Projektarbeit

Digethic–Digital Business School

Certified Data Scientist

Forecasting the emission of PM2.5 pollution with Machine Learning

(verfasst im Rahmen der EN ISO / IEC

17024-Zertifizierungsprüfung)

Prepared by: Hamid Motamedi
Instructor: Alexandra Arkhipova
Date: 08.06.2022

Eidesstattliche Erklärung

Hiermit versichere ich an Eides statt, dass ich die vorliegende Projektarbeit eigenständig und ohne Mitwirkung Dritter angefertigt habe. Quellenangaben wurden entsprechend als solche gekennzeichnet.

Ort,

Unterschrift

Contents

List of Figures

List of Tables

1 Introduction

Air quality forecasting is rather complicated and dominated by meteorological conditions, and emission inventory [Ma, J. *et al* 2020]. The complex mixtures of local emission sources and regional transportation of air pollutants make accurate PM2.5 prediction a very challenging yet crucial task, especially under high pollution conditions [F.j Chang *et al.* 2020]. Therefore, there are still great uncertainties in the current forecast of emission quality, which does not meet the requirements of current air pollution control. Accurate air quality forecasting is important both for responding to severe air pollution and for self-protection of human health. PM2.5 describes fine inhalable particles with diameters that are generally 2.5 micrometers or smaller. Air quality monitors measure concentrations of PM [EPA]. Understanding of environmental data and its analysis contributes to the preservation of biodiversity, which is an important cause of climate change. Assessing the severity of PM2.5 pollution requires a set of statistical measures as well as forecasting with machine learning. Because particle pollution affects air quality and has a major socioeconomic effect on human lives. This work deals with the analysis of PM2.5 as a target from a time series dataset converted to the supervised machine learning modeling system. Finally, predictions are attained in the Python programming language via Jupyter notebook using their libraries and functions. In this regard, accuracy of different models will be calculated, and the results will be discussed. The efficiency of different models will be compared and explained. Ultimately, the implementation is uploaded via VS Code to a GitHub repository for recovery and made available.

2 Data and Methods

2.1 Data description

This hourly data set contains the PM2.5 data of the US Embassy in Beijing. Meanwhile, meteorological data from Beijing Capital International Airport is also included. The dataset was downloaded from the UCI machine learning repository. The PM2.5 data is a time series dataset because of its seasonal course (see in figure

No: Row number
Year: Year of data in this row
Month: Month of data in this row
Day: Day of data in this row
Hour: Hour of data in this row
PM2.5: Concentration of Pollution
DEWP: Dew Point
TEMP: Temperature
PRES: Pressure
Cbwd: Combined wind direction
Iws: Cumulated wind speed (m/s)
Is: Cumulated hours of snow
Ir: Cumulated hours of rain

Table 1: Feature description.

3). Time series forecasting can be framed as a supervised learning problem. This re-framing of time series data enables access to the problem's standard linear and nonlinear machine learning algorithms [Brownlee 2020]. The dataset used in this project is a table with 43824 instances in 13 columns and a target of some NAN-Value in "pm2.5" (see figure 1). The acronyms in the title of the table are as described in the table 1.

	No	year	month	day	hour	pm2.5	DEWP	TEMP	PRES	cbwd	Iws	Is	Ir
0	1	2010	1	1	0	NaN	-21	-11.0	1021.0	NW	1.79	0	0
1	2	2010	1	1	1	NaN	-21	-12.0	1020.0	NW	4.92	0	0
2	3	2010	1	1	2	NaN	-21	-11.0	1019.0	NW	6.71	0	0
3	4	2010	1	1	3	NaN	-21	-14.0	1019.0	NW	9.84	0	0
4	5	2010	1	1	4	NaN	-20	-12.0	1018.0	NW	12.97	0	0

Figure 1: Head of Raw and columns.

2.2 Data analysis

The categorical feature "windDirection" was first separated by *drop()* to have an image of different features. Correlations can be seen by *heatmap* and *seaborn* pairplot plotting in figures 2 and figures 4, respectively. In order to have an overview

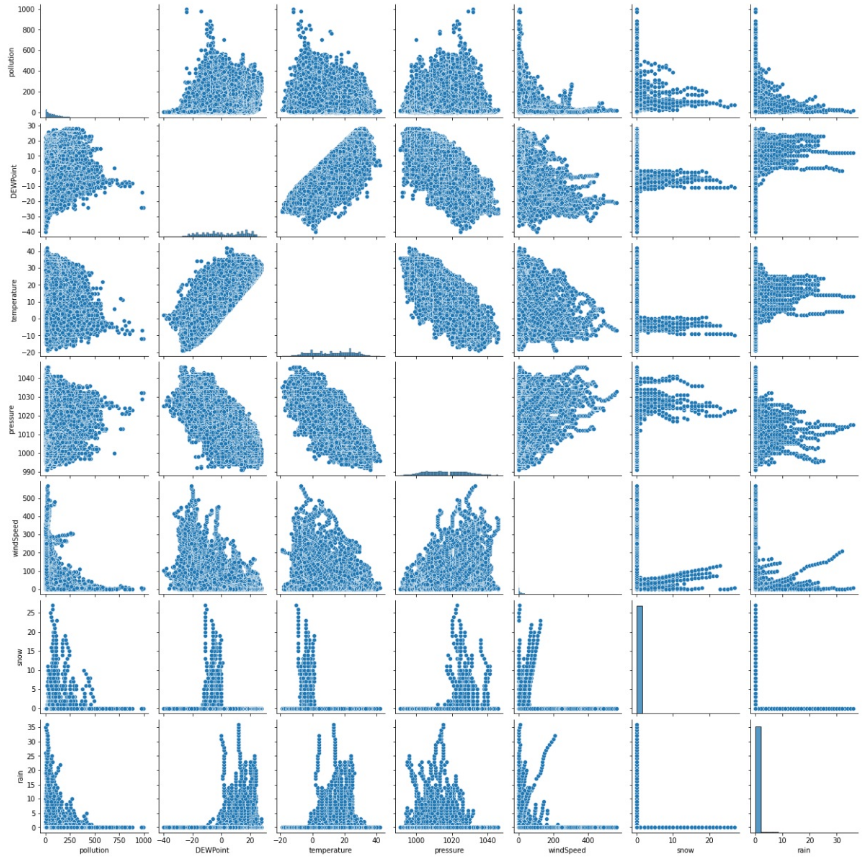

Figure 2: Seaborn visual pairplot Correlation of features.

of data in the whole time record, the target is illustrated in the figure 3. This seasonal trend can more or less be seen in all other environmental features of the data as well. Figure 4 shows that there is a stronger negative correlation between "windSpeed" and pollution as well as a positive correlation between "DewPoint" and pollution. There is also a positive correlation between temperature and DewPoint as well as a negative correlation between pressure and DewPoint, which causes an indirect relation to pollution. There is also a natural negative correlation between temperature and pressure, which in turn impacts the target during the windSpeed as a negatively correlated variable.

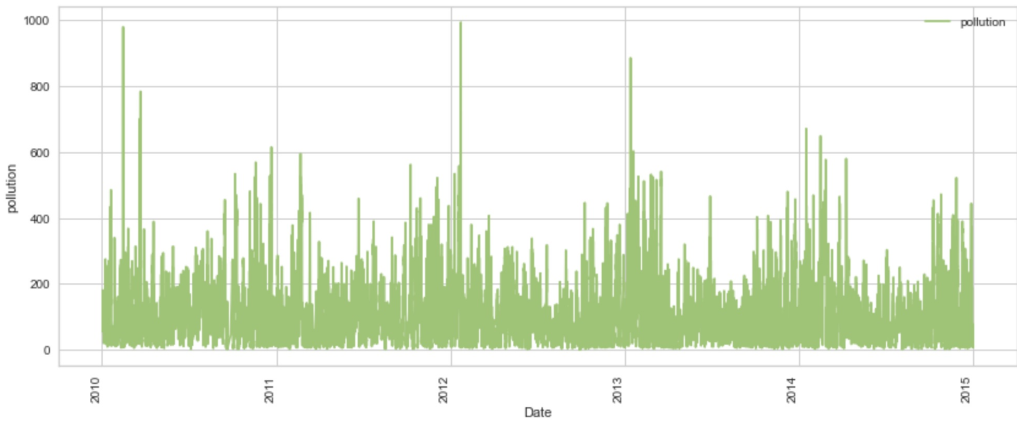

Figure 3: Pollution changes seasonally over a five-year period.

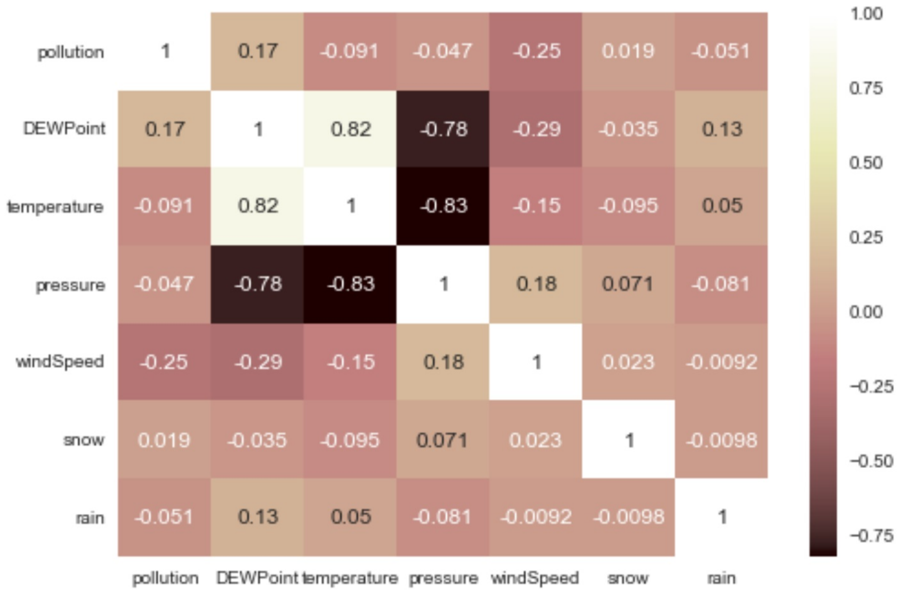

Figure 4: An illustration of positive and negative correlations in features.

2.3 Cleaning & preprocessing

2.3.1 NAN-Values and indexing of features

First, column "No" had to be deleted via *drop()* function so that instances are not numbered in parallel. Following that, the columns for the year, month, day, and time should be merged into a column called "Date" using the *to _datetime()* function. Then this column will be added to other columns using *columns.tolist()* and we will drop the four other previous useless columns. It is important to deal with NAN-Value in the target using the function *notna()*. The function *set_index ('Date')* is used to achieve the Date Index and new feature names as shown in 5.

	pollution	DEWPoint	temperature	pressure	windDirection	windSpeed	snow	rain
Date								
2010-01-02 00:00:00	129.0	-16	-4.0	1020.0	SE	1.79	0	0
2010-01-02 01:00:00	148.0	-15	-4.0	1020.0	SE	2.68	0	0
2010-01-02 02:00:00	159.0	-11	-5.0	1021.0	SE	3.57	0	0
2010-01-02 03:00:00	181.0	-7	-5.0	1022.0	SE	5.36	1	0
2010-01-02 04:00:00	138.0	-7	-5.0	1022.0	SE	6.25	2	0

Figure 5: Data with a date index.

2.3.2 Encoding of categorical feature

Encoding is widely used in machine learning. However, encoder-decoder has the disadvantage of losing information [K. cho *et al.* 2014]. Encoding is a required preprocessing step when working with categorical data for machine learning algorithms [Brownlee 2020]. In order for all features of data to be included in the calculations, the categorical columns should be converted to numeric variables. As shown in the figure 6, encoding takes over this task in this work for "windDirection". The function *LabelEncoder()* can be used from library *sklearn* for this term.

```
In [4]:  values = df1.values
         encoder = LabelEncoder()
         values[:,4] = encoder.fit_transform(values[:,4])
         values = values.astype('float32')
         values[:,4]

Out[4]: array([2., 2., 2., ..., 1., 1., 1.], dtype=float32)
```

Figure 6: Encoding of one categorical feature in the fifth column.

2.3.3 Feature Scaling

Many machine learning algorithms perform better when numerical input variables are scaled to a standard range. This includes algorithms that use a weighted sum of the input, like linear regression, and algorithms that use distance measures, like k-nearest neighbors [Brownlee 2020]. The general mathematical formula from *sklearn.preprocessing* is shown below:

$$X_std = (X - X.min(axis = 0))/(X.max(axis = 0) - X.min(axis = 0))$$

$$X_scaled = X_std * (max - min) + min$$

In this work, a function called *MinMaxScaler* is used to scale the data range. Then a dataframe had to be created for generating the lag feature using the Pandas library from scaled data, which was displayed in figure 7.

	pollution	DEWPoint	temperature	pressure	windDirection	windSpeed	snow	rain
0	0.129779	0.352941	0.245902	0.527273	0.666667	0.002372	0.000000	0.0
1	0.148893	0.367647	0.245902	0.527273	0.666667	0.003947	0.000000	0.0
2	0.159960	0.426471	0.229508	0.545454	0.666667	0.005522	0.000000	0.0
3	0.182093	0.485294	0.229508	0.563637	0.666667	0.008690	0.037037	0.0
4	0.138833	0.485294	0.229508	0.563637	0.666667	0.010265	0.074074	0.0

Figure 7: Dataframe from scaled data.

2.3.4 Converting time series to supervised Learning

The use of prior time steps to predict the next time step is called the sliding window method. In short, it may be called the window method in some literature. In statistics and time series analysis, this is called a lag method. The number of previous time steps is called the window width or size of the lag [Brownlee 2020]. Visual example of the *lagplot* is shown in the figure 8. After that, it should be checked if the dataframe has some NAN-value, which will be cleaned with a function *dropna(inplace=True)*. Because of the improved output from the *pandas* library, this work used a lag size of five, as shown in the code block below in figure for a significant result 9.

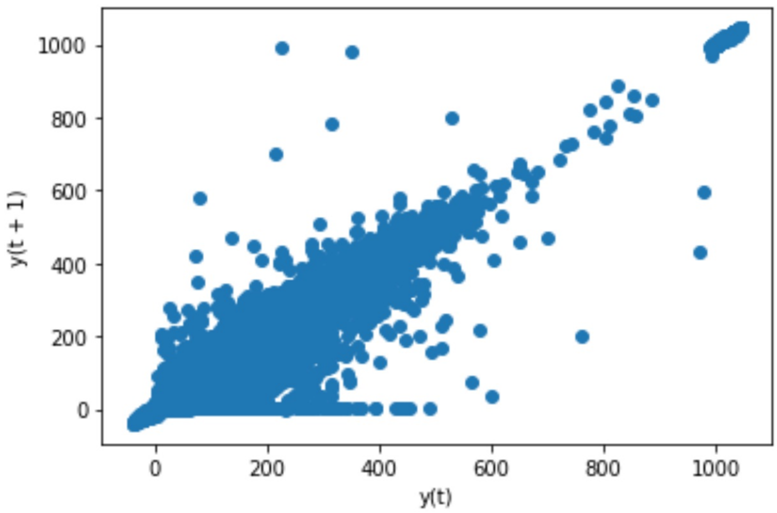

Figure 8: Lag Plot shows relation from t to t+1.

```
lookback = 5
for i in newdf.columns:
    for j in range(lookback):
        newdf[str(i)+str(j+1)] = newdf[i].diff(j+1)
```

Figure 9: Sliding Window.

2.4 Methodology

2.4.1 Data Splitting for Training and Testing

Train-test split is used to estimate the performance of machine learning algorithms that are suitable for prediction-based algorithms and applications. In the case of this work, a code block was used to create the training data and test data, which can be seen in figure 10. The trainset is used to fit the model, and the statistics of the trainset are known. The second set is called the testset. This set is solely used for predictions [GFG]. In this experiment, train and test data were split into

```
split_index = int(newdf.shape[0]*0.7) # the index at which to split df into train and test

# ...train
X_train = newdf.drop('pollution',axis=1).values[:split_index]
y_train = newdf['pollution'].values[:split_index]

# ...test
X_test = newdf.drop('pollution',axis=1).values[split_index:] # original is split_index:-1
y_test = newdf['pollution'].values[split_index:] # original is split_index:-1
```

Figure 10: Train-Test splitting.

70% and 30% of the total dataset, respectively, and then y_test was predicted using predict x_test after fitting x_train and y_train, and visualized according to this split in different modeling formats.

2.4.2 Splitting and reset date index

A code in figure 11 was used to represent the models with the useful time order for prediction, which is significant for visualization of the predicted models.

```
x = df[split_index:].reset_index()

date_diff = len(x)-len(y_test)
x = x[date_diff:]
```

Figure 11: Splitting the date index for the purpose of prediction visualization.

2.4.3 Residuals plots

In the context of regression models, residuals are the difference between the observed value of the target variable (y) and the predicted value ($\hat{y}$), i.e. the error of the prediction [Residuals plot 2016-2019]. The residuals plot shows the difference between the residuals on the vertical axis and the dependent variable on the horizontal axis. It contributes to identify areas within the target, that may be more or less prone to error (see in figure 12).

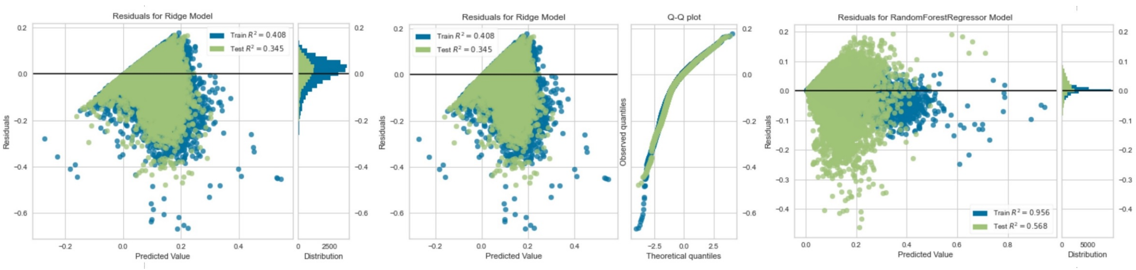

Figure 12: Ridge Models; Residuals for Ridge and Random forest models (Left: Histogram of Distribution, center: QQ-Plot, right: Residuals for "Random forest Regressor" model).

A common use of the residuals plot is to analyze the variance of the error of the regressor. A random distribution around the horizontal axis means a linear regression model is usually appropriate for the data. In this case, there is not complete fairly and uniform random. The histogram does not show a exact normal distribution around zero too. Quantiles of the QQ-plot do not follow a straight line, which is what a normal distribution has to be. From the data distribution it can be seen that the data narrowly supports a linear modeling problem.

2.4.4 k-Nearest neighbors (kNN)

k-nearest neighbors algorithm is a non-parametric method for estimating probability density functions [Ertel, W. *et al.* 2016]. kNN can provide better results than existing statistical approaches. That is, when the problem has a lot of input data with a lot of variables. In this case, a range between 1 and 100 neighbors was used for the calculation of Mean Squared Error. The accuracy of the computing shows that this method is most efficient at choosing around 18 for k in prediction (see in figure 13).

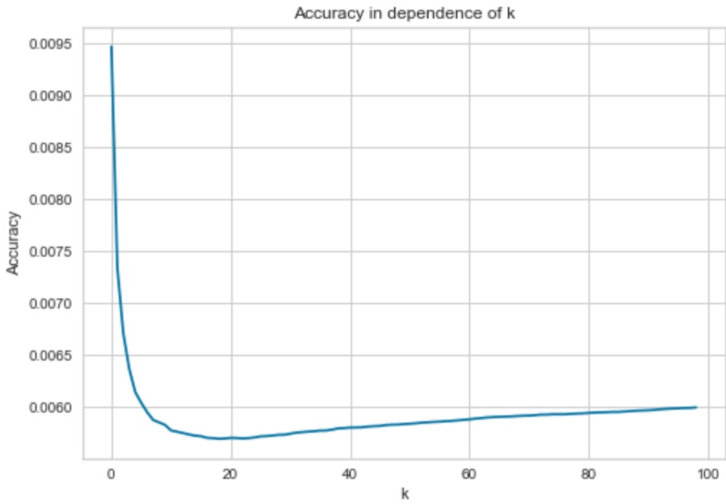

Figure 13: kNN processing accuracy for the 100 k range in horizontal axis. Unit of error measurement at the vertical axis is Mean Squared Error (MSE).

3 Results

The six models used in this work are as follows: Linear Regression, Decision Tree, Random Forest, Support Vector Machine, Gradient Boosting, and k-Nearest-Neighbor. These are shown in figure 14 and figure 15. The models are created with the *sklearn* libraries via Python.

The average values achieved by each method with 70% of the chosen data elements used for the training set and the remaining 30% used for the testing set. Mean Squared Error (MSE), Root Mean Squared Error (RMSE), and Mean Absolute Error (MAE) are most useful when the dataset contains outliers or unexpected values. However, RMSE has been widely adopted to standardize the units of measures of MSE. MSE is more sensitive to outliers than MAE [Chicco, D. *et al.* 2021]. The output of this error calculation for each model in this work has been shown in Table 2. Using two libraries called *plotly.graph_objs* and *plotly.offline*, all the calculated models for the prediction of the pollution were presented, so that for the entire course of

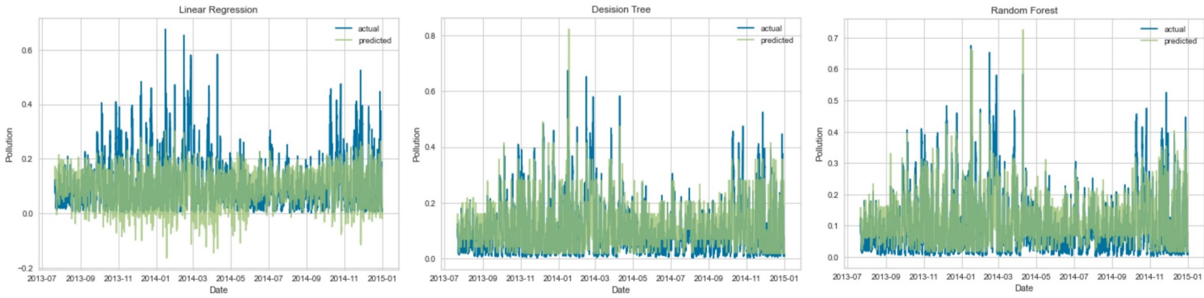

Figure 14: Predicted and actual pollution for the Date axis. Models are Linear Regression, Decision Tree and Random Forest.

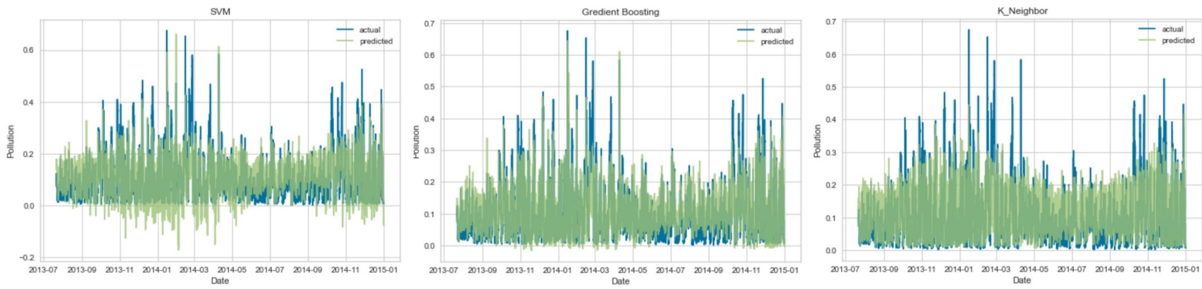

Figure 15: Predicted and actual pollution for the Date axis. Models are Support Vector Machine, Gradient Boosting and kNN for the Date axis.

Table 2: Results of MSE, RMSE and MAE

Model	MSE	RMSE	MAE
LinearRegression	0.0054	0.0737	0.0523
Decision Tree	0.0049	0.0697	0.0485
Random Forest	0.0043	0.0659	0.0458
SVM	0.0052	0.0718	0.0557
Gradint Boosting	0.0038	0.0613	0.0419
kNN	0.0060	0.0772	0.0546

the date variable, a precise visualization of the models in relation to the target was created. This can be seen in the figure 16.

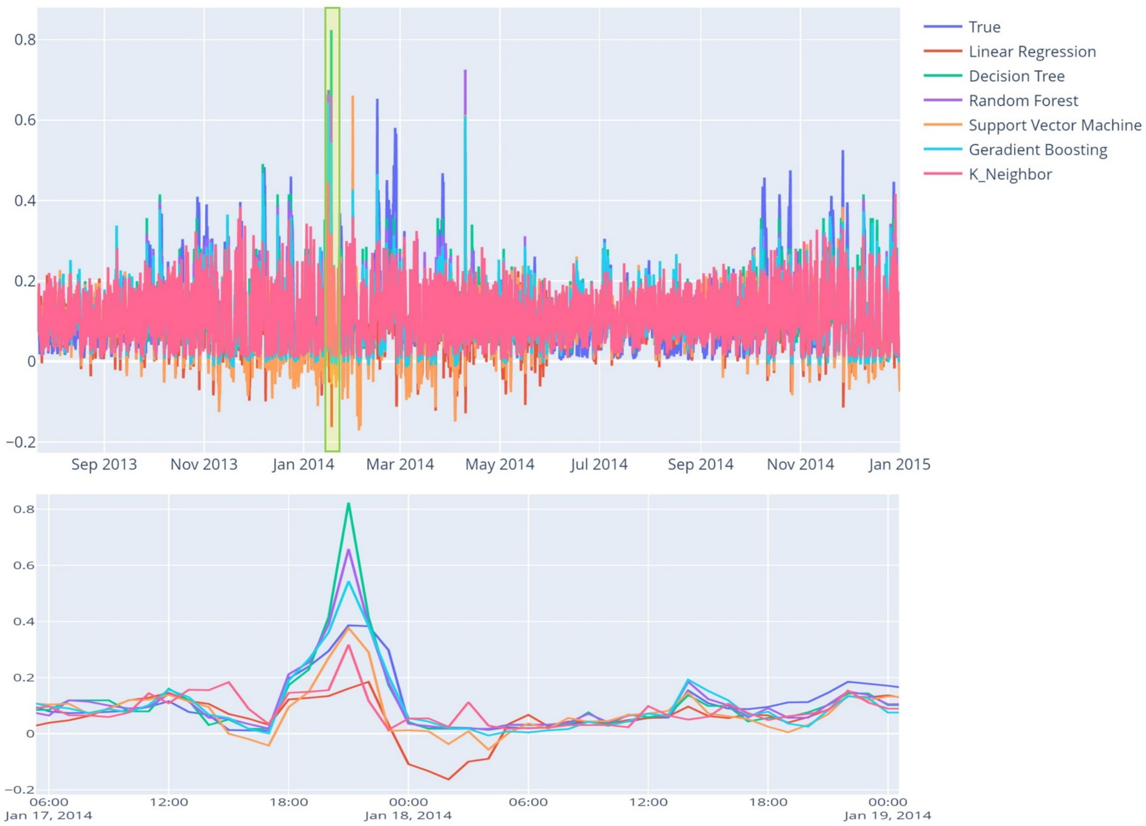

Figure 16: Maximum recorded rate of pollution and its predictive models. The entire date variable with all models, including the target, is represented on the upper graphic plot. A narrow yellow highlight includes the highest level of the measured target variable. This has been enlarged in the lower part of the graphic plot so that the lines of all variables with their peaks can be seen. The color of the lines represents the respective models to the top right.

.

4 Conclusion

This project attempted to test a time series of PM2.5 data for possible prediction models based on conversion to a supervised learning model. The models used are Linear Regression, Decision Tree, Random Forest, SVM, Gradient Boosting, and kNN. Ridge models and kNN do not refer to a suitable linear modeling but rather Gradient Boosting and random forest make more favorable error values. This can generally indicate better modeling in nonlinear or tree-based models. The predictive error was calculated for all of these methods in table 2. An ideal value for MSE, RMSE, and MAE is 0.0, which means that most of the predicted models could make sense for the target variable. As shown in the figure 14 and 15, the linear regression and SVM do not demonstrate an expected form with their measured accuracy from the table 2. All other models shown in figures have a better visual fit to the target variable, while Gradient Boosting and random forest show the best error quote for three measured errors too. There is the highest peak of pollution in figure 16, where on January 18 at around 21:00 almost all models more or less predicted a peak for this hour, whereby it was also expected based on the course of the target variable. It is disadvantageous just in the case of linear regression for this peak hour.

These methods could conclude that time series problems are better solved with non-linear methods in supervised learning. It recommends the LSTM and SARIMA models for future work with time series data. Because it is important to be able to predict the alarm time of a risky pollution rate for sensitive groups during peak hours.

5 References

[Brownlee 2020] Brownlee, J. (2020, August 14). Time Series forecasting as supervised learning. Machine Learning Mastery. Retrieved May 24, 2022, from online available: https://machinelearningmastery.com/time-series-forecasting-supervised-learning/

[Chicco, D. *et al.* 2021] Chicco, D., Warrens, M. J., & Jurman, G. (2021). The coefficient of determination R-squared is more informative than SMAPE, MAE, MAPE, MSE and RMSE in regression analysis evaluation. PeerJ Computer Science, 7, e623.

[Ertel, W. *et al.* 2016] Ertel, W., & Black, N. T. (2016). Grundkurs Künstliche Intelligenz. Wiesbaden: Springer Fachmedien Wiesbaden.

[EPA] Environmental Protection Agency. (n.d.). Particulate Matter (PM2.5) Trends. EPA. Retrieved May 24, 2022, from online available: https://www.epa.gov/air-trends/particulate-matter-pm25-trends

[F.j Chang *et al.* 2020] Fi John Chang, Li-Chiu Chang, Che Chia Kang, Yi Shin Wang, Angela Huang, Explore spatio temporal PM2.5 features in northern Taiwan using machine learning techniques https://doi.org/10.1016/j.scitotenv.2020.139656

[GFG] How to split a dataset into train and test sets using Python. GeeksforGeeks. (2022, May 25). Retrieved May 27, 2022, from online available: https://www.geeksforgeeks.org/how-to-split-a-dataset-into-train-and-test-sets-using-python/

[K. cho *et al.* 2014] K. Cho, B. Van Merriënboer, D. Bahdanau, and Y. Bengio, "On the properties of neural machine translation: encoder-decoder approaches," 2014, https://arxiv.org/abs/1409.1259.

[Ma, J. *et al* 2020] Ma, J., Yu, Z., Qu, Y., Xu, J. and Cao, Y. (2020). Application of the XGBoost Machine Learning Method in PM2.5 Prediction: A Case Study of Shanghai. Aerosol Air Qual. Res. 20: 128-138. doi: online available: 10.4209/aaqr.2019.08.0408. https://doi.org/10.4209/aaqr.2019.08.0408

[Residuals plot 2016-2019] Residuals plot. Residuals Plot - Yellowbrick v1.4 documentation. (n.d.). Retrieved May 27, 2022, from online available: https://www.scikit-yb.org/en/latest/api/regressor/residuals.html