**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?**

**Introduction**

**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, 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.

**Methods- which + why are chosen**

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

In this study two types of *multiple decision-trees models* are described, the more suitable model is used in practise. The third model should be a Regression model and was chosen with respect that the model wont include all relevant parameters and the noise of the data wont make a adequate prediction of continuous values wont be possible. So the target was modify to predict if the measure value wil be over or under the limit of 200

compare the decision tree model with the regression model.

For the best results the feature importance and accuary of the prediction of a regularized linear model, Ridge Regression, and a Logistic Regression was previous validated.

Parameters which are not included in the sample and noise in the data is quite probable. In respect to the already large number of features, including sample columns with time shift, leads to a complex feature influence on the predicted target. So decided to use a fail-safe model

The last mentioned model was chosen with a view to compare the feature importance between a decision tree model and a regression model.

**Random Forest**

Decision tree models, like Random Forest, can be use for regression and also classification problems.

While growing trees uses a random subset of features to find the best feature of the subset to split the node. In comparison, other decision models use the best feature of the whole sample. Each tree grows parallel and separate from the other trees, in the end the decisions of each tree are used to generate the final decision. The random forest “..results in a greater tree diversity, which (once again) trades a higher bias for a lower variance, generally yielding an overall better model” (Géron 2019 , p. 199).

The best split of a branch can be determined by Gini-index or cross entropy (r2d3. website). Of consequence/advantageous is that the properties on which a decision is based, is comprehensible (white box) and less complex than in other machine learning methods like neural networks.

Both decision tree models use column sampling to achieve a shorter run time and to prevent overfitting (Chen & Guestrin 2016, S. 9).

**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, 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 a large input dataset of up to xxxxxx 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 NO2 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)

XGBoost combines … from random forest and gradient boosting, resulting in lower prediction error. The Gradient Boosting takes smaller learning rate (slower teps) by doing the predictions sequentially.

* 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)*

**Logistic Regression**

The model calculates a binary prediction of under- and overshut of nitrogen dioxide boundary of 200 μg/m³, so no continuous values are generated like in the decision tree model and the Rdige Regression.

This boundary value is a short term exposure value, the EU policy allows, beneath further exeptions, a 18 time overstep of this limit during a year. To simplify the calculations just the short term exposure value is used (<https://www.umweltbundesamt.de/themen/luft/luftschadstoffe-im-ueberblick/stickstoffoxide/stickstoffdioxid-gesundheitliche-bedeutung-von#grenzwerte-stickstoffdioxid>

If the estimated probability is greater than 50%, then the model predicts that the instance belongs to that class (called the positive class, labeled “1”), or else it predicts that it does no

Binary classification

**S.144**

**Hyperparameter tuning**

The regularization strength of the logistic regression model is controlled by the parameter C. C is the inverse of the regularization parameter, so a lower value of C induces a stronger regularization. Via the regularization term in the cost function the weights can be minimized. To find the best value for C a cross-validation estimator already implemented in the sklearn LogisticRegressionCV() was chosen. The best predictions could be archived by a low regularization of C = 2.

To avoid overfitting of the Logistic Regression model some further parameters in sklearn LogisticRegression() function can be added. It’s useful to regularize the model by Ride Regression (penalties = l2) in combination with a fast and sufficient solver for large datasets, called sag. In general regularization techniques try to find the best balance between bias and variance of a dataset. Ridge Regression is usually applied for data with multicollinearity; high correlation between multiple features. The *normalization of the cost function?* Can also remove noise from the data and minimize overfitting. (https://www.bogotobogo.com/python/scikit-learn/scikit-learn\_logistic\_regression.php)

Another possibility to improve the validity of the model is to standardize the input features, to archive a unit variance of all features. Regularized models react sensitive to different scales of the input features. By Standardization the mean value is subtracted and divided by the standard deviation. To use the same transformation later on the testing set the StandardScaler is firstly used on the training dataset and with respect to use the fatures of the testing dataset.

**Standardization**

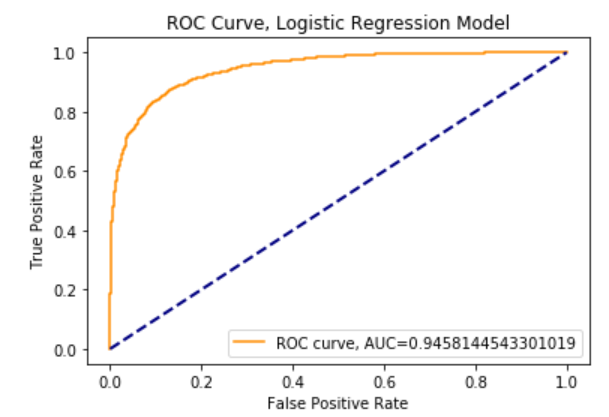
The preprocessing module further provides a utility class **[StandardScaler](https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html" \l "sklearn.preprocessing.StandardScaler" \o "sklearn.preprocessing.StandardScaler)** that implements the Transformer API to compute the mean and standard deviation on a training set so as to be able to later reapply the same transformation on the testing set. This class is hence suitable for use in the early steps of a

https://scikit-learn.org/stable/modules/preprocessing.html#preprocessing-scaler

Standarize feature values

If you scale each feature to have the same standard deviation (i.e. divide each column of the model matrix by its standard deviation), then the magnitude (absolute value) of the coefficients is a direct measure of the importance of the corresponding features. See Schielzeth 2010 Methods in Ecology and Evolution "Simple means to improve the interpretability of regression coefficients" <https://doi.org/10.1111/j.2041-210X.2010.00012.x>.

(stack overflow: <https://stats.stackexchange.com/questions/423469/how-to-measure-the-impact-influence-of-a-feature-y-on-logistic-regression-mode> )



**PCA**  
The Principal Component Algorithm is a common method to prevent overfitting, extract noise and also to minimize large dataset by removing nonrelevant information. It keeps the relevant data and declassify the rest of the dataset as noise. The input features, except the target, are summarized and reduced to newly transformed principal components.

To find the most suitable number of components we set, instead of a concrete number, a ratio of variance of 95 % (p. 225). Those preserved components were used as new input features in the models of the Logistic- and Ridge Regression. Before the transformation into new feature space the training and testing sets were standardized, the target variables of both sets remained unchanged.

**Ridge Regression**

* For multiple regression data with multicollinearity, variances are large-> so may far from true value
* Regularized linear model
* To reduce standard errors -> Adding degree of bias to regression estimates
* **Multikollinearität** ist ein Problem der [Regressionsanalyse](https://de.wikipedia.org/wiki/Regressionsanalyse) und liegt vor, wenn zwei oder mehr erklärende Variablen eine sehr starke [Korrelation](https://de.wikipedia.org/wiki/Korrelation) miteinander haben. Zum einen wird mit zunehmender Multikollinearität das Verfahren zur Schätzung der Regressionskoeffizienten instabil und Aussagen zur [Schätzung](https://de.wikipedia.org/wiki/Sch%C3%A4tzung) der Regressionskoeffizienten zunehmend ungenau (wiki)

**Results**

For validation of the model predictions Accuracy, Precision, Recall and F1-Score were used as validation parameters. The combination of those four parameters is useful due to their limited explanatory power by individual view.

All three models show similar parameter values with a F1 Score always around 90 %, this can be interpreted as a well performance of all three models. The parameter of F1 Score is more robust against an unequal amount of false positive to false negative that’s why it is here used for a rough estimation.

Tab xxx: Predictions of the measurement category, above or lower than the NO2 limit value of 40 u/m³.

|  |  |  |  |
| --- | --- | --- | --- |
|  | XGBoost | Logistic Regression | Ridge Regression |
| Accuracy | 88.8 % | 87.5 % | 84.2 % |
| Precision | 90.2 % | 88.5 % | 86.3 % |
| Recall | 93.2 % | 93.2 % | 90.5 % |
| F1 Score | 91.6 % | 90.8 % | 88.3 % |

The validation parameters should be seen in combination with their weights of the input features. The feature importance of the logistic regression and the XGBoost model show both a strong overfitting through the high relevance of especially the first feature.

It is not possible directly to compare coefficients from different models, but between variations of a single model e.g. by changing one hyperparameter. In this case a dimensionality reduction was applied at the logistic regression model. The used PCA algorithm leads to a reduction from 287 to 91 features, by preserving a ratio of variance of 95 %. Performing the logistic regression after dimensionality reduction improved the principal components to coefficient values lower or equal to 0.90. In comparison without PCA the most important feature had a coefficient of 3.4 with a wide gap to less important features. The values of the validation parameters decreased marginally caused by the limitation of the features.

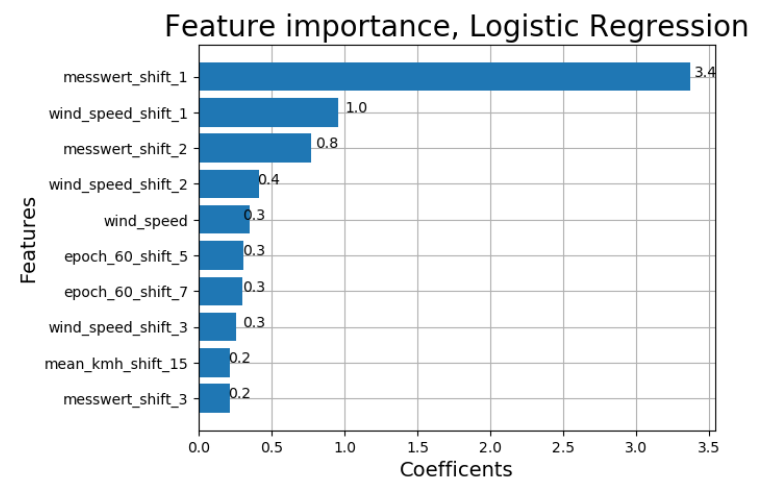


Fig : Coefficients of the ten most useful features of the logistic regressin model without using a PCA.

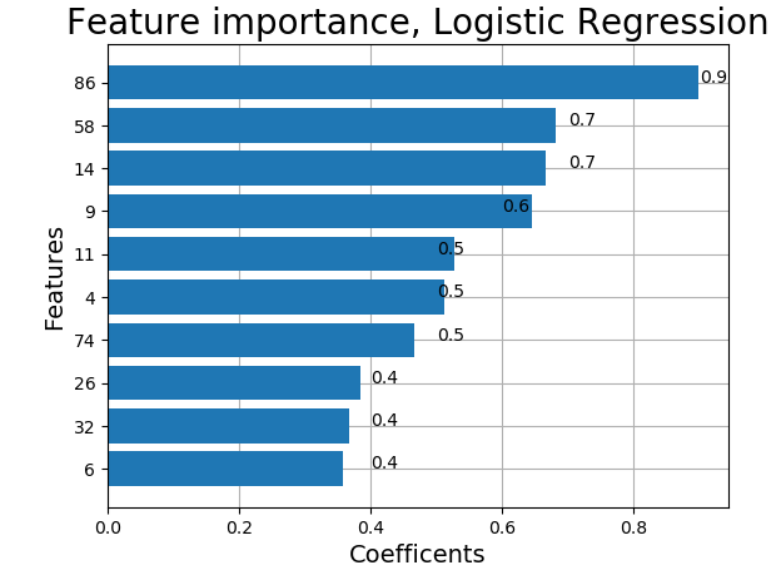
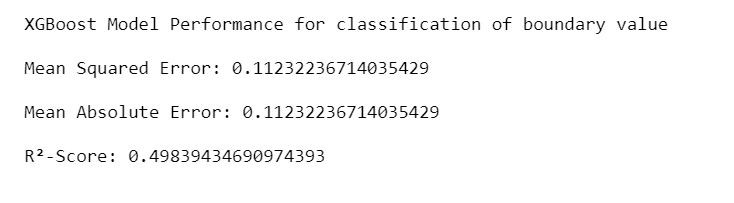
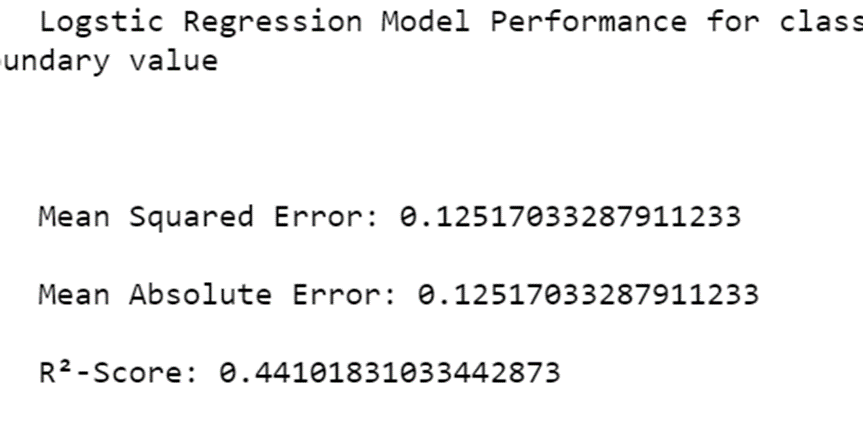


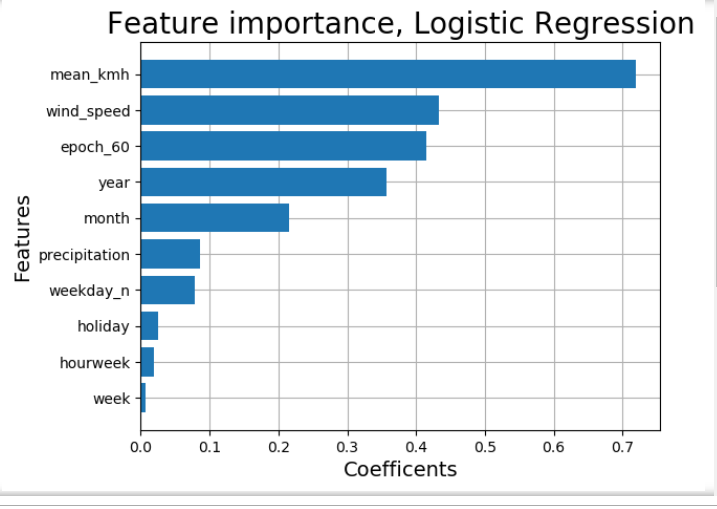
Fig : Coefficients of the ten most useful features of the logistic regressin model after using a PCA. Except the PCA all other hyperparameters and the dataset remained unchanged.

Ridge Reg= 0.95 ration

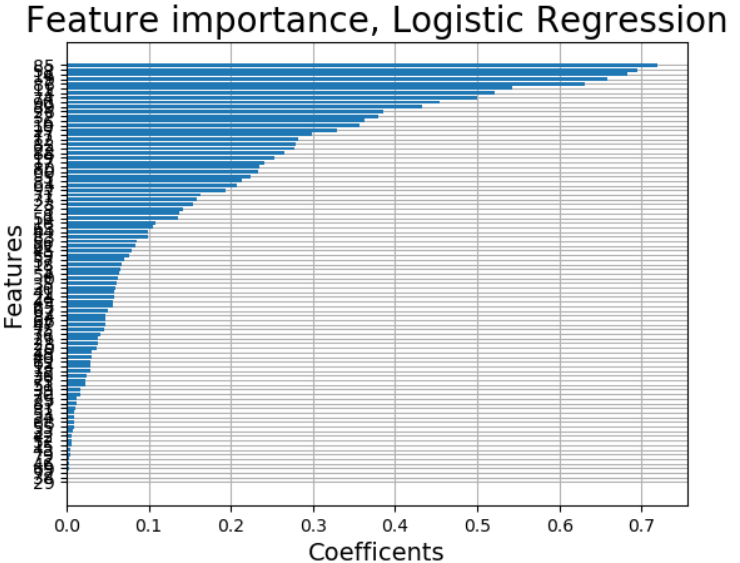
Tab xxx: Model predictions of hourly NO2 contentration.

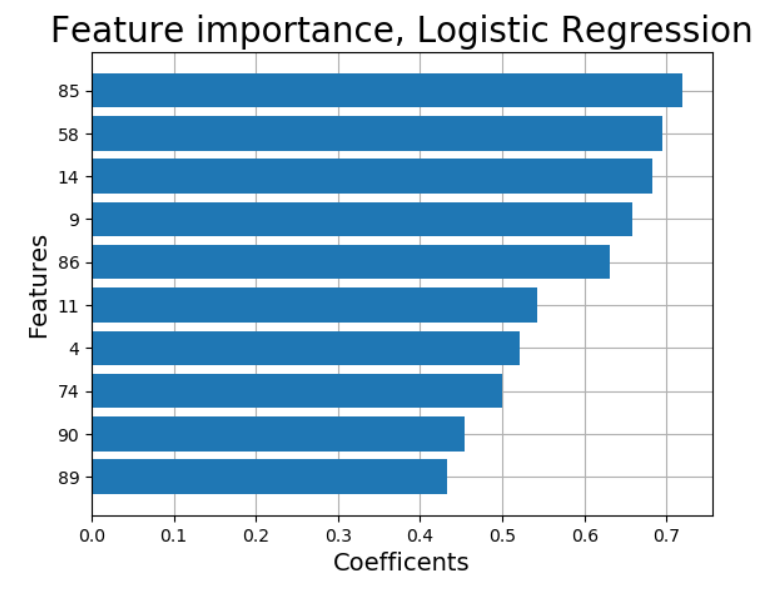
|  |  |  |
| --- | --- | --- |
|  | XGBoost | Ridge Regression |
| MSE | 32.2 | 164.3 |
| MAE | 4.1 | 9.9 |
| R²-Score | 0.82 | 0.7 |

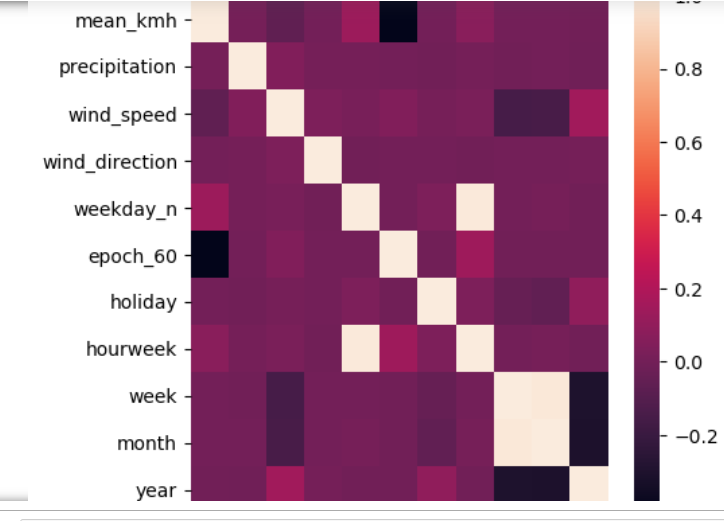


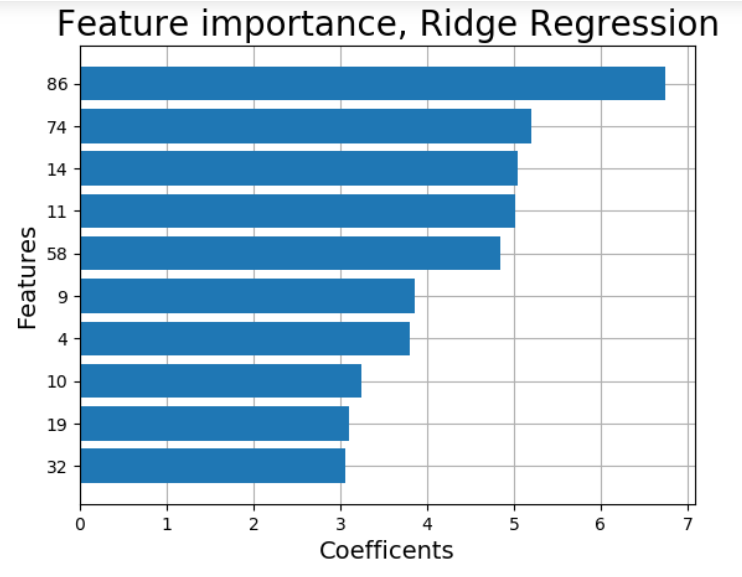
**PCA = 95 % ratio of variance that reserves**

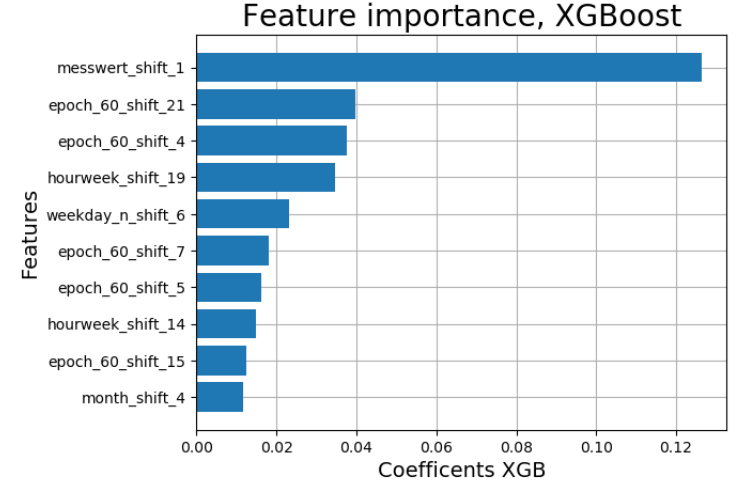
**2 Plots (all features, and top 10 features)**

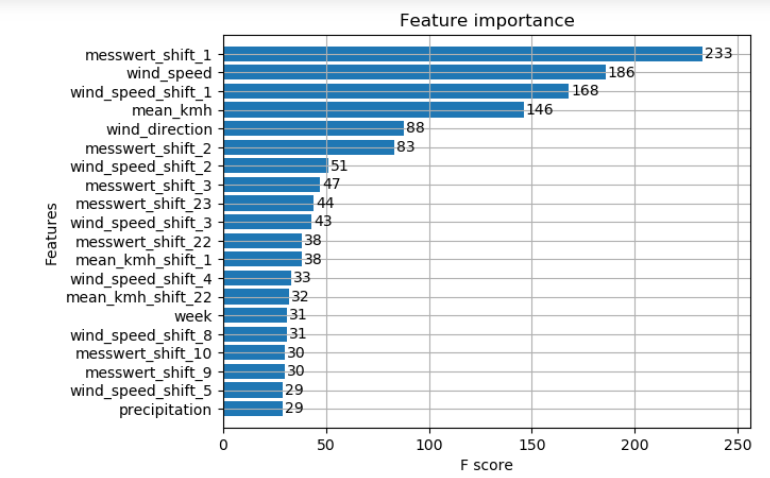


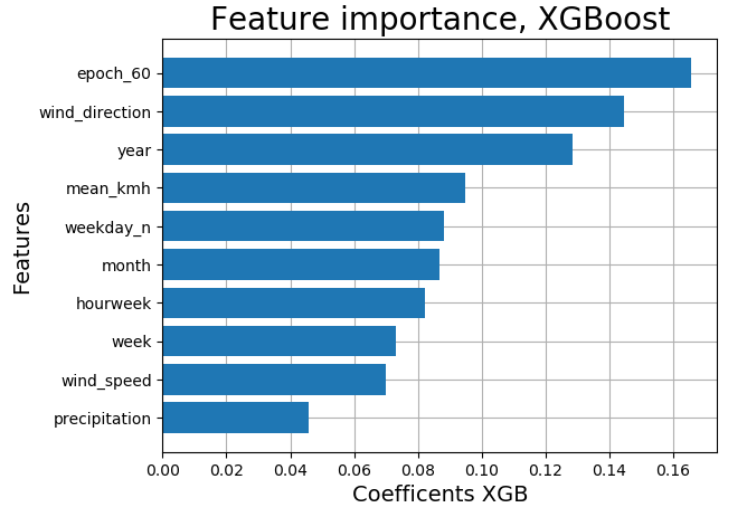


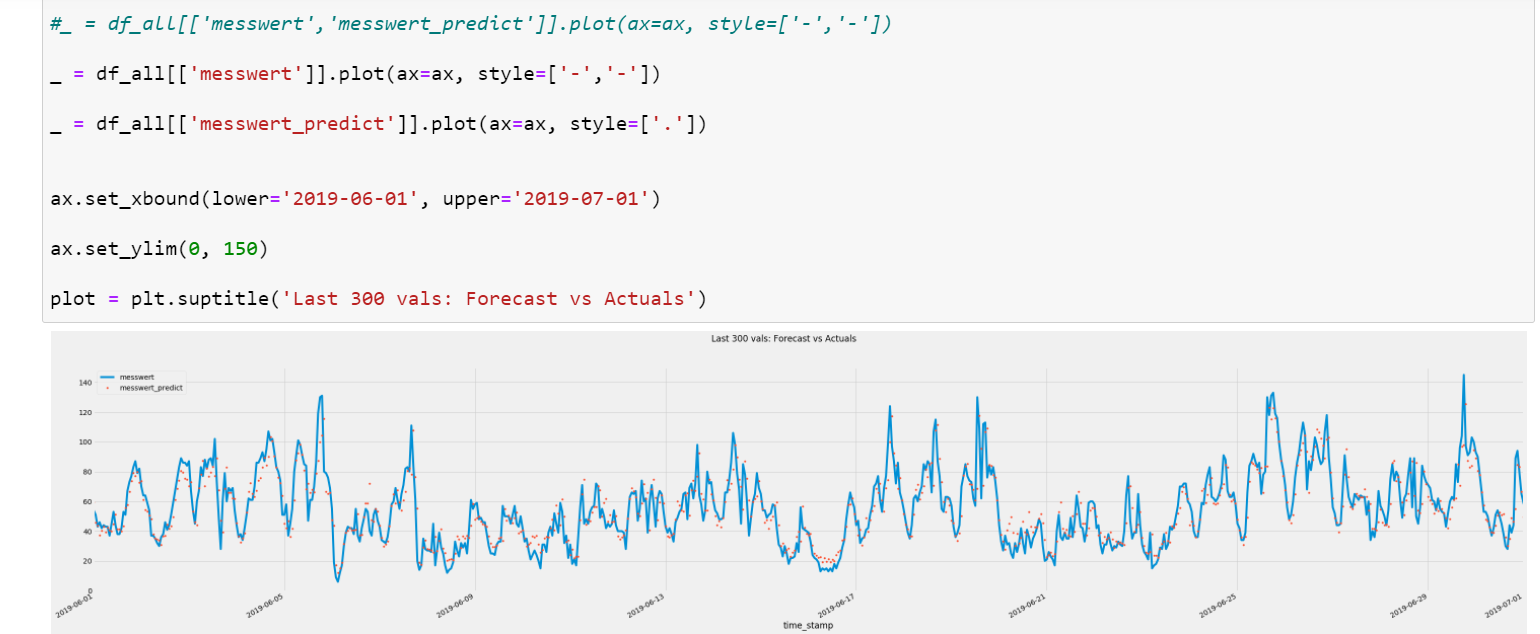


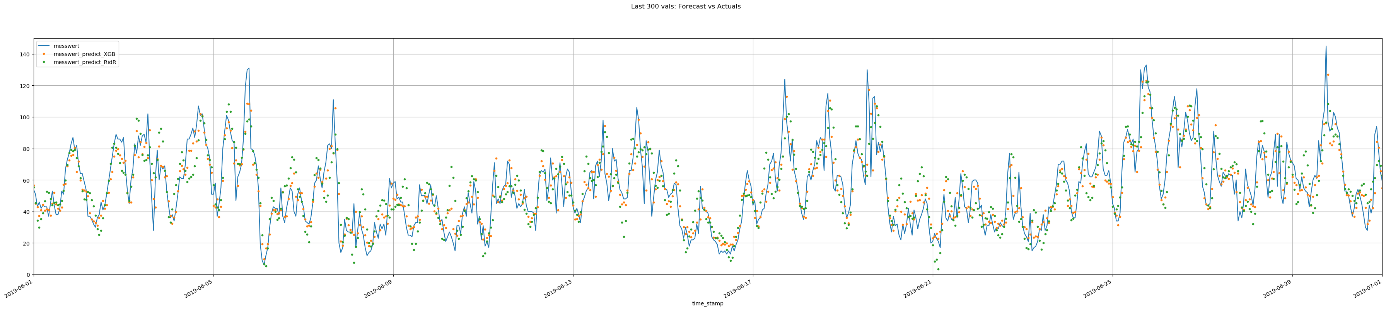


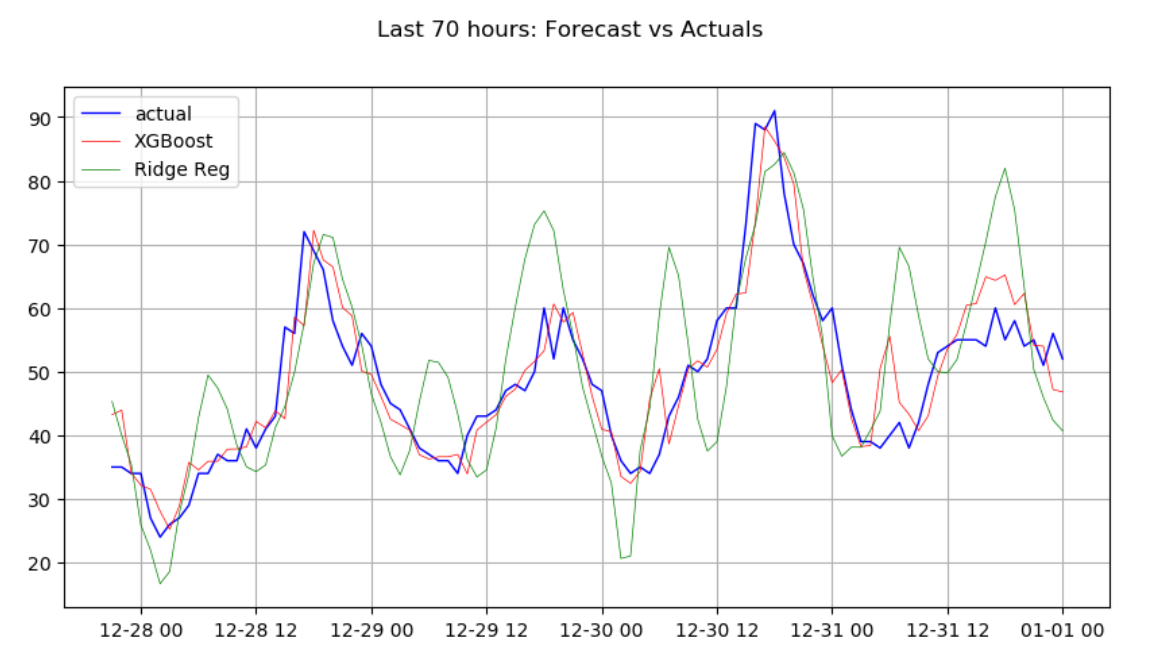


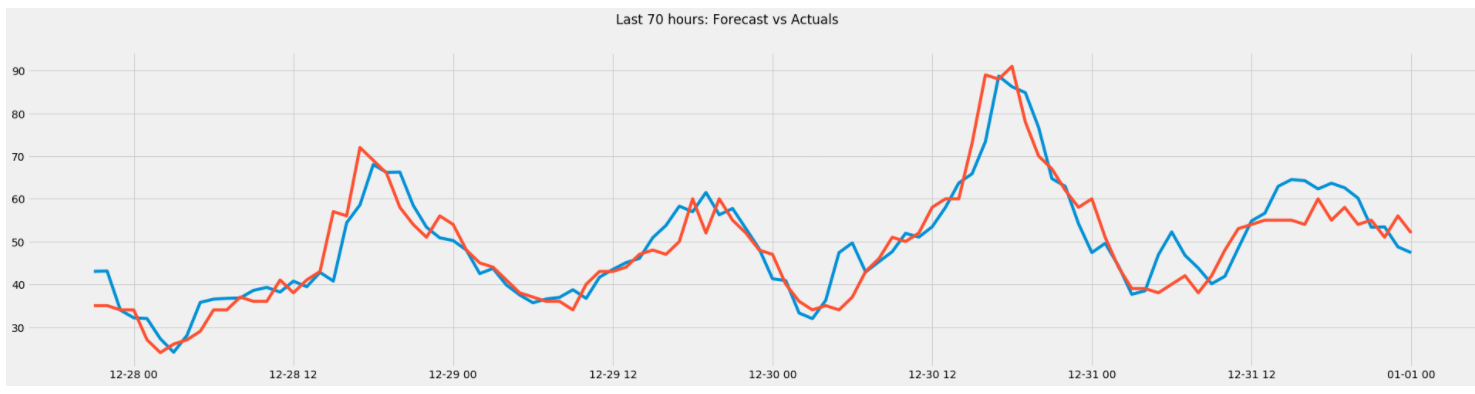




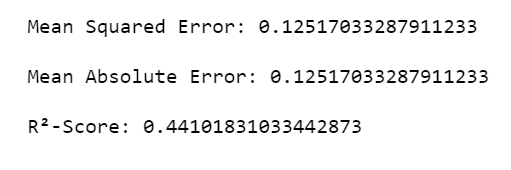
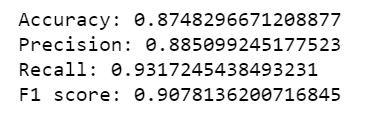


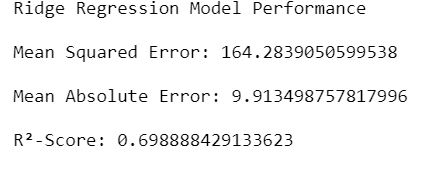






Logistic REG: (40 u/m³, + Wind.cols, standard+pca= None = best solution)



XGBoost : continuous NO2 values

