Helsinki Air Quality Index Prediction

Architectural Decisions

Introduction

The purpose of this document is to collect and justify all the architecture level decisions done during the project. In addition the document provides an overview of the air quality index definition and contains references to the used data sources.

The project followed IBM's lightweight Cloud Garage Method for data science process model and this document is structured to follow the process steps.

All project assets can be found from Github repository: https://github.com/rautis/Advanced_Data_Science_Capstone

Air quality index

Air quality index definition¶
(https://www.hsy.fi/fi/asiantuntijalle/ilmansuojelu/ilmanlaatutiedotus/Sivut/Ilmanlaatuindeksi.aspx)

The main purpose of the index is to provide a simple verbal description with colour coding that can be used when communicating the current air quality to the citizens. Air quality index is divided into five classes: from good to very bad. The air quality index classes describes what kind of health implications the corresponding class has. In addition the air quality index uses multiple international norms.
The air quality index is calculated as a combination of multiple air pollution measurements. At the moment the index is calculated from the following concentration measurements:

- sulfurdioxside (SO2)
- nitrogendioxcide (NO2)
- amount inhalable particles (PM10)
- amount of micro-particles (PM2.5)
- carbonmono-oxside (CO)
- ozone (O3)

The underlying table summarises the verbal descriptions (with colour coding) and numerical values of the air quality index with potential health and environment implications:

AIR QUALITY	INDEX VALUE	POTENTIAL HEALTH IMPLICATIONS	OTHER IMPLICATIONS
Very bad	>150	Possible for sensitive demographics	Clear implications for vegetation and materials in the long run
Bad	101-150	Possible for sensitive individuals	Clear implications for vegetation and materials in the long run
Tolerable	76-100	Unlikely	Clear implications for vegetation and materials in the long run
Satisfactory	51-75	Very unlikely	Clear implications for vegetation and materials in the long run
Good	< 50	Not observed	Mild implications for vegetation and materials in the long run

Data source

The data source used in the project are part of the City of Helsinki open data initiative. For more details see:

The project relies on three different data sets:
- Air quality index data from the Helsinki city center
- Weather data from the same region
- Traffic patterns

Data is available either via API calls or as a set of downloadable CSV files.

Architectural decision ID	Architectural decision	Justication
AD1	Download separate CSV files and use Python and Pandas to access them.	Since the time series data is historical it will not change during the project. Therefore the CSV data access is more convenient and friendly towards the open data servers. CSV file access is implemented with Python and Pandas. Pandas provides convenient way to access CSV files and can be considered as a “de facto” way of implementing it. In addition Pandas data frame provides a solid basis for data cleansing and exploration. For the data application that is used to provide continuous air quality index predictions the open data API should be used to ensure that the latest data is used for predictions. This however is not in the scope of this project since the project goal includes only the delivery of the data product (i.e. prediction model).

Initial data exploration

In order to get a better understanding of the data and to ensure the data quality the initial data exploration is conducted.

Architectural decision ID	Architectural decision	Justication
AD2	The following tool set is used in the data exploration: • Jupyter notebook • Pandas • Matplotlib • Seaborn	Initial data exploration needs to be interactive, flexible and it should use the best possible tools for the task. Therefore Jupyter notebook provides an excellent platform. It can be run either locally or on cloud. Notebooks can be easily shared. Since the project decided use CSV files as an initial data source Pandas is a natural tool for accessing them. In addition Pandas data frame can be used to extract various statistical metrics from the data. Finally while Matplotlib is used as a main toolkit for visualisation Seaborn is used for heatmaps since it is more convenient and powerful.

Extract, transform, load (ETL)

ETL phase ensures that the data from the original data source is transformed into more efficient data format and that the data is ready for feature extraction.

Architectural decision ID	Architectural decision	Justication
AD3	The following tool are used in the ETL phase: • Jupyter notebook • Pandas • Apache Parquet with gzip compression	Since this phases reused some of the methods developed during the initial data exploration (e.g. cleaning up the date columns) it made sense to continue with the same base technologies i.e. Jupyter notebook and Pandas. Finally since there is quite a lot of data that is numeric Apache Parquet data format provides an efficient (both from performance and storage perspective) data format.

Feature creation

Three new features will be created:
- Rolling mean with windows size = 3
- Rolling standard deviation with window size = 3
- Wiener filter

In addition the final data set includes data obtained from two different source:
- Air quality index time series (2014-2019)
- Weather data (2014-2019)

Traffic data was dropped from the final feature set since the project was unable to obtain daily traffic data.

Architectural decision ID	Architectural decision	Justication
AD4	Data set is augmented with three new features: • Rolling mean with windows size = 3 • Rolling standard deviation with window size = 3 • Wiener filter	After some initial modelling it was evident that the models are incapable of noticing trend changes. This can be alleviated with the introduction of filtered data that is more robust with respect to changes

Architectural decision ID	Architectural decision	Justication
AD5	Time series data is normalised using Scikit-learn robust scaler	According to some studies the LSTM Networks do not really perform too well with time series data. This can be alleviated with data normalisation. Robust scaler is selected since it preserves outliers.

Model definition

For the model definition phase defines the models that use in the prediction. It is important to have a solid baseline model that gives a lower bound that the deep learning models can be compared.

Architectural decision ID	Architectural decision	Justication
AD6	The following toolset is used in modelling: • Keras with TensorFlow backend • Jupyter notebook • Python statsmodel library • Scikit-learn • Matplotlib	Keras has a very intuitive API model model creation making it efficient to use. Statsmodel library has an ARIMA implementation that will be used as a baseline model. Finally Scikit-learn is used to estimate model performance.
AD7	Use autoregressive integrated moving average (ARIMA) as a baseline model	ARIMA can be fitted to time series data and can be used for forecasting. This will make it a good baseline to compare against.
AD8	Use Long-Short-Term-Memory (LSTM) recurrent neural network (RNN) in the deep learning model	LSTM is the most powerful RNN and in order to achieve best enough performance it makes sense to use it. Especially since RNNs might not be a perfect fit for time series predictions.
AD9	Lag size will be used as a variable parameter and for this the project need to find a best possible value.	

Model training

The model training phase takes the defined models and carrier out the actual training. Since the available data set is quite small and the defined models are relatively simple the training part is integrated with the modelling Jupyter notebook.

Architectural decision ID	Architectural decision	Justication
AD10	Model training will be done in a single machine with GPU	Since the models are not very large the training time is relatively short even for many hundreds epics. Therefore it is sufficient to train the model in a single PC with relatively powerful GPU.

Model evaluation

The defined models are first evaluated with a training set. Then the best models are tested with a validation data.

Architectural decision ID	Architectural decision	Justication
AD11	10% of the input data will be preserved for model validation	In practise this means approximate 7 months of time series data. This should be sufficient to get an understanding of the model performance.
AD12	Mean squared error (MSE) will be used to compare different models	MSE can be used a comparative measure since it gives a numerical estimator how well the model explains the given data. Smaller value indicates a better model

Model deployment

Model deployment is implemented as a simple Python script that wraps the store model behind a REST endpoint.

Architectural decision ID	Architectural decision	Justication
AD13	The deployment will use the following tools: • Python • Flask	Since the model is deployed into a REST service Flask provides a nice framework for service implementation. It is fast to develop and deploy.
AD14	Final model is deployed into a working RESTful server	Since the model is used to predict air quality it makes sense to wrap the model behind an API that can be called with currently known parameter