Tutorial

Predictive analytics for accuracy in quality assessment in manufacturing

Follow this use case on machine learning applied to IIOT real data

☆ Save 👍 Like

Site feedback

By David Mesa López, Plamen Kiradjiev

Published September 2, 2020

Today, with more than 1.8 million industrial robots in different factories, the quantity of cutting-edge technology in factories around the world has reached a new record. This fast development in the industrial world is representative of the increasing automation and digitalization in factories. In fact, international studies assume that companies will invest an estimated 250 billion dollars in IoT alone by 2020, which is just one of many different technologies in this environment.

Within this context, the German Ministry of Economics and Energy announced different Centers of Competences where the research of the Industry 4.0 paradigm was to be investigated. *SmartFactoryKL* is a network of more than 50 member organizations from industry and research. These partners perform research and development projects related to Industry 4.0 and the factory of the future. The work ranges from developing their vision and preparing descriptions all the way to industrial implementation.

The production line of this type of solution represents a reconfigurable and flexible system where different technologies from different manufacturers come together. From the connection perspective, it is highly interoperable through universal plug-in connections for electricity, compressed air, and industrial ethernet.

Figure 1. SmartFactoryKL manufacturing station

IBM® participates as a partner in SmartFactoryKL and is responsible for the integration as well as analytics. To develop the solutions, a digital twin of the SmartFactoryKL has been implemented and combined with data from all production modules. The main goal is to achieve the highest possible quality. In this use case, a predictive quality mechanism is going to be implemented to detect anomalies.

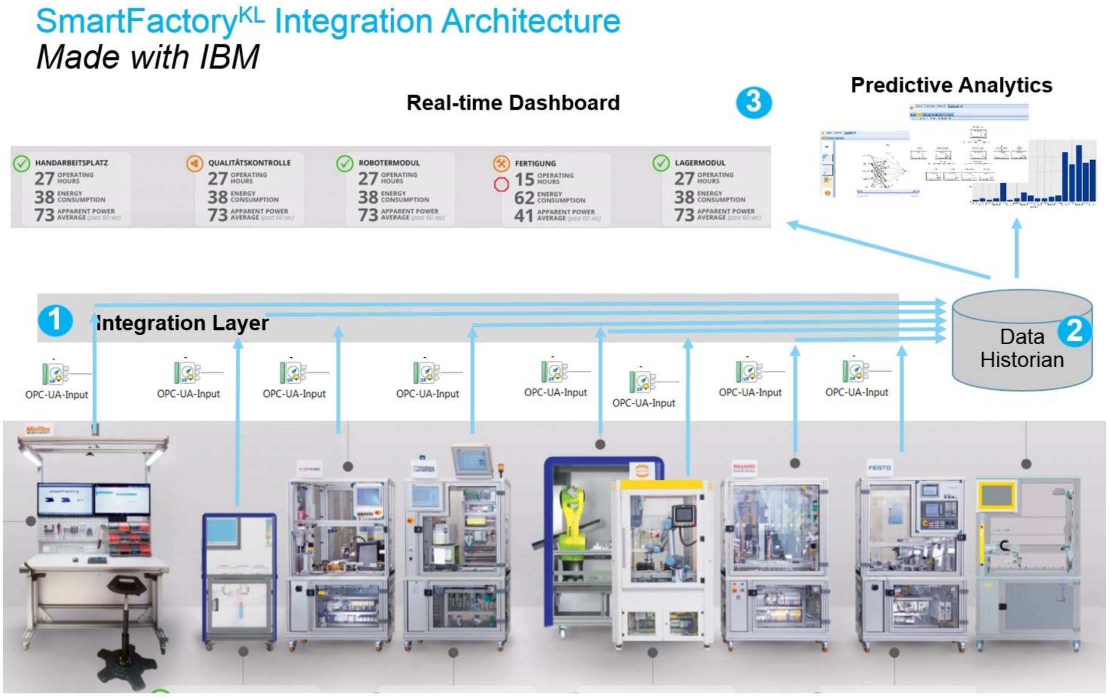
SmartFactory^KL Integration Architecture
Made with IBM
Real-time Dashboard
3
Predictive Analytics
HANDARBEITSPLATZ 27 OPERATING HOURS 38 ENERGY CONSUMPTION 73 APPARENT POWER AVERAGE (past 60 sec)
QUALITÄTSKONTROLLE 27 OPERATING HOURS 38 ENERGY CONSUMPTION 73 APPARENT POWER AVERAGE (past 60 sec)
ROBOTERMODUL 27 OPERATING HOURS 38 ENERGY CONSUMPTION 73 APPARENT POWER AVERAGE (past 60 sec)
FERTIGUNG 15 OPERATING HOURS 62 ENERGY CONSUMPTION 41 APPARENT POWER AVERAGE (past 60 sec)
LAGERMODUL 27 OPERATING HOURS 38 ENERGY CONSUMPTION 73 APPARENT POWER AVERAGE (past 60 sec)
1
Integration Layer
2
Data Historian
OPC-UA-Input
OPC-UA-Input
OPC-UA-Input
OPC-UA-Input
OPC-UA-Input
OPC-UA-Input
OPC-UA-Input
OPC-UA-Input

Figure 2. Digital twin implementation

Use case environment

A specific advantage of machine learning for manufacturing is the capability to process and interpret both structured and unstructured data from production processes. AI-based systems can identify patterns and anomalies to detect problems automatically. These systems can warn you in time to make suggestions to eliminate the problem or in a best case scenario, eliminate the problems themselves. Raw data is the main asset that is needed to implement this type of solution. Data for this use case is taken directly from one of the production modules, more precisely from the weight measurement module where some random vibrations have been made with the intention of simulating a real case in which one of the modules starts to fail. The extracted data consists of eight different fields that define the use case.

Site feedback

- `Time`: The timestamp of the process
- `AmplitudeBandWidth` and `AmplitudeMean`: Statistical parameters that measure system vibrations
- `NumberOverloads` and `NumberUnderloads`: Parameters that indicate whether the measures have been taken with different numbers of product loads
- `StabilizationTime`: The time it took for the measure to stabilize
- `StableWeight`: Flag that indicates whether the WeightValue is real or calculated
- `WeightValue`: Value of the weight of the product

In this case, the mission of the implemented solution is going to be to detect whether the weight value of the module is reliable, based on the other inputs. If the goal is reached, it indicates that the model is able to predict whether the production line is going to behave as expected based on the vibrations of the machines, which means that the high quality is reached. To define whether the quality of the weight sample is good, a new variable must be created. In this use case, we know that the weight module is working as expected because the `WeightValue` parameter has a value of 38 or a value of 0. If it's 38, it means that you are measuring the normal weight of a unit of product. If it's 0, it means that you are measuring an empty space. In this context, this new variable is going to be the target that the model is going to predict. It will have a value of 1 for reliable measurements (38 or 0) and -1 for not reliable measurements.

The solution is implemented in IBM Watson SPSS® Modeler, which enables predictive analytics with various statistical and mathematical techniques for AI modeling or data mining.

Site feedback

Creating an IBM Watson account and SSPS model

IBM Watson SPSS Modeler flows in Watson Studio provide an interactive environment for quickly building machine learning pipelines that flow data from ingestion to transformation to model building and evaluation, without needing any code. To complete this tutorial, you need an IBM Cloud account. You can obtain a free trial account, which gives you access to IBM Watson Studio.

1. Create a free IBM Cloud account.
2. Sign in to Watson Studio using the account that you created for your IBM Cloud account. If you do not have the Watson Studio service available, you can create an instance, then click **Get started**.
3. Click **Create a project**, then **Create an empty project**.
4. In the **New project** window, name the project, then click **Create**.
5. Click **Add to project**, then **Data** to import the data that you want to use.
6. Click **Add to the project**, and select **Modeler Flow**.

Implementing the model

To implement the model:

1. Create a **Data** node. From the node menu on the left, click **Import**, then **Data Asset**, and drag it to the flow.
2. Open the node, and attach the data from the data assets of your project by clicking **Change data asset > Data assets**, where your data should be available.

3. Click **Outputs**, and drag the **Data Audit** node to the flow to get an overview about the data.
4. Connect the Data Audit node with the Data node. The Data Audit node displays the fields used for the system and presents different parameters to understand the data. You can configure the node if you want to compute some statistical variables. This is an ideal way to get quick understanding of the input data.
5. Click **Field Operations**, and drag the **Derive** node to the flow.
6. Enter **writeWeight_accuracy** in the Derived Field Name.
7. Select the conditional mode with nominal measurement in the **Derive as** field.
8. Define the condition that you want to apply in the model.

    1. In the `if` statement, enter **(WeightValue >= 37.6)** and **(WeightValue <= 38.4)** or **(WeightValue >= -0.4)** and **(WeightValue <= 0.4)**.
    2. In the `then value equals` statement, write **1**.
    3. In the `else value equals` statement, write **-1**.

    Here, you are selecting the values that you considerate valid, in this case, 0 and 38. This is the quality measure of the model. As you can deduce from the `if` statement, the value 1 is assigned to valid values and -1 is assigned to values that are not valid.

9. Click **Field Operations**, and drag, connect, and open a **Type** node. Configure the role for `Weight_accuracy` as target because you want to predict this field. `WeightValue` and `NumberOverloads` must be configured as **None** because `WeightValue` is a variable related with the target, and `NumberOverloads` has just one value. Configure the measure for `StabilizationTime` as flag and `Weight_accuracy` as Ordinal.
10. Click **Modeling**, and drag and connect the **Auto Classifier** node. Then, run the model. The SPSS software selects and builds the model for you. Then, another green node appears.
11. Drag and connect to the auto-created node, one **Analysis** node and one **Table** node from the output menu to analyze the results. Make sure that you save the model. If you followed these steps exactly, your model should look like the model in the following image.

Model

Figure 3. Model

Conclusions about the data

The most important part of any AI process is the decision-making step based on the knowledge that you are able to extract from the raw data by using the machine learning model. This section gives examples of some conclusions that could be reached due to the implemented solution. To analyze the results, the auto-created green node has an option called "View Model" where all of the model's features reside. For this specific modeling option, the SSPS software runs different algorithms for you as you can see when you click **View Model**. For this example, the conclusions have been drawn from the C5.0 algorithm, an algorithm that is based on a decision tree that can be seen by clicking the C5.0 option. The two best tools that you can use to analyze a model is searching features in the decision tree or in the top decision rules. If Auto Classifier results don't show the C5.0 result, open the node, then click **Expert > Select Models**.

Site feedback

1. Global results of the analysis. The first output that is possible is the accuracy with which the model is able to predict the value of `Weight_accuracy` based on the other variables. This result means that the model is able to categorize the different samples with a certain value and make the model more reliable. In this example, the accuracy of the prediction is fairly high so the model improves the confidence of the results. This measure doesn't show whether the accuracy variable is 1 or -1. It shows the likelihood that the model will correctly categorize the specific sample. So, this is a direct parameter that defines the quality of the model itself.

   Global results

   Figure 4. Global results

2. Another feature that the model brings is that you can observe that the calculated values always give you values that are wrong. This is a good performance indicator in the calculated process. You can see this feature at the beginning of the tree model, where if `StableWeight` is False, the measure of the weight is not reliable.

Begin of the decision tree

Figure 5. Beginning of the decision tree

3. Another important feature of the use case that would be interesting to know is the range of values of the different fields that allow the machine to produce a good measurement. For example, this is important because you can limit the vibrations values to this range for the production procedures to ensure a good measurement of the final product. In this case, the `AmplitudeMean` should be a minor value of 3319.0 to ensure a good behavior with a `StabilizationTime` between 383 – 1225. This process could be followed to analyze the `-1` (or incorrect) measurement.

   Top decision rule for +1 prediction

   Figure 6. Top decision rule for +1 prediction

4. The last thing that is going to be analyzed in this section is the different paths that you can observe in the decision tree. There are many different conclusions that can be extracted from this decision tree, and the one shown in the following image is just a quick example. If you follow the bottom path, you should notice that for samples with an `AmplitudeMean` greater than 7700 you must have an `AmplitudeBandWidth` less than 2106 and greater than 745 or less than 166 to be considered as a reliable measure.

   Bandwidth analysis

   Figure 7. Bandwidth analysis

In conclusion, it can be said that the model achieved the original goal, predict the quality of the measurement with a likelihood of 93.38%. The first real implementation of this solution is obviously to predict, based on the vibrations in the production line, if the quality of the measure is as high as expected, extracting the `-1` predictions and comparing them with the reliable samples. Another important point is that, as shown previously in these conclusions, we can extract the boundary conditions in which unreliable measurements appear. Being aware of these conditions is possible to force the production line to work under limit parameters that ensure the highest possible productivity while providing us with measures that we know can be trusted.

Legend ⓘ

Categories

Analytics IBM SPSS Modeler

Machine learning

Predictive analytics

Site feedback

Table of Contents

Related

Tutorial

Build a predictive machine learning model quickly and easily with IBM SPSS Modeler

July 9, 2020 →

Site feedback

Tutorial

Fraud detection with SPSS Modeler

February 17, 2020 →

Tutorial

Build and compare models using IBM SPSS Modeler

July 9, 2018 →

Build
Smart ↓
Build
Secure ↑

IBM Developer
About
FAQ
Third-party notice

Follow Us
Twitter
LinkedIn
Facebook
YouTube

Explore
Newsletters
Code patterns
APIs
Articles
Tutorials
Open source projects
Videos
Events
Cities

Select a language
English
Português
Español
한글

Site feedback

Community
Career Opportunites
Privacy
Terms of use
Accessibility
Cookie preferences