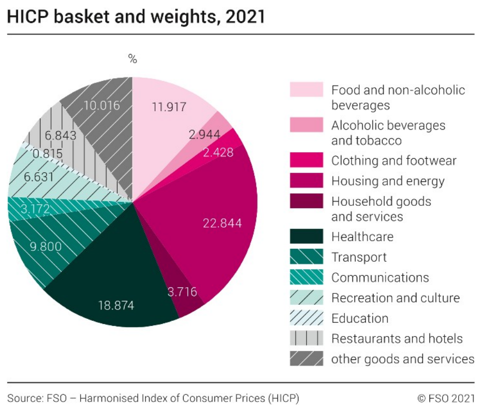
HICP = Harmonized Index of Consumer Prices

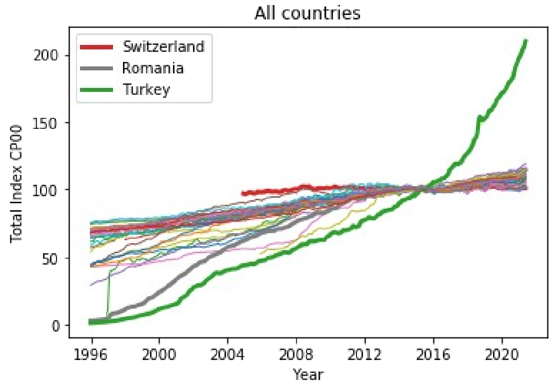
The weights

- describe the composition of product groups in the consumer basket
- are different in each country
- are yearly updated

The indices

- describe the temporal development of the prices of product groups
- are relative to 2015 = 100
- are monthly updated

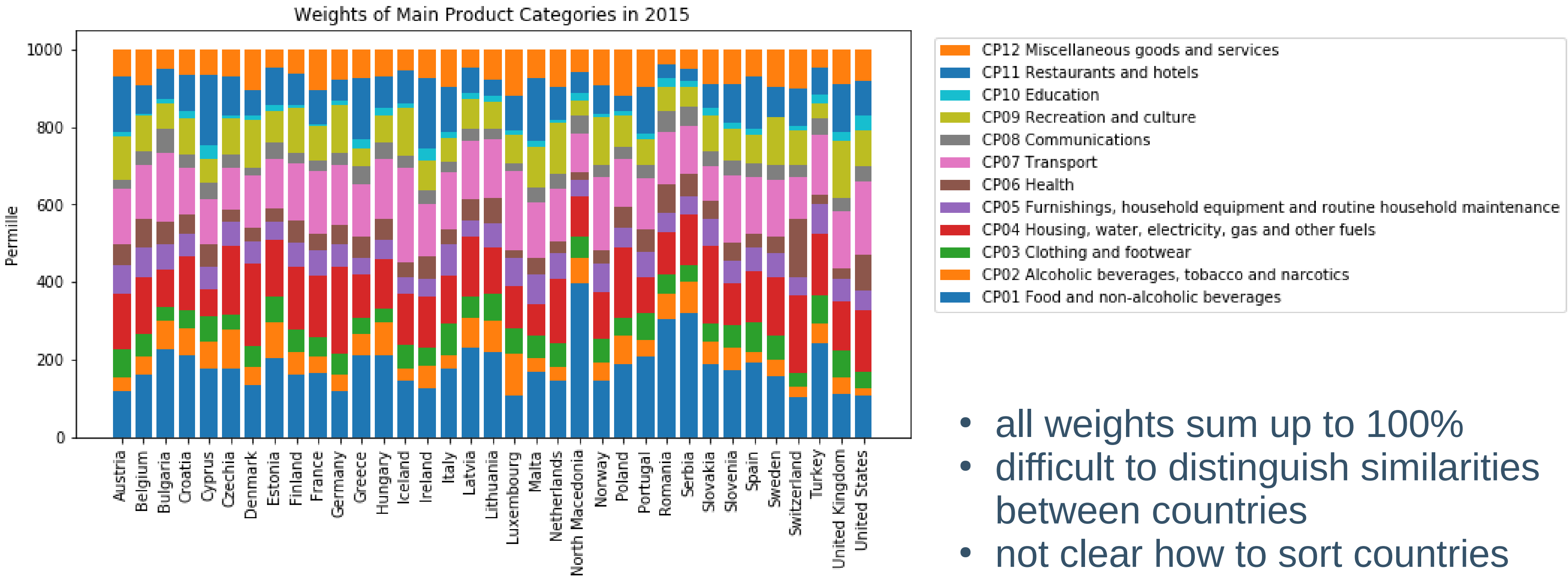




Exploratory Data Analysis

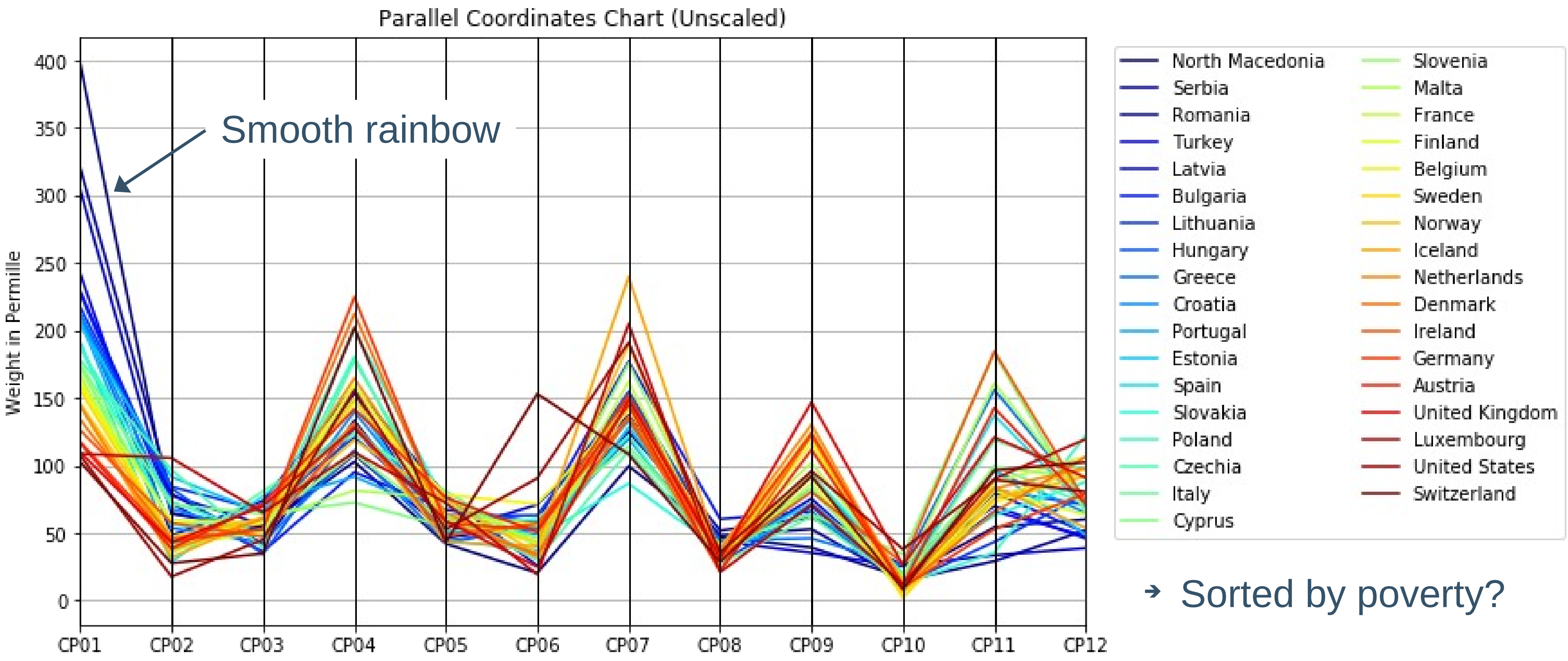
Some selected Findings

EDA – Analysis of the weights



- all weights sum up to 100%
- difficult to distinguish similarities between countries
- not clear how to sort countries

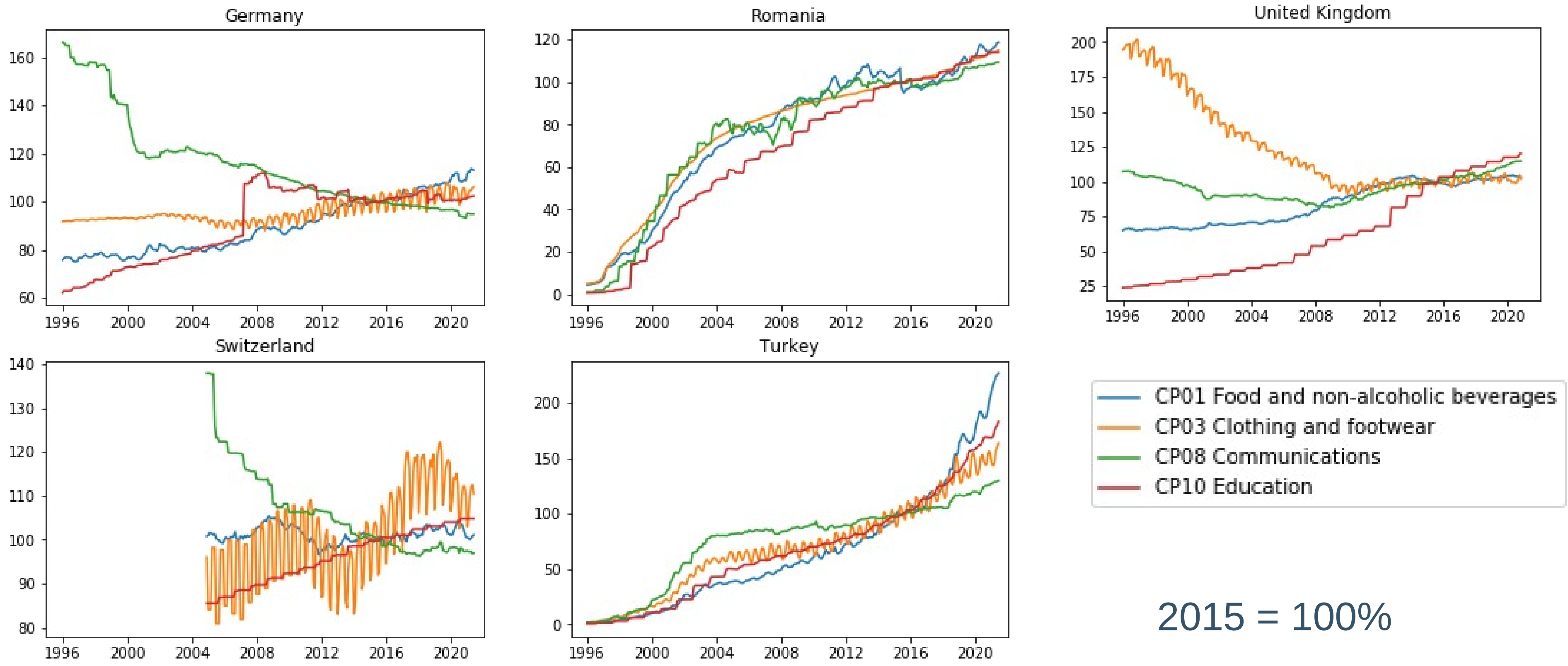
EDA – Analysis of the weights



→ Sorted by poverty?

EDA – Time Series Analysis

Indices of interest for ML task 2



2015 = 100%

EDA – Compute the monthly rates of change

Often used in ML:

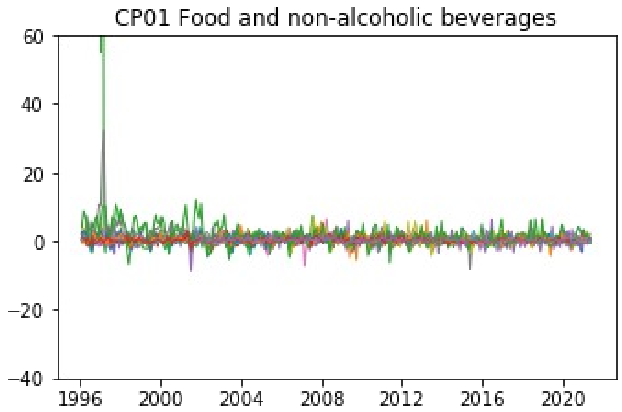
The (absolute) change

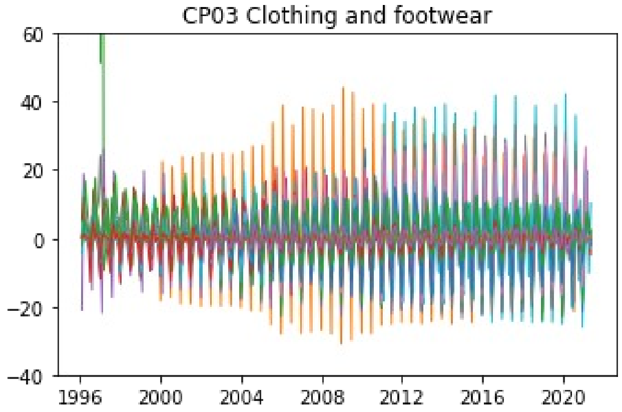
$$\Delta I(t_n) = I(t_n) - I(t_{n-1})$$

Here:

The (relative) rate of change

$$M(t_n) = \left(\frac{I(t_n) - I(t_{n-1})}{I(t_{n-1})} \right) \cdot 100\%$$





Machine Learning Task 1

For each of the main product categories 1-12, which characteristics (= features) have the strongest impact on the weights dataset?

Use ML techniques to judge about ***feature importance****.*

ML1 – Data Preparation

	Feat. 1	Feat. 2	Feat. 3	...	Feat. 29	Weight CP01	Weight CP02	...	Weight CP12
Country 1									
Country 2									
Country 3									
⋮									
⋮									
⋮									
Country 31									

Features = Characteristics of Countries

Targets = Weights in 2015

ML1 – Linear Regression

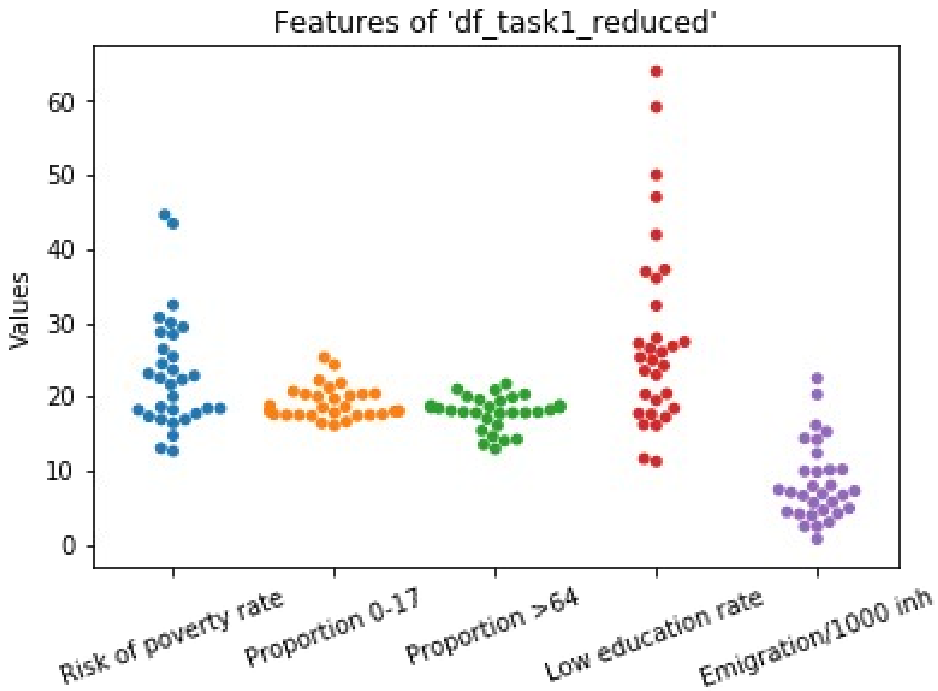
If all features are **scaled** to the same range, their coefficients in a **Linear Regression** model reflect their importance.

<u>But:</u>

- 29 features, 31 samples
- no validation or test set

➔ high risk of over-fitting

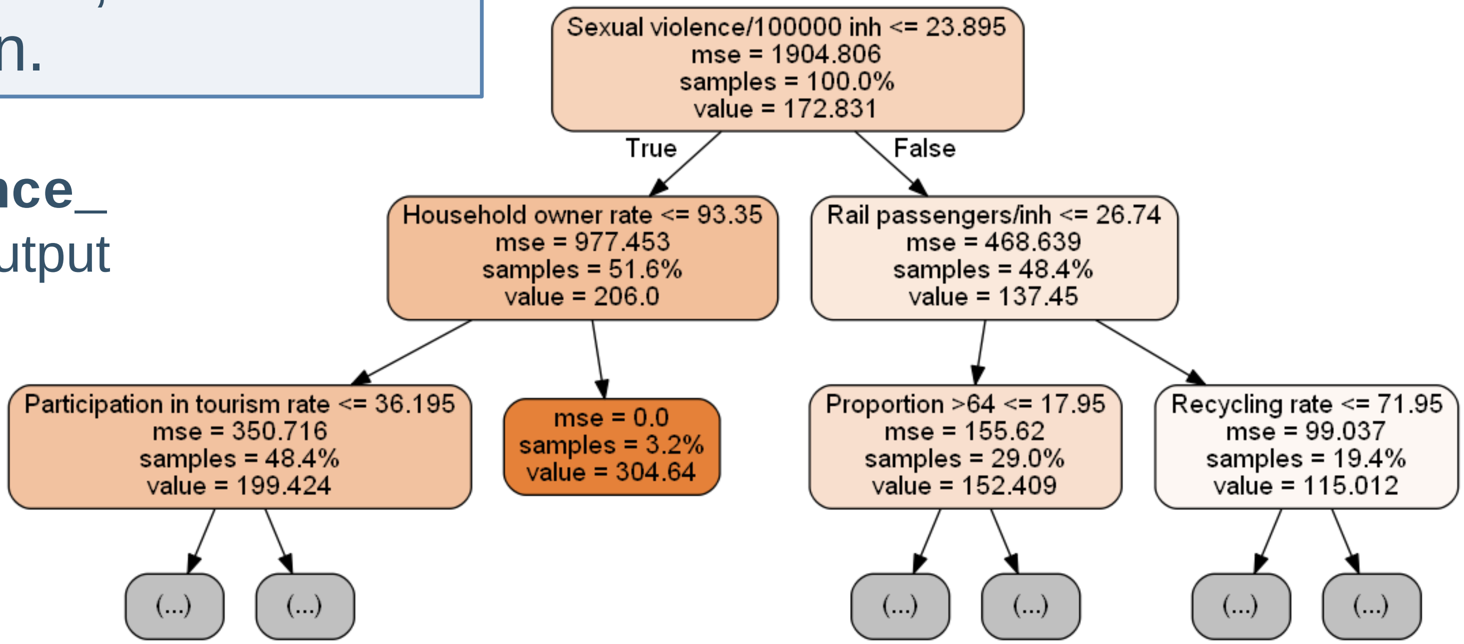
➔ only use 5 features



ML1 – Decision Tree Regressor

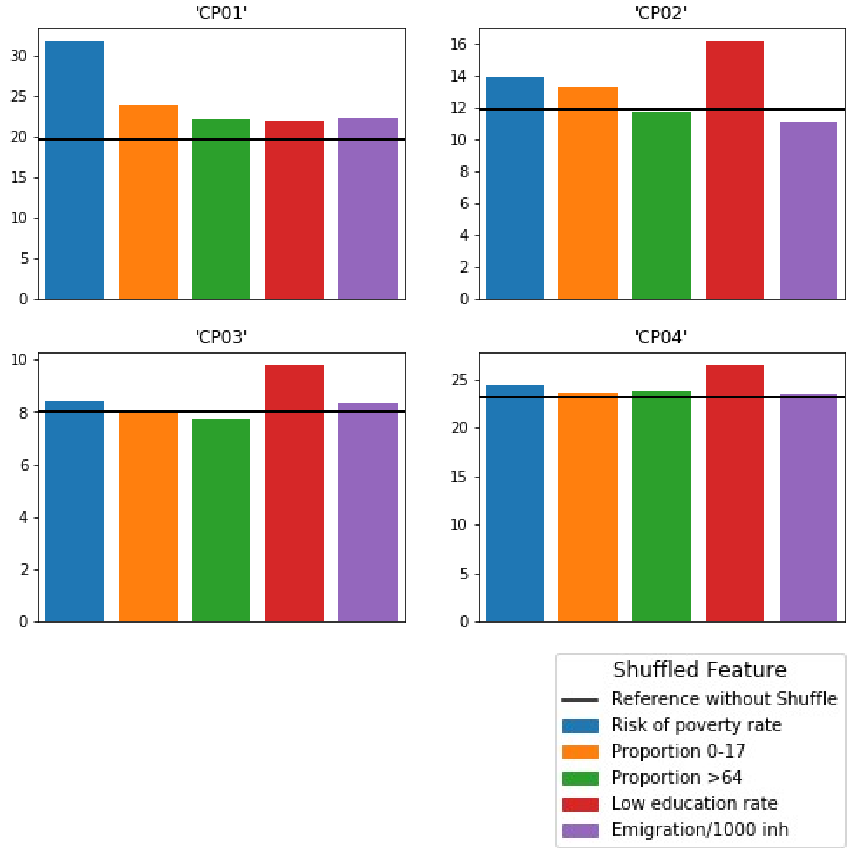
A **Decision Tree** provides an intuitive visualization about which feature matters most, which is the second, and so on.

`feature_importance_` provides numerical output



ML1 – Feature Permutation in kNN

Idea of **feature permutation**:
If it does not matter if a feature were disturbed, then that feature was not an important one.



<u>Algorithm:</u>

1. Shuffle each feature randomly
2. Repeat it five times
3. Compare MAEs (e.g. in kNN)

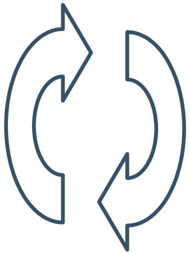
Machine Learning Task 2

For 20 pre-selected indices,
predict all months (i.e. 12) of year 2019 out of years 2005-2018.

Use different approaches and ML models and compare.

ML2 – Recursive prediction of one time-series

1. Data preparation: create 12 lagged features
2. Split into train and test set
3. Learn a ML model that is able to predict the next month out of 12 previous months
4. Use ML model to predict the next month
5. Feed this result to the known values and predict therefrom the next month



Recursion

ML2 – Recursive prediction of one time-series

↓12 lagged time-series

time-series

train set

test set

recursive prediction and feed back

x1	x2	x3	x4	x5	x6	x7	x8	x9	x10	x11	x12	x13
x2	x3	x4	x5	x6	x7	x8	x9	x10	x11	x12	x13	x14
x3	x4	x5	x6	x7	x8	x9	x10	x11	x12	x13	x14	x15
x4	x5	x6	x7	x8	x9	x10	x11	x12	x13	x14	x15	x16
x5	x6	x7	x8	x9	x10	x11	x12	x13	x14	x15	x16	x17
x6	x7	x8	x9	x10	x11	x12	x13	x14	x15	x16	x17	x18
x7	x8	x9	x10	x11	x12	x13	x14	x15	x16	x17	x18	x19
x8	x9	x10	x11	x12	x13	x14	x15	x16	x17	x18	x19	x20
x9	x10	x11	x12	x13	x14	x15	x16	x17	x18	x19	x20	x21
x10	x11	x12	x13	x14	x15	x16	x17	x18	x19	x20	x21	x22
x11	x12	x13	x14	x15	x16	x17	x18	x19	x20	x21	x22	x23
x12	x13	x14	x15	x16	x17	x18	x19	x20	x21	x22	x23	x24
x13	x14	x15	x16	x17	x18	x19	x20	x21	x22	x23	x24	x25
x14	x15	x16	x17	x18	x19	x20	x21	x22	x23	x24	x25	x26
x15	x16	x17	x18	x19	x20	x21	x22	x23	x24	x25	x26	x27
x16	x17	x18	x19	x20	x21	x22	x23	x24	x25	x26	x27	x28
x17	x18	x19	x20	x21	x22	x23	x24	x25	x26	x27	x28	x29
x18	x19	x20	x21	x22	x23	x24	x25	x26	x27	x28	x29	x30
x19	x20	x21	x22	x23	x24	x25	x26	x27	x28	x29	x30	x31
x20	x21	x22	x23	x24	x25	x26	x27	x28	x29	x30	x31	x32
x21	x22	x23	x24	x25	x26	x27	x28	x29	x30	x31	x32	x33
x22	x23	x24	x25	x26	x27	x28	x29	x30	x31	x32	x33	x34
x23	x24	x25	x26	x27	x28	x29	x30	x31	x32	x33	x34	x35
x24	x25	x26	x27	x28	x29	x30	x31	x32	x33	x34	x35	x36
x25	x26	x27	x28	x29	x30	x31	x32	x33	x34	x35	x36	x37
x26	x27	x28	x29	x30	x31	x32	x33	x34	x35	x36	x37	x38
x27	x28	x29	x30	x31	x32	x33	x34	x35	x36	x37	x38	x39
x28	x29	x30	x31	x32	x33	x34	x35	x36	x37	x38	x39	x40
x29	x30	x31	x32	x33	x34	x35	x36	x37	x38	x39	x40	x41
x30	x31	x32	x33	x34	x35	x36	x37	x38	x39	x40	x41	x42
x31	x32	x33	x34	x35	x36	x37	x38	x39	x40	x41	x42	x43
x32	x33	x34	x35	x36	x37	x38	x39	x40	x41	x42	x43	x44
x33	x34	x35	x36	x37	x38	x39	x40	x41	x42	x43	x44	x45
x34	x35	x36	x37	x38	x39	x40	x41	x42	x43	x44	x45	x46
x35	x36	x37	x38	x39	x40	x41	x42	x43	x44	x45	x46	x47
x36	x37	x38	x39	x40	x41	x42	x43	x44	x45	x46	x47	x48

ML2 – Recursive prediction of one time-series

```
from sklearn.linear_model import LinearRegression
lr1 = LinearRegression()

# initialize y_pred
y_lr1_pred = np.empty((20,12))

# outer loop over all indices
for i in np.arange(df_task2.shape[1]):

    # fit LR model for index i
    lr1.fit(X_train[i], y_train[i])

    # create a local copy
    X_pred = X_test[i].copy()

    # inner loop over all months
    for month in np.arange(12):

        # predict month
        y_lr1_pred[i,month] = lr1.predict(X_pred)

        # shift data by 1 to the left, append y_pred, reshape to (1,12)
        X_pred = np.append(X_pred[:,1:], y_lr1_pred[i,month]).reshape(1,12)
```

ML2 – One-step multi-output prediction of one time-series

1. Data preparation:
   create 12 lagged features and 1 un-lagged and 11 lagged targets
2. Split into train and test set
3. Learn a multi-target ML model that is able to predict the next 12 months out of 12 previous months
4. Use ML model to predict the next 12 months

ML2 – One-step multi-output prediction of one time-series

↓12 lagged time-series   time-series   ↑11 lagged time-series

x1	x2	x3	x4	x5	x6	x7	x8	x9	x10	x11	x12	x13	x14	x15	x16	x17	x18	x19	x20	x21	x22	x23	x24
x2	x3	x4	x5	x6	x7	x8	x9	x10	x11	x12	x13	x14	x15	x16	x17	x18	x19	x20	x21	x22	x23	x24	x25
x3	x4	x5	x6	x7	x8	x9	x10	x11	x12	x13	x14	x15	x16	x17	x18	x19	x20	x21	x22	x23	x24	x25	x26
x4	x5	x6	x7	x8	x9	x10	x11	x12	x13	x14	x15	x16	x17	x18	x19	x20	x21	x22	x23	x24	x25	x26	x27
x5	x6	x7	x8	x9	x10	x11	x12	x13	x14	x15	x16	x17	x18	x19	x20	x21	x22	x23	x24	x25	x26	x27	x28
x6	x7	x8	x9	x10	x11	x12	x13	x14	x15	x16	x17	x18	x19	x20	x21	x22	x23	x24	x25	x26	x27	x28	x29
x7	x8	x9	x10	x11	x12	x13	x14	x15	x16	x17	x18	x19	x20	x21	x22	x23	x24	x25	x26	x27	x28	x29	x30
x8	x9	x10	x11	x12	x13	x14	x15	x16	x17	x18	x19	x20	x21	x22	x23	x24	x25	x26	x27	x28	x29	x30	x31
x9	x10	x11	x12	x13	x14	x15	x16	x17	x18	x19	x20	x21	x22	x23	x24	x25	x26	x27	x28	x29	x30	x31	x32
x10	x11	x12	x13	x14	x15	x16	x17	x18	x19	x20	x21	x22	x23	x24	x25	x26	x27	x28	x29	x30	x31	x32	x33
x11	x12	x13	x14	x15	x16	x17	x18	x19	x20	x21	x22	x23	x24	x25	x26	x27	x28	x29	x30	x31	x32	x33	x34
x12	x13	x14	x15	x16	x17	x18	x19	x20	x21	x22	x23	x24	x25	x26	x27	x28	x29	x30	x31	x32	x33	x34	x35
x13	x14	x15	x16	x17	x18	x19	x20	x21	x22	x23	x24	x25	x26	x27	x28	x29	x30	x31	x32	x33	x34	x35	x36
~~x14~~	~~x15~~	~~x16~~	~~x17~~	~~x18~~	~~x19~~	~~x20~~	~~x21~~	~~x22~~	~~x23~~	~~x24~~	~~x25~~	~~x26~~	~~x27~~	~~x28~~	~~x29~~	~~x30~~	~~x31~~	~~x32~~	~~x33~~	~~x34~~	~~x35~~	~~x36~~	~~x37~~
~~x15~~	~~x16~~	~~x17~~	~~x18~~	~~x19~~	~~x20~~	~~x21~~	~~x22~~	~~x23~~	~~x24~~	~~x25~~	~~x26~~	~~x27~~	~~x28~~	~~x29~~	~~x30~~	~~x31~~	~~x32~~	~~x33~~	~~x34~~	~~x35~~	~~x36~~	~~x37~~	~~x38~~
~~x16~~	~~x17~~	~~x18~~	~~x19~~	~~x20~~	~~x21~~	~~x22~~	~~x23~~	~~x24~~	~~x25~~	~~x26~~	~~x27~~	~~x28~~	~~x29~~	~~x30~~	~~x31~~	~~x32~~	~~x33~~	~~x34~~	~~x35~~	~~x36~~	~~x37~~	~~x38~~	~~x39~~
~~x17~~	~~x18~~	~~x19~~	~~x20~~	~~x21~~	~~x22~~	~~x23~~	~~x24~~	~~x25~~	~~x26~~	~~x27~~	~~x28~~	~~x29~~	~~x30~~	~~x31~~	~~x32~~	~~x33~~	~~x34~~	~~x35~~	~~x36~~	~~x37~~	~~x38~~	~~x39~~	~~x40~~
~~x18~~	~~x19~~	~~x20~~	~~x21~~	~~x22~~	~~x23~~	~~x24~~	~~x25~~	~~x26~~	~~x27~~	~~x28~~	~~x29~~	~~x30~~	~~x31~~	~~x32~~	~~x33~~	~~x34~~	~~x35~~	~~x36~~	~~x37~~	~~x38~~	~~x39~~	~~x40~~	~~x41~~
~~x19~~	~~x20~~	~~x21~~	~~x22~~	~~x23~~	~~x24~~	~~x25~~	~~x26~~	~~x27~~	~~x28~~	~~x29~~	~~x30~~	~~x31~~	~~x32~~	~~x33~~	~~x34~~	~~x35~~	~~x36~~	~~x37~~	~~x38~~	~~x39~~	~~x40~~	~~x41~~	~~x42~~
~~x20~~	~~x21~~	~~x22~~	~~x23~~	~~x24~~	~~x25~~	~~x26~~	~~x27~~	~~x28~~	~~x29~~	~~x30~~	~~x31~~	~~x32~~	~~x33~~	~~x34~~	~~x35~~	~~x36~~	~~x37~~	~~x38~~	~~x39~~	~~x40~~	~~x41~~	~~x42~~	~~x43~~
~~x21~~	~~x22~~	~~x23~~	~~x24~~	~~x25~~	~~x26~~	~~x27~~	~~x28~~	~~x29~~	~~x30~~	~~x31~~	~~x32~~	~~x33~~	~~x34~~	~~x35~~	~~x36~~	~~x37~~	~~x38~~	~~x39~~	~~x40~~	~~x41~~	~~x42~~	~~x43~~	~~x44~~
~~x22~~	~~x23~~	~~x24~~	~~x25~~	~~x26~~	~~x27~~	~~x28~~	~~x29~~	~~x30~~	~~x31~~	~~x32~~	~~x33~~	~~x34~~	~~x35~~	~~x36~~	~~x37~~	~~x38~~	~~x39~~	~~x40~~	~~x41~~	~~x42~~	~~x43~~	~~x44~~	~~x45~~
~~x23~~	~~x24~~	~~x25~~	~~x26~~	~~x27~~	~~x28~~	~~x29~~	~~x30~~	~~x31~~	~~x32~~	~~x33~~	~~x34~~	~~x35~~	~~x36~~	~~x37~~	~~x38~~	~~x39~~	~~x40~~	~~x41~~	~~x42~~	~~x43~~	~~x44~~	~~x45~~	~~x46~~
~~x24~~	~~x25~~	~~x26~~	~~x27~~	~~x28~~	~~x29~~	~~x30~~	~~x31~~	~~x32~~	~~x33~~	~~x34~~	~~x35~~	~~x36~~	~~x37~~	~~x38~~	~~x39~~	~~x40~~	~~x41~~	~~x42~~	~~x43~~	~~x44~~	~~x45~~	~~x46~~	~~x47~~
x25	x26	x27	x28	x29	x30	x31	x32	x33	x34	x35	x36	x37	x38	x39	x40	x41	x42	x43	x44	x45	x46	x47	x48

useless rows

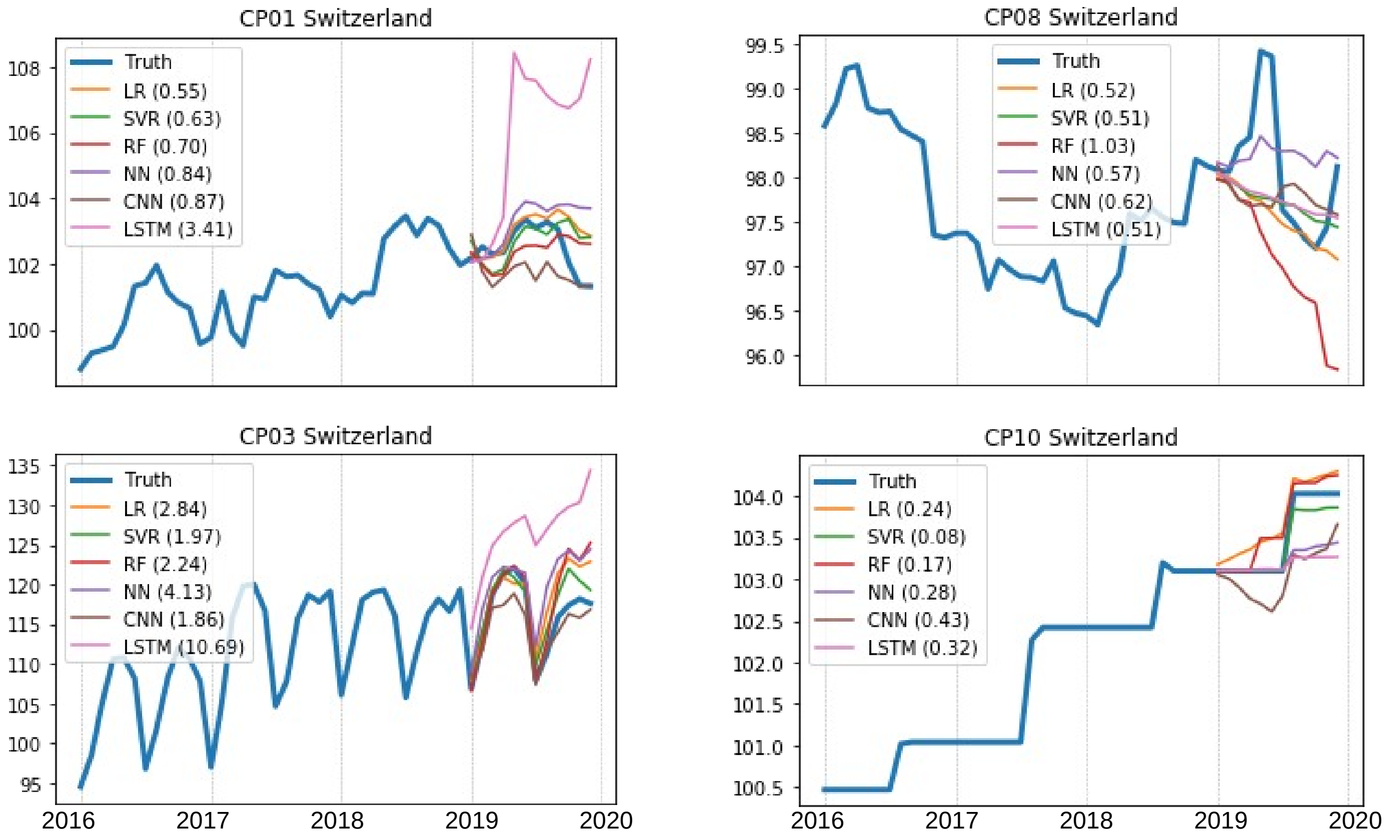
last row = test set

ML2 – One-step multi-output prediction of one time-series

```
lr2 = LinearRegression()

# initialize y_pred
Y_lr2_pred = np.empty((20,1,12))

# outer loop over all indices
for i in np.arange(df_task2.shape[1]):

    # fit LR model for index i
    lr2.fit(X_train[i], Y_train[i])

    # predict next year
    Y_lr2_pred[i] = lr2.predict(X_test[i])
```

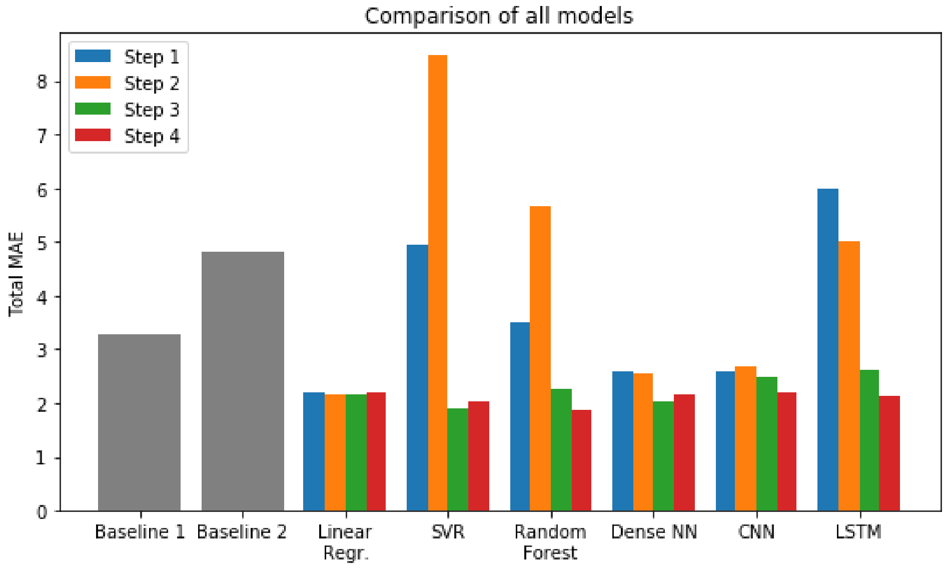
ML2 – Comparison and Results

<u>Step 3:</u>
Recursive approach
based on
rates of change



ML2 – Comparison and Results

- Rather good performance with Linear Regression
- Several models perform better with stationary rates of change
- Deep learning is not superior here
- Approach number 4 runs faster
- Problem: dataset is too small and contains too much randomness



Keep it simple!

Summary

Exploratory Data Analysis

- Data preparation straightforward
- Insights into main product groups and sub-categories
  - for the weights
  - for the indices
- Find similarities between countries
- Computation of rates of change
- Discover seasonality

Machine Learning Task 1

- Use ML to judge about feature importance
- Features are strongly correlated

Machine Learning Task 2

- Time-series prediction using different approaches and ML models (scikit-learn, tensorflow)
- Good results with basic models
- No enhancement with more effort