Visual analytics for exploring air quality data in an AI-enhanced IoT environment

Ilias Kalamaras
Information Technologies Institute - Centre for Research and Technology Hellas
kalamar@iti.gr

Ioannis Xygonakis
Information Technologies Institute - Centre for Research and Technology Hellas
xygonakis@iti.gr

Konstantinos Glykos
Information Technologies Institute - Centre for Research and Technology Hellas
glykos@iti.gr

Sigmund Akselsen
Telenor Group
sigmund.akselsen@telenor.com

Arne Munch-Ellingsen
Telenor Group
arne.munch-ellingsen@telenor.com

Hai Thanh Nguyen
Telenor Group & Norwegian University of Science and Technology
HaiThanh.Nguyen@telenor.com

Andreas Jacobsen Lepperod
Norwegian University of Science and Technology
andreasleppis@gmail.com

Kerstin Bach
Norwegian University of Science and Technology
kerstin.bach@ntnu.no

Konstantinos Votis
Information Technologies Institute - Centre for Research and Technology Hellas
kvotis@iti.gr

Dimitrios Tzovaras
Information Technologies Institute - Centre for Research and Technology Hellas
dimitrios.tzovaras@iti.gr

ABSTRACT

Visual analytics have an important role in the exploration and analysis of large amounts of data in IoT applications. Data visualizations can provide overviews of different aspects of data and user interaction can assist exploration. Recent advances in machine learning and Artificial Intelligence have provided methods that can be used in conjunction with visual analytics to enhance user perception. However, AI methods are often used as "black boxes", making them difficult for end-users to trust. In this paper, a novel visual analytics platform is presented, targeting two goals: a) an architecture for the creation of custom interactive visual analytics dashboards using well-defined components linked to each other, and b) the inclusion of components specifically for making AI methods more explainable. The proposed architecture and components are being used in the context of the AI4IoT pilot within the AI4EU project, which targets air quality monitoring through AI and visualization.

MEDES '19 November, 12-14, 2019, Limassol, Cyprus
© 2020 ACM. ISBN 978-1-4503-4895-9...$15.00
DOI:

CCS Concepts

•Human-centered computing → Visualization techniques; Visual analytics; Visualization toolkits;

Keywords

Visual analytics; reactive workflows; explainable AI; IoT

1. INTRODUCTION

Large-scale IoT (Internet-of-Things) applications employ a large number of sensors and result in a very large amount of collected data. In the context of IoT data analysis, two tasks are usually of relevance: exploring the large amounts of data to find subsets and patterns of interest; and analyzing the available data to make assessments and predictions. Air quality monitoring is an application area that can be significantly assisted by IoT installations and by relevant data analysis and exploration methods. As such, it is the subject of the AI4IoT pilot within the AI4EU project [1], which envisions the European ecosystem for Artificial Intelligence.

Visual analytics can assist the exploration of large amounts of data. They involve custom data visualization methods and they enable the operator to interact with them, in order to view the data from different perspectives and focus on details of interest. However, the wide variety of applications make different types of visualizations relevant in each case. Usually a custom set of visualizations needs to be compiled in a *dashboard*, in order to allow easy exploration of a particular dataset.

Visual analytics can themselves be assisted by data analytics methods, especially recent advances in machine learning and AI (Artificial Intelligence). AI methods are used to automatically extract patterns from data and make predictions. Their results can be presented through visualization to the end user. While AI methods are often very accurate in their results, they usually operate as "black boxes", without providing insight into how they work. This often makes them untrustworthy, since the operator is not confident about *why* a method produces the result it produces. Visual analytics can be used to make AI methods more transparent and *explainable*, by visualizing, apart from their results, the way they work.

This paper presents a novel platform for visual analytics with two main goals: a) facilitating the creation of custom visual analytics dashboards through the composition of basic visualization and AI components, and b) providing visualizations that reveal the way specific AI methods work, along with their results. The architecture of the proposed platform is based on the concept of components with well-defined inputs and outputs, that can be linked to each other into arbitrary workflows. The flow of information between components is *reactive*, meaning that when the inputs of a component change, the outputs are automatically re-calculated. Within this overall architecture, specific visualization components are hereby presented which are designed to make the AI models used more explainable.

2. RELATED WORK

2.1 Visual analytics for air quality management

Air quality data involve heterogeneous spatio-temporal data, hence the visualization becomes challenging. The challenge is inherent to the nature of the data, since they are high-dimensional and are produced from different types of sensors (e.g PM 2.5, CO2, NO2) that are distributed on different locations. Visual analytics are used to explore, interpret and identify data patterns, and different representations of the data reveal different aspects of them. Depending on the use case, simple charts can be used to explore the evolution of a specific variable or more sophisticated plots, involving clustering methods, can be deployed to identify data patterns and their evolution over time.

Specifically, time-series plots are used to depict the evolution of a single or multiple variables, heatmaps are used to illustrate the spatial distribution of a variable on a map [7], bar charts and histograms reveal the parameter distribution over time [7], colored dots on a map depict the intensity of the measured parameter based on color coding [7], circular heatmaps are used to compactly depict the change of a variable over the course of a month [12], while circular heatmaps on a map are used to explore the data from a spatio-temporal point of view [14].

There are also more sophisticated visualizations that are aimed to experts and data analysts. The Time-Correlation-Partitioning (TCP) tree is a concise visual representation depicting the correlations of multiple environmental quality parameters and their change over time based on information entropy [8]. Experts of the field can use this representation to explore correlation variations of air quality variables. Clustering and dimensionality reduction methods are used towards effectively visualizing air quality patterns and their evolution through time. Cluster storyline visualizations represent air quality patterns based on clusters derived from data and capture their evolution through time in a timeline graph [26]. Weighted complete graphs have also been proposed to depict the total correlation of all dimensions in data [20]. Polar systems with embedded circular pixel bar charts have also been proposed as a technique to visualize multivariate variables and vector variables (e.g wind) [20].

The above mentioned works are ad-hoc dashboards specifically designed for a specific application. Although a custom dashboard can be designed from scratch for a specific application, frameworks exist that facilitate this process. Vega [3] is a Web visualization library in which a visualization is composed of scales, axes and marks, while user input can be propagated through user-defined reactive signals, achieving interaction between the user and the visualization, as well as between visualizations. However, Vega does not support custom data analysis functions and services, which need to be handled outside the Vega specification. Orange [2] is a Python-based tool that allows the creation of a data analysis and visualization workflow, by inserting component blocks on a canvas and linking them together. The functionality of Orange is quite close to the functionality of the hereby proposed workflow-based architecture. However, Orange is offered as a stand-alone tool, and does not support the composition of the multiple components in a dashboard. In the architecture proposed hereby, our effort is to offer similar functionalities in a Web environment, allowing the end user to knit the available components in a custom dashboard.

2.2 Visual analytics for explainable AI

Artificial Intelligence has advanced immensely during the last years, moving towards the direction of complex and highly accurate models. As a counter-effect, the created prediction models are black-boxes even to machine learning experts, who are not able to fully comprehend why the model outputted a prediction. This questions the trustworthiness of a model, especially in critical applications such as medical diagnoses, where the root cause of the outcome should be transparent to doctors. Towards this end, the interpretability of a model is a vital factor for model deployment in a critical application, where a solution must be trustworthy and transparent.

Popular frameworks and methods that explain the prediction of any model (model agnostic), are LIME [21] and SHAP [16]. LIME is a technique that attempts to provide model prediction explanation, mostly for image and text models, by approximating the model with a local interpretable (linear) model trained on samples around the input sample, and provide as output a list of explanations or image segments, depicting the contribution of each feature to the prediction. SHAP computes the contribution of each feature to each prediction using Shapey values from game theory [16] and provides visualizations that explain individual feature importance both to individual predictions (e.g SHAP force plot, SHAP image plot) and to multiple predictions, as well as interaction between features (SHAP feature summary plot, SHAP dependence plot respectively) [15, 16].

Towards the interpretation of deep neural network (DNN) models, there are various methods that have been developed in order to explain the prediction. A group of techniques create heatmaps indicating the inputs (e.g. pixels of an image) that contribute most to a prediction, based on criteria defined by method, e.g. Sensitivity Analysis heatmaps [22], Layer-wise Relevance Propagation (LRP) [19], Saliency maps [23], Integrated Gradients [25], etc. Apart from image recognition, LRP has been applied to DNN models on applications with spatio-temporal data [24], audio data [6] and more [19].

In the context of visual analytics tools, ActiVis is an interactive visualization tool for neural network model exploration, featuring neuron activation view, computation graph overview of the model architecture upon data sample or subset selection [10]. CNNVis is a visual tool to understand and diagnose convolutional neural networks (CNN), by visualizing multiple sides of each neuron and the neuron activations [13]. RNNVis is another visual analytic tool to understand and diagnose Recurrent Neural Networks and their variants e.g long short term memory (LSTM), networks and gated recurrent units (GRU) [18]. For more information on deep learning visualizations there is an extensive review [9].

3. WORKFLOW-BASED DASHBOARD CREATION

The problem of custom dashboard design for an IoT application can be stated as follows: Given a set of measurements collected by several IoT sensors scattered in a geographical area, and a number of AI models applied on the data, create a dashboard that allows the operator to explore the available raw measurements and get insight in how the models work, in order to enhance the operator's trust on the models. This description indicates the following requirements for a dashboard creation system:

- IoT measurements are highly dynamic, with new measurements being collected at real time; the dashboard should update as soon as new measurements are available.
- The dashboard should be *interactive* in order to allow the operator to engage with the data and explore them.
- The dashboard should provide means to look into the applied models and visualize their internals, towards enhancing the transparency and explainability of the models.

In order to fulfill these requirements, a component-based architecture is proposed for dashboard design. The architecture is ultimately implemented as a library that can be used within web applications to facilitate the implementation of a dashboard. In the proposed architecture, a dashboard is composed using two types of primitive entities: components and links.

A *component* is a basic unit of functionality that accepts a number of inputs and produces a number of outputs, as depicted in Fig. 1. A component can also have side-effects, such as displaying on the screen. In case of components that display on the screen, user actions on the screen can also be used to produce the outputs of the component.

An example of a component with no side-effects is the REST call component, shown in Fig. 2a, which sends a REST request to a web service. It accepts the request URL, method, headers and body as its inputs and returns the service response as its output. An example of a component with side-effects is the scatterplot component, shown in Fig. 2b, that draws a scatterplot on the screen. It accepts the data to visualize, the attributes to use for the x and y axes, and the attributes to use for the point color and size. The

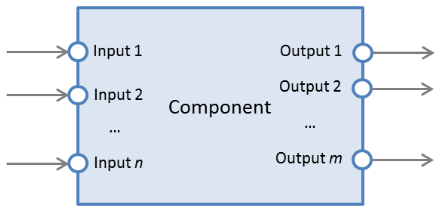
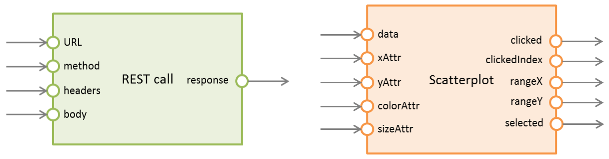


Figure 1: Conceptual view of a component.



(a) REST call (b) Scatterplot

Figure 2: Example components.

outputs are produced when the user performs actions on the scatterplot on the screen, such as clicking an item or selecting a rectangular range, in which cases the clicked or selected items and ranges are sent to the outputs.

Components are *reactive*: when one of the inputs changes its value, the outputs that depend on this input are automatically updated. Similarly, when the user performs an action on the screen that affects some of the outputs, the outputs are instantly updated. This provides a useful mechanism for implementing highly interactive dashboards. Table 1 provides a summary of some indicative components implemented in the proposed architecture.

It should be noted that the components listed in Table 1 do not contain complex data analysis methods, apart from basic data manipulation. Such components have been deliberately left out, due to their high computational needs that would slow down a client-side implementation of the architecture. Instead, in our implementation, methods such as prediction, classification, regression, clustering, etc. have been separated and made available as web services, which can be readily called through the "RESTCall" component.

The power of the proposed architecture comes when components are linked to each other in a workflow. A *link* connects the output of a component to the input of another, as depicted in Fig. 3. In this example, the data returned as the response of the REST call component are provided as the "data" input of the scatterplot component. If the other necessary inputs of the components are specified, the data will be visualized in a scatterplot view. The advantage of the reactive nature of the components is that once an input of the REST component changes, thus returning a different response, the scatterplot will be automatically updated to display the new data. A transformation function can be assigned to a link, in order to transform the data from the source before arriving at the target input.

A workflow can be specified in a declarative manner, as a JSON document, describing the components and the links between them. A component is given a unique name and is instantiated by specifying its type (one of the types of Table 1), any arguments necessary for the instantiation of the component (e.g. its container HTML element) and any default values for its inputs. A link is specified by the source output,

Table 1: Indicative implemented components in the proposed architecture.

Category	Component	Functionality
Data loading	DSVLoader	Loads a DSV file as a JSON object.
Data loading	RESTCall	Makes a REST call and returns the result.
Data loading	WebSocketConnection	Gets data from a web socket stream.
Data manipulation	Transformation	Transforms its input using a custom transformation function.
Data manipulation	Combination	Combines multiple inputs into a single output object.
Data manipulation	Filter	Filters data based on a data attribute and a condition.
Visualization	Datatable	Creates a data table view using the jQuery Datatables library.
Visualization	BarPlot	Creates a bar chart using the Vega library.
Visualization	LinePlot	Creates a line plot using the Vega library.
Visualization	ScatterPlot	Creates a scatterplot using the Vega library.
Visualization	Heatmap	Creates a heatmap view using the Vega library.
Visualization	ForceDirected	Creates a force-directed graph layout using the Vega library.
Visualization	MapView	Creates a map view using the Leaflet library.
Visualization	AnnotatedLineChart	Creates an annotated line chart (see Section 4.1).
Visualization	SHAPChart	Creates a SHAP chart (see Section 4.2).
Input controls	HTMLSelect	Creates an HTML select box.
Input controls	HTMLSlider	Creates an HTML slider.

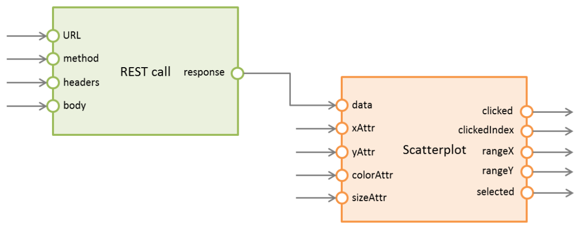


Figure 3: Link example.

the target input and an optional transformation function to transform the data from the source to the target. An example JSON specification of a workflow, corresponding to the example of Fig. 3 can be seen below.

```
{
  "components": {
    "restCall": {
      "type": "RESTCall",
      "defaults": {
        "url": "http://my_web_service.json",
        "method": "GET"
      }
    },
    "scatterplot": {
      "type": "ScatterPlot",
      "args": ["scatterplotContainerDiv"],
      "defaults": {"xAttr": "x", "yAttr": "y"}
    }
  },
  "links": [
    {
      "from": "restCall.response",
      "to": "scatterplot.data"
    }
  ]
}
```

The proposed architecture has been implemented as a JavaScript library that can be imported in web applications. The inputs and outputs of each component are implemented as reactive streams using the RxJS[1] library. Individual visualization and data manipulation components use libraries such as D3[2], Vega[3] and Leaflet[4]. It should be noted that a workflow defines *functionality*, not *appearance*. It defines which components to use and how to link them but does not specify where the components will be placed on the screen and how they will appear; this is left to HTML, CSS, etc. Procedures are currently ongoing for releasing the library as an open-source project.

4. COMPONENTS FOR VISUALIZING EXPLAINABLE MODELS

Within the overall architecture described above, components can be designed that serve specific visualization purposes. The components developed in the context of the AI4EU project should provide comprehensive information and analysis results to the human viewer, as well as insights in the ways that the AI models work. The latter is relevant to the guidelines for explainable and trustworthy AI, which are central in the AI4EU concept. This section presents the visualization components implemented hereby to address these purposes.

4.1 Annotated line chart

The proposed annotated line chart allows the user to see which parts of a time-series influence most the result of a prediction model. Existing visualizations for text prediction and image classification [21] highlight the areas (letters or pixels) that mostly affect the prediction, so that the user can decide whether the algorithm works in an expected manner or not. The annotated line chart uses the same principle in the context of time-series visualization, highlighting the time instances that mostly affect the predicted values.

An example annotated line chart can be seed in Fig. 4. The chart depicts a time-series of a particular measurement of interest, such as particle concentration in the air. The historical measurements are depicted as a solid line, while predicted values, here by an ARIMA model, are depicted as

[1]https://rxjs-dev.firebaseapp.com/

[2]https://d3js.org/

[3]https://vega.github.io/vega/

[4]https://leafletjs.com/

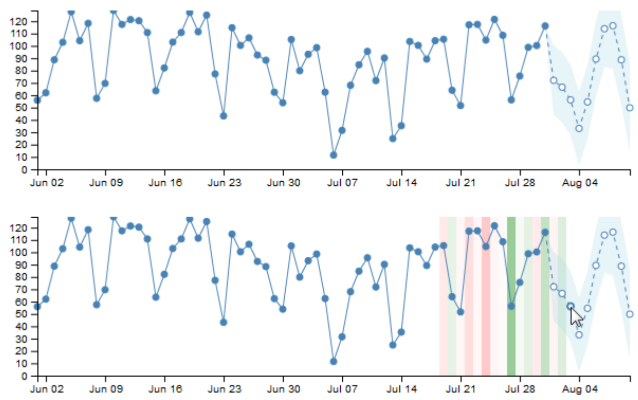


Figure 4: Example of annotated line chart.

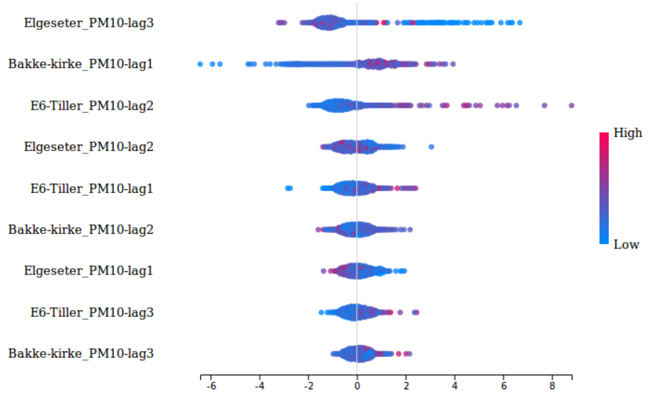


Figure 5: Example of a SHAP summary plot.

white dots connected with dashed lines. The 95% confidence interval for each prediction is illustrated as a shaded area around each point.

In order to add information about how the ARIMA model makes its predictions, the chart is enhanced as follows. When the user moves the mouse pointer over one of the predicted points (bottom of Fig. 4), the historical measurements used to produce this prediction are highlighted, according to the corresponding weight learned by the model (m highlighted moments for an m-order auto-regressive model). Positive weights are colored in green and negative weights in red, while larger absolute values are colored in higher opacity.

In this manner, areas of high opacity denode the time instances that mostly affect the selected prediction. In the example of Fig. 4, the most opaque time instance (the vivid green bar) appears 7 time intervals before the selected prediction, which coincides with the dominant period of the time series (7 intervals), indicating that instances at the same position in the previous period are highly relevant to the current prediction. This behaviour seems expected on behalf of the ARIMA model; if a rather accurate prediction was produced using time instances that seem irrelevant, the model could be questioned for over-fitting.

4.2 SHAP chart

The SHAP framework [16] provides a consistent methodology to evaluate the contribution of each classifier input (feature) to the final predicted output as it was aforementioned in Section 2.2. SHAP has initially been implemented to provide explanations for a machine-learning model predicting hypoxaemia event risk factors during general anaesthesia [17]. As it is noted in this study, hypoxaemia risk estimation accompanied with explanations was really useful for doctors performing surgery, since they were able to evaluate the risk estimation and decide whether or not the estimation is consistent with their knowledge and hence regard it as trustworthy.

SHAP values measure the contribution of each individual feature to the prediction that a machine-learning model made, with the exact method of computing them described in [15, 16]. The relation between SHAP values and prediction is straight-forward; given an input feature vector, the predicted value is equal to the sum of all SHAP values of the input vector plus a base-reference value, that is the expected model outcome given no input [15]. Hence the name SHAP, that stands for SHapley Additive exPlanation.

SHAP summary plots [15] illustrate the distribution of feature importance, given a trained model and a dataset. Each classifier (feature vector) input is represented as a colored dot, with the color indicating the value of that feature. A dot is placed on the x-axis, according to the contribution of that feature to the final model outcome, and can have either positive or negative contribution. With the SHAP summary plot we can infer which are the most important features that the model relies on, and how high and low feature values impact feature importance. Fig. 5 provides an example plot. The vertical axis corresponds to the different features used, while the horizontal axis to the SHAP values. The top features exhibit large absolute SHAP values, which indicates that they are important for the specific model, while features at the bottom do not contribute much.

5. PILOT APPLICATION

As an example application of the proposed architecture and components, a dashboard has been created for the AI4IoT pilot of the AI4EU project, focusing on air quality monitoring. The overall purpose of the pilot is to make use of IoT sensors to make assessments and predictions of air quality. A detailed description of the pilot can be found in [4].

5.1 Data sources

The pilot is run in Trondheim, Norway, a city with around 200,000 inhabitants. Air quality in Trondheim is usually fine, but there is a high variation, and days of severe pollution can occur, especially during the winter. The available data include pollutants from the Norwegian Environment Agency[5] measured by industrial sensors, weather data from the Norwegian Meteorological Institute[6], and traffic data from the Norwegian Road Authorities[7] counted by inductive loops.

The pilot site includes three stations for measuring pollutants and one station for measuring weather parameters. In addition, a number of traffic counting stations, 35 in total, are used to record traffic-related parameters, such as the number of passing vehicles in both driving directions. More details on the dataset is available in [11]. In addition,

[5]https://api.nilu.no/
[6]https://frost.met.no/
[7]https://www.vegvesen.no/trafikkdata

trials with portable air-quality micro-sensing units are ongoing. These units consist of a board with sensors reporting temperature, humidity, PM 10, PM 2.5, NO2 as well GPS coordinates. This board has an integrated modem that supports both LTE-M and NB-IoT connectivity. So far only off-the-shelf low-cost micro-sensors (in the price range less than USD 40) have been used in these designs. The initial test of the data quality of sensors from the unit developed in this project (compared to an industrial sensor of particle dust in the same location) has indicated that the measurements were influenced by variations in temperature, humidity and pollutant levels. So far these data have not been included for training prediction models in this project. The future plan includes systematic testing of more micro-sensors. It also includes combination of the mentioned public data with other datasets (e.g. cleaning actions by municipality, installed fireplaces, people mobility data).

5.2 Machine Learning methods used

In previous work [11], we experimented with a variation of machine learning methods and included additional features in order to improve the prediction of NOX, CO2, PM2.5 and PM10 forecasts. We used ARIMA as baseline and compared its result with Ridge, Random Forest, Gradient Boosting, and Dropouts meet Multiple Additive Regression Trees (DART) following a multi-output strategy of the single target regressors. We also added deep learning approaches to the experiments, namely Multilayer Perceptrons (MLP) and Gated Recurrent Units (GRU), which are both designed for multi-output regression to compare all the mentioned models with deep learning. When creating the models we also incorporated other data sources in the air quality data set such as weather observations, traffic volume count, and wood burner data set. The experiments showed that DART has the strongest results of predicting the overall air quality for all the pollutants (PM2.5, PM10, NO2) when predicting both 24 and 48 hours ahead. Further, we found that GRU can classify sudden changes better than the other methods.

Although the accuracy of the above mentioned methods is important, explaining why the used models perform as they perform is equally important to make trustworthy estimations. In our study, we have so far used the ARIMA and Random Forest methods as case studies of how visualization can help in this direction, as can be seen in the following example. Further experimentation with the other types of models will be considered in the future.

5.3 Example dashboard

An example dashboard created using the proposed architecture is depicted in Fig. 6. Its purpose is to visualize the concentration of air pollutants over time in multiple geographical areas, as well as to show the correlations between measurements of different areas, which may be indications for major sources of pollution. At a different level, the dashboard illustrates the usage of the proposed architecture to design a custom analysis.

A map of the area around Trondheim is displayed in the top left part, showing the locations of the four stations gathering pollutant and weather information. The heatmap layer shows the values of a selected pollutant at a specific point in time. Below the map, three line charts depict monthly averages of the three main pollutants of interest (PM 10, PM2.5 and NO2) through time, in a selected area on the map, along with the predictions made by an ARIMA model. The annotated line chart type described in Section 4 has been used, in order to depict the predictions as well as the model weights.

The map and the line charts are linked: when the user clicks on the map, the line charts are updated to show the measurements for this specific area; similarly, when the user clicks on a time instance in one of the line charts, the heatmap layer of the map is updated to show the corresponding pollutant measurements at this time instance. This linking is achieved by connecting the corresponding outputs of the map component to the necessary inputs of the line chart components, and vice versa, through any appropriate intermediate transformations. The dashboard designer can add such links at any time, in order to make one component react to the changes of another.

The right panel displays an analysis of the importance of geographical areas in model predictions. To make the presentation clearer, we start from the bottom. A SHAP chart (see Section 4) shows the significance of the measurements of different areas in predicting the next value of a selected parameter. The SHAP chart uses weekly averages of the measurements. The user selects an area on the map and one of the pollutant types. The corresponding time-series is used as the target values for a Random Forest regression model, while the measurements of the same pollutant in the other areas are used as predictors. A number of lagged time instances from all time series is used as the predictor features. The SHAP values of the predictions for all features are displayed in the SHAP chart. The features are ordered according to the variance of the SHAP values in each feature: high-variance features at the top contain several large absolute SHAP values, i.e. they largely influence the predictions.

At the top right part, a heatmap depicts the correlations among areas. For each area of the vertical axis, a SHAP-based analysis has been performed for the selected pollutant and lag number, similar to the one that produced the SHAP chart. The considered features are listed in the heatmap's horizontal axis. The variances of the SHAP values for each considered feature are used to color the heatmap cells, normalized per area. A high cell value means that the corresponding feature of the horizontal axis is important to make predictions about the pollutant in the corresponding area of the vertical axis. In the example shown that uses 2 lags, we can see that most influence for an area comes from the same area with a lag of 1 time interval, which is rather expected. In addition, we can see that the features of the E6-Tiller area are rather important for predictions in all areas, which may indicate a central role of this area in pollutant monitoring. The operator can first use this heatmap for an overview of influences between areas, and then focus on a specific area to see in detail why the influences are such, by viewing the corresponding SHAP chart below.

The different parts of the dashboard are implemented as components of the proposed architecture described in Section 3. The workflow that provides the functionality for this dashboard is depicted in Fig. 7. Components that render visualizations or controls on the screen are colored in orange, those that call external web services are colored in green and those that perform intermediate transformations and manipulations are colored in blue.

6. CONCLUSIONS

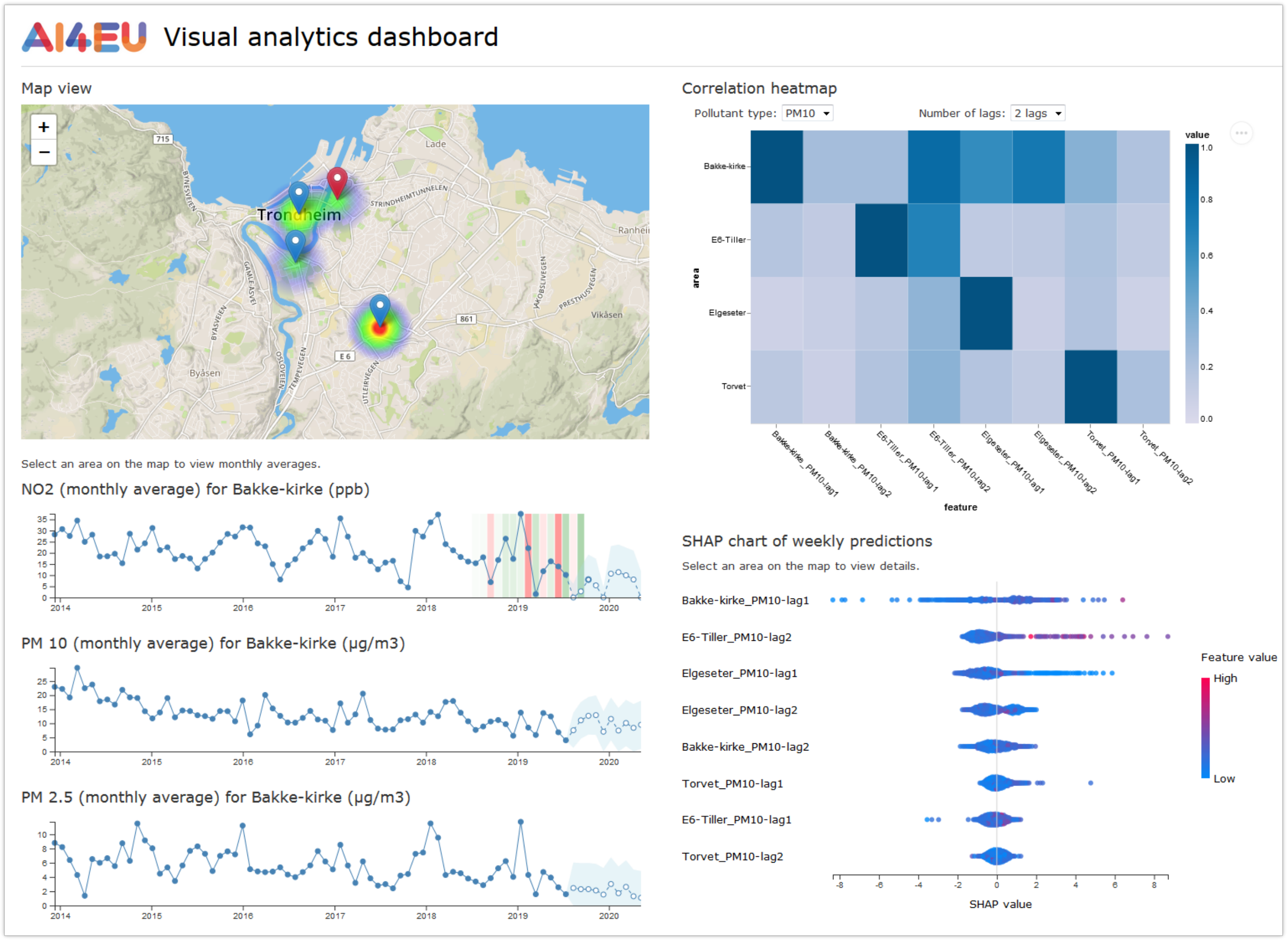

AI4EU Visual analytics dashboard
Map view
Correlation heatmap
Pollutant type: PM10
Number of lags: 2 lags
Select an area on the map to view monthly averages.
NO2 (monthly average) for Bakke-kirke (ppb)
PM 10 (monthly average) for Bakke-kirke (μg/m3)
PM 2.5 (monthly average) for Bakke-kirke (μg/m3)
SHAP chart of weekly predictions
Select an area on the map to view details.
Bakke-kirke_PM10-lag1
E6-Tiller_PM10-lag2
Elgeseter_PM10-lag1
Elgeseter_PM10-lag2
Bakke-kirke_PM10-lag2
Torvet_PM10-lag1
E6-Tiller_PM10-lag1
Torvet_PM10-lag2
Feature value
High
Low
SHAP value
area
feature
value

Figure 6: Example dashboard.

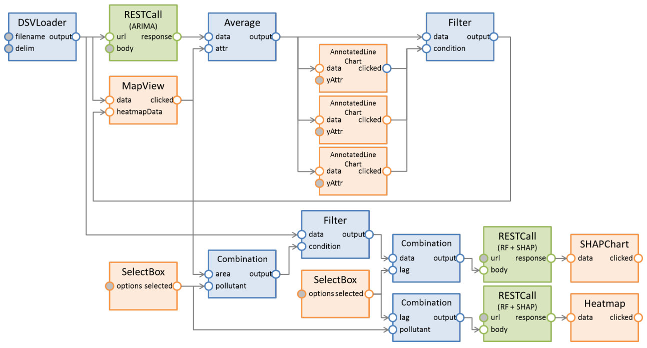


Figure 7: Workflow behind the dashboard of Fig. 6.

This paper presented a proposed architecture for the design of custom interactive visual analytics dashboards, which can find usages in applications employing large amounts of data, such as IoT applications. In the proposed architecture, each (visible or not) part of a dashboard is considered as a component performing a focused set of operations with well-defined inputs and outputs. This allows components to be linked to each other and form custom reactive workflows defining the logic behind a visual analytics dashboard. Visualization methods can be designed as components to cover specific needs and be readily used within workflows. In this paper, two components have been specifically designed towards achieving explainability in AI models: the Annotated Line Chart visualizes the parameters of an ARIMA prediction model, while the SHAP Chart provides insight in the most significant features used for Random Forest regression. Such visualizations provide hints into how the analysis outcome is produced, which can be of assistance to the operator in trusting the models.

The work presented is work in progress. The proposed architecture is generic enough to include any functionality that can be structured in a component form, providing a basis for further development. New visualization components will be considered in the future, especially ones relevant to the explainability of AI models such deep neural networks. Other types of data available in the pilot, such as weather and traffic data, will be used to explore different types of models and visualizations. Furthermore, the implementation of the architecture as a library permits its inclusion in other projects as well. Existing component-based visual analytics platforms, such as VisualBox [5], can benefit from the input-output specification imposed by the proposed architecture and by the linking mechanism between components. We will examine the integration of our proposed platform in such projects in the near future.

7. ACKNOWLEDGMENTS

This work is part of the AI4EU project (www.ai4eu.eu), funded from the EU Horizon 2020 research and innovation programme, under the Grant Agreement No 825619.

8. ADDITIONAL AUTHORS

9. REFERENCES

[1] European project AI4EU. https://www.ai4eu.eu/, September 2019.

[2] Orange - Data Mining Fruitful and Fun. https://orange.biolab.si/, September 2019.

[3] Vega - A Visualization Grammar. https://vega.github.io/vega/, September 2019.

[4] S. Akselsen, P. E. Aurdal, K. Bach, J. P. Costeira, I. Kalamaras, A. J. Lepperød, P. Lima, I. Martinkenaite, O. J. Mengshoel, A. Munch-Ellingsen, H. T. Nguyen, D. Tzovaras, T. Veiga, K. Votis, L. Wienhofen, W. Zhang, and P. Øzturk. On the need for explanations, visualisations and measurements in data-driven air quality monitoring and forecasting. In *1st International Workshop on Evaluation and Benchmarking of Human-Centered AI Systems (EBHAIS-2019)*, 2019.

[5] P. E. Aurdal. *VisualBox – A Generic Data Integration and Visualization Tool*. Master's thesis, The Arctic Univeristy of Norway (UiT), Tromsø, Norway, 2019.

[6] S. Becker, M. Ackermann, S. Lapuschkin, K.-R. Müller, and W. Samek. Interpreting and explaining deep neural networks for classification of audio signals. *arXiv preprint arXiv:1807.03418*, 2018.

[7] P. Chen. Visualization of real-time monitoring datagraphic of urban environmental quality. *EURASIP Journal on Image and Video Processing*, 2019(1):42, 2019.

[8] F. Guo, T. Gu, W. Chen, F. Wu, Q. Wang, L. Shi, and H. Qu. Visual exploration of air quality data with a time-correlation-partitioning tree based on information theory. *ACM Transactions on Interactive Intelligent Systems (TiiS)*, 9(1):4, 2019.

[9] F. M. Hohman, M. Kahng, R. Pienta, and D. H. Chau. Visual analytics in deep learning: An interrogative survey for the next frontiers. *IEEE transactions on visualization and computer graphics*, 2018.

[10] M. Kahng, P. Y. Andrews, A. Kalro, and D. H. P. Chau. Activis: Visual exploration of industry-scale deep neural network models. *IEEE transactions on visualization and computer graphics*, 24(1):88–97, 2017.

[11] A. J. Lepperød. *Air Quality Prediction with Machine Learning*. Master's thesis, Norwegian University of Science and Technology (NTNU), Trondheim, Norway, 2019.

[12] H. Li, H. Fan, and F. Mao. A visualization approach to air pollution data exploration - a case study of air quality index (PM2.5) in Beijing, China. *Atmosphere*, 7(3):35, 2016.

[13] M. Liu, J. Shi, Z. Li, C. Li, J. Zhu, and S. Liu. Towards better analysis of deep convolutional neural networks. *IEEE transactions on visualization and computer graphics*, 23(1):91–100, 2016.

[14] W. Lu, T. Ai, X. Zhang, and Y. He. An interactive web mapping visualization of urban air quality monitoring data of China. *Atmosphere*, 8(8):148, 2017.

[15] S. M. Lundberg, G. G. Erion, and S.-I. Lee. Consistent individualized feature attribution for tree ensembles. *arXiv preprint arXiv:1802.03888*, 2018.

[16] S. M. Lundberg and S.-I. Lee. A unified approach to interpreting model predictions. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 4765–4774. Curran Associates, Inc., 2017.

[17] S. M. Lundberg, B. Nair, M. S. Vavilala, M. Horibe, M. J. Eisses, T. Adams, D. E. Liston, D. K.-W. Low, S.-F. Newman, J. Kim, et al. Explainable machine-learning predictions for the prevention of hypoxaemia during surgery. *Nature biomedical engineering*, 2(10):749, 2018.

[18] Y. Ming, S. Cao, R. Zhang, Z. Li, Y. Chen, Y. Song, and H. Qu. Understanding hidden memories of recurrent neural networks. In *2017 IEEE Conference on Visual Analytics Science and Technology (VAST)*, pages 13–24. IEEE, 2017.

[19] G. Montavon, W. Samek, and K.-R. Müller. Methods for interpreting and understanding deep neural networks. *Digital Signal Processing*, 73:1–15, 2018.

[20] H. Qu, W.-Y. Chan, A. Xu, K.-L. Chung, K.-H. Lau, and P. Guo. Visual analysis of the air pollution problem in Hong Kong. *IEEE Transactions on visualization and Computer Graphics*, 13(6):1408–1415, 2007.

[21] M. T. Ribeiro, S. Singh, and C. Guestrin. Why should I trust you?: Explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 1135–1144. ACM, 2016.

[22] W. Samek, T. Wiegand, and K.-R. Müller. Explainable artificial intelligence: Understanding, visualizing and interpreting deep learning models. *arXiv preprint arXiv:1708.08296*, 2017.

[23] K. Simonyan, A. Vedaldi, and A. Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps. *arXiv preprint arXiv:1312.6034*, 2013.

[24] I. Sturm, S. Lapuschkin, W. Samek, and K.-R. Müller. Interpretable deep neural networks for single-trial EEG classification. *Journal of neuroscience methods*, 274:141–145, 2016.

[25] M. Sundararajan, A. Taly, and Q. Yan. Axiomatic attribution for deep networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 3319–3328. JMLR. org, 2017.

[26] Z. Zhou, Z. Ye, Y. Liu, F. Liu, Y. Tao, and W. Su. Visual analytics for spatial clusters of air-quality data. *IEEE computer graphics and applications*, 37(5):98–105, 2017.