**Aim:**

**Predict the particulate matter in the atmosphere close to frequently used main streets in the Stuttgart valley basin. How precise is our algorithm to the real measured concentration of particulate matter?**

**Abstract**

**Why important?**

Air pollution is currently a major task for municipality (Stadtverwaltung) of bigger cities in europe. Especially in rush hour the concentration is frequently overshot. Due to the breach of the EU policy (EU-Richtlinie 2008/50/EG) the Eu strives for proceedings against multiple german cities eg. Stuttgart and Darmstadt ,which would lead soon to monetary fines for the communes and the state. High values of air pollution affects also the human health in long and short-term, concerned are mainly residents who live or work close to such streets.

The overstepped air pollution values are highly spatial and temporal located, limited to small areas close to main streets and manly during rush hour. A good example is since the public discussion the well known Neckartor in Stuttgart. So a regional and dense grid of measuring stations is needed to generate a lot of high-quality input data to apply machine learning methods. The topographical position of Stuttgart within a valley basin leads in the combination of weather conditions to those overshot particulate matter concentration. In this context inversions, also called stationary temperature inversions, are common phenomenon in Stuttgart valley basin and are highly positive correlated to air pollution. So a closer look to those weather conditions and their parameters like wind direction and speed, precipitation and further meteorological parameters, would be interesting and part of the regression analysis.

**Introduction**

**Methods- which + why are chosen**

The following subsection give a short overview of three selected machine learning algorithms, their operation and performance.

**XGBoost – ensemble of Decision trees**

**Paper :** [**https://arxiv.org/abs/1603.02754**](https://arxiv.org/abs/1603.02754)Chen & Guestrin 2016

Buch . Géron 2019 Hands on ML

A very successful and widely used Gradient Boosting is XGBoost, an Extreme Gradient Boosting which is based on collections of decision trees. It was developed by Chen xxx year and is available by its corresponding python library. Benefits of XGBoost are:

* scalability in each processing steps/scenarios
* portability
* short running time

Like other Gradient Boosting methods it’s also possible to combine XGBoost with other cost functions (via the loss hyperparameter?-check) (Géron 2019, S. 210). Combinations with XGboost are often used successfully in ML competitions likewise the Netflix prize competition in 2009.

Similar to well-established decision tree models, likewise random forest which also make use of multiple decision-trees predictions, XGBoost is built up on boundary values, tree pruining and …

XGBoost is known for well performance on decision-tree-predictions based on large and sparse datasets. The small timesteps (hourly steps), which are used in this work, leads to large input datasets of up to 17.520 observations during the two-year study time (1year=8760 hours). Due to may defective input data with possible outliers, XGBoost seems to be a good choice for predictions of PM concentration.

* Good: highly scalable to larger datasets, optimized for extremely efficient computational performance, and handles sparse data with a novel approach.
* XGboost higher scalability (Chen & Guestrin 2016)

XGboost prevents overfitting via a regularized model (Chen & Guestrin 2016, S. 7). In the so called regularized objective the predictions from all (corresponding) leaves of each tree are calculated. These predictions are sum up (depending on their leaf weights) to the final prediction (Chen & Guestrin 2016, S. 2f)

In comparison random forest, XGBoost enables to ..

* Both: Column samling ( possible in random forest and in XGboost)
  + Subsampling columns : shorter running time and prevent overfitting (Chen & Guestrin 2016, S. 9)
* Boosting takes slower steps, making predictors sequentially instead of independently. ... By combining the advantages from both **random forest** and gradient boosting, **XGBoost** gave the a prediction error ten times lower **than** boosting or **random forest** in my case. (

https://liuyanguu.github.io/post/2018/07/09/extreme-gradient-boosting-xgboost-better-than-random-forest-or-gradient-boosting/#random-forest

Figure XGBoost.multi.trees.plot

**Softmax Regression**

Instead of the closely related logistic regression the Softmax Regression can be use with more than two features/variables or classes. Based on multinomial Regression it is also called Multinomial Logistic or Maximum Entropy Classifier. Due to the property of Softmax Regression for multi-class classifications it is also commonly used in advanced machine learning algorithms like network sense. In this work Softmax Regression is used for predictions on how much each single feature (wind, rain..) influences the target variable, the hourly PM concentration.

* 🡪 multiple calssi defined for different features 🡪 AIM: maximise predicions on all the clasisfiers , helps if input feature should belong to each of the classes in bild
  + 🡪 Extense log reg to softmax reg
* One of the benefits of a simple model like softmax is that we can visualize the weights for each of the classes, and see what it prefers – check (https://medium.com/@awjuliani/simple-softmax-in-python-tutorial-d6b4c4ed5c16)

