Requirements for implementation

A quality prediction system for soft fruit based on a Bayesian Belief Network

dr. R.J. Vlek, dr. D.J.M. Willems, dr. H. Rijgersberg

WAGENINGEN
UNIVERSITY & RESEARCH

Requirements for implementation

A quality prediction system for soft fruit based on a Bayesian Belief Network

Authors: dr. R.J. Vlek, dr. D.J.M. Willems, dr. H. Rijgersberg

Institute: Wageningen Food & Biobased Research

This research project has been carried out by Wageningen Food & Biobased Research commissioned by the Dutch Ministry of Economic Affairs and funded by Driscoll's B.V., in the context of GreenCHAINge project (project number 6239090303).

Wageningen Food & Biobased Research
Wageningen, November 2018

Report 1879

Version: final
Confidential until: December 2021
Reviewer: dr. N.J.J.P. Koenderink
Approved by: dr.ir. H. Wensink
Client: Driscoll's B.V.
Sponsor: the Dutch Ministry of Economic Affairs

PO box 17, 6700 AA Wageningen, The Netherlands, T + 31 (0)317 48 00 84, E info.wfbr@wur.nl, www.wur.eu/wfbr. Wageningen Food & Biobased Research is part of Wageningen University & Research.

Contents

Definitions

In this document, the following definitions and abbreviations are used.

- **API**: Application Programming Interface; this is the software interface of the software library in which the Bayesian network has been developed and is loaded. The API is necessary to communicate between the different software components and the Bayesian network library.
- **BBN**: Bayesian Belief Network, see Model
- **Client**: The fruit company, i.e. Driscoll's B.V.
- **GUI**: graphical user interface; the part of the software that displays information on a screen for the user to interact with.
- **Model**: The Bayesian Belief Network that predicts the quality of fruits.
- **Software provider**: The company that develops the integration of the model in the software environment.
- **Web API**: Web-based API that can be used to input and output data to and from the Bayesian network in a web-based interface.
- **WFBR**: The research institute, part of Wageningen Research, that has developed the Bayesian network.

Summary

The GreenCHAINge project is carried out in consortium with various companies, and supported by the Dutch Ministry of Agriculture. It is financed by this ministry through a Topsector subsidy and by the participating companies in the consortium through their cash and in kind contributions. The GreenCHAINge project comprises several sub-projects, each focussing on a different fruit or vegetable product. The work reported in this document resides under the sub-project dedicated to soft fruit, and is carried out – independently - by Wageningen Food & Biobased Research (WFBR) in collaboration with Driscoll’s B.V.

The Wageningen Food & Biobased Research institute has developed a Bayesian model for quality prediction of soft fruit, specifically strawberries, based on a variety of input data. The model was developed in the Netica environment and requires the Netica library to be executed. Integration of the model in the existing workflow at Driscoll’s B.V. requires it to be encapsulated by additional software, such as an interface to the user, interfaces to available data sources, and to the model itself. This document describes the requirements and proposed architecture for deploying the model at Driscoll’s B.V., and while requirements and architecture are described in a generic fashion, this report is primarily targeted at stakeholders within Driscoll’s B.V.

1 Introduction

The GreenCHAINge project is carried out in consortium with various companies, and supported by the Dutch Ministry of Economic Affairs. It comprises several sub-projects, each focussing on a different fruit or vegetable product. The work reported in this document resides under the sub-project dedicated to soft fruit, and is carried out by Wageningen Food & Biobased Research (WFBR) in collaboration with Driscoll's B.V.

It has been one of the goals of the 'soft fruit' sub-project (named DP3) to develop a model capable of predicting quality in soft fruit. Research and development of the model was conducted independently at Wageningen Research. The model allows to predict the expected quality of soft fruit down the chain, given a set of input parameters and known conditions for storage and transportation.

The model itself - a Bayesian Belief Network (see Annex 4 for a glance at its content) - was developed in Netica, a dedicated software environment, and requires the Netica library to be executed and generate quality predictions[1]. In order to deploy this model in a corporate environment, additional software development and integration efforts are required. This primarily concerns establishing an interface to the user (GUI), and providing interfaces to available data sources, and to the model itself. These activities are not in the scope of the GreenCHAINge project, but transfer of required knowledge and requirements to achieve deployment of the model is part of this project[2].

This document describes requirements and accompanying (proposed) architecture for the software to be developed, realising an in-company quality prediction system based on the developed Bayesian Belief Network. The proposed approach aims at a user-friendly interaction between the model and dedicated company personnel, and at integration with existing business processes at the client company. While requirements and architecture are described in a generic fashion, this report is primarily targeted at stakeholders within the client company.

This report starts by discussing the front end, including guidelines for a user interface (Chapter 2). The architecture of the back end that encapsulates the actual model is discussed in Chapter 3. Finally, Chapter 4 discusses requirements on different areas of this back end in more detail. When combined, the proposed front and back end make up the complete quality prediction system for soft fruit.

[1] Although running the model in Netica is the most obvious choice requiring least effort, the model could be ported – using a tool developed by Wageningen Research - to other implementations of Bayesian Networks, such as Huggin or Weka. Each of these alternatives has its own advantages and disadvantages, mainly concerning computational performance, memory use, and license cost.

[2] Detailed low-level requirements concerning which data at the client company should be connected to which model input parameters, and exactly how these data should be pre-processed before feeding to the model, are not part of this document. In order to formulate such requirements, more detailed insights are needed regarding the client's software environment and properties of available data. During implementation Wageningen Research could play an advisory role in this process.

2 Front end – User Interface

The purpose of the BBN is to predict the quality of specific batches of fruit based on various parameters. Practical application of the model requires a software environment. This software environment would need to consists of a user interface, that allows a user to interact with the model (entering parameters and visualising the resulting predictions). It would also need to consist of a back end that implements the model, and facilitates a connection between the model and the user interface, as well as to available sources of data (if not all data is provided via the user interface). In this chapter, the user interface is discussed. In the next chapter, the software requirements for the back end are discussed.

The user interacts with the user interface in various ways. We distinguish the following use-cases:

- The user would like to make a quality prediction for a specific batch of fruit using the model.
- The user would like to run 'what-if' scenario's with the model, to explore how parameters and conditions affect quality, and to optimize these to obtain the desired fruit quality.

Based on these two use cases, we envision a user interface with two dialogs. The first dialog (see Figure 1 and associated flow-chart in Figure 2) allows the user to enter a batch number and prediction period (up to 20 days ahead). Based on this information the model predicts quality over time (days), using data (obtained from databases) associated with the batch number. The second dialog (see Figure 3 and associated flow-chart in Figure 4) allows the user to define a fictive (or real) scenario without a batch number, for which the model predicts the quality.

There are several ways to go about implementing the user interface. The proposed interface could be developed as a standalone (browser) application[3], or as an extension to already existing enterprise software. The latter approach seems preferable from a user perspective, since it enables the user to access the BBN functionality within an already familiar software environment. Depending on decisions made during detailed design of the user interfaces, training for users of the interface is required. In all cases users require understanding of the predicted quality as displayed via the user interface, as well as of the parameters they are supposed to enter via the interface.

Of course there are alternative approaches to interfacing with a user, such as providing the model with input via a database (interfaced to the user elsewhere) containing all required data, and collecting the resulting predictions from the model in the same (or another) database.

[3] Facilitating display on PC, tablet or smart-phones

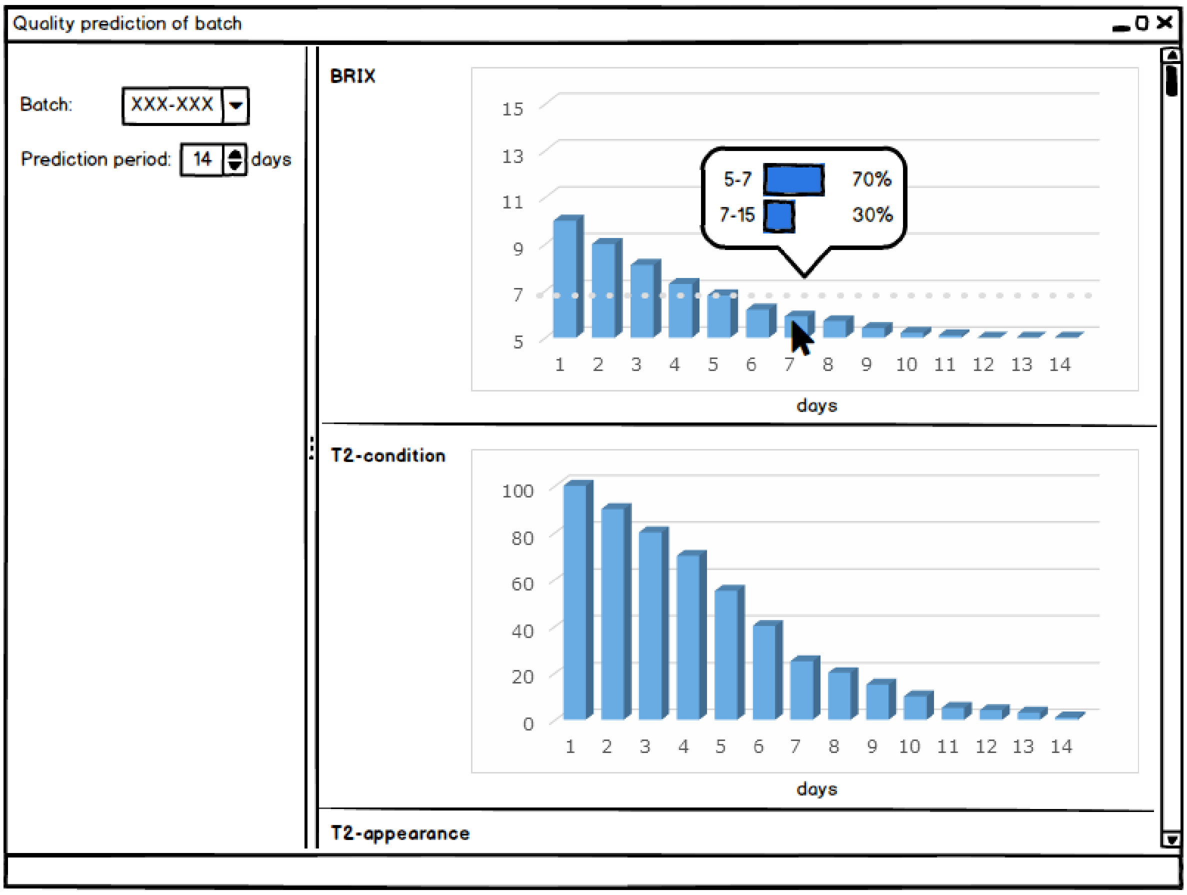


Figure 1 *Mock-up of a screen where the user is allowed to make a quality prediction for a specific batch of fruit using the model. The batch number is used to look up associated data from a database, such as harvest day, region, cultivar. These data are provided as input to the model for generating a quality prediction. In these mock-ups, the resulting prediction is visualised as a bar graph with predicted BRIX and T2-condition values over a period of 14 days. The mock-up also shows how the user can reveal more detailed information from the BRIX graph by a mouse click on a specific date. A pop-up window shows the predicted likelihood (percentage) that the BRIX values of this batch reside in BRIX class 5-7 or 7-15 on this date.*

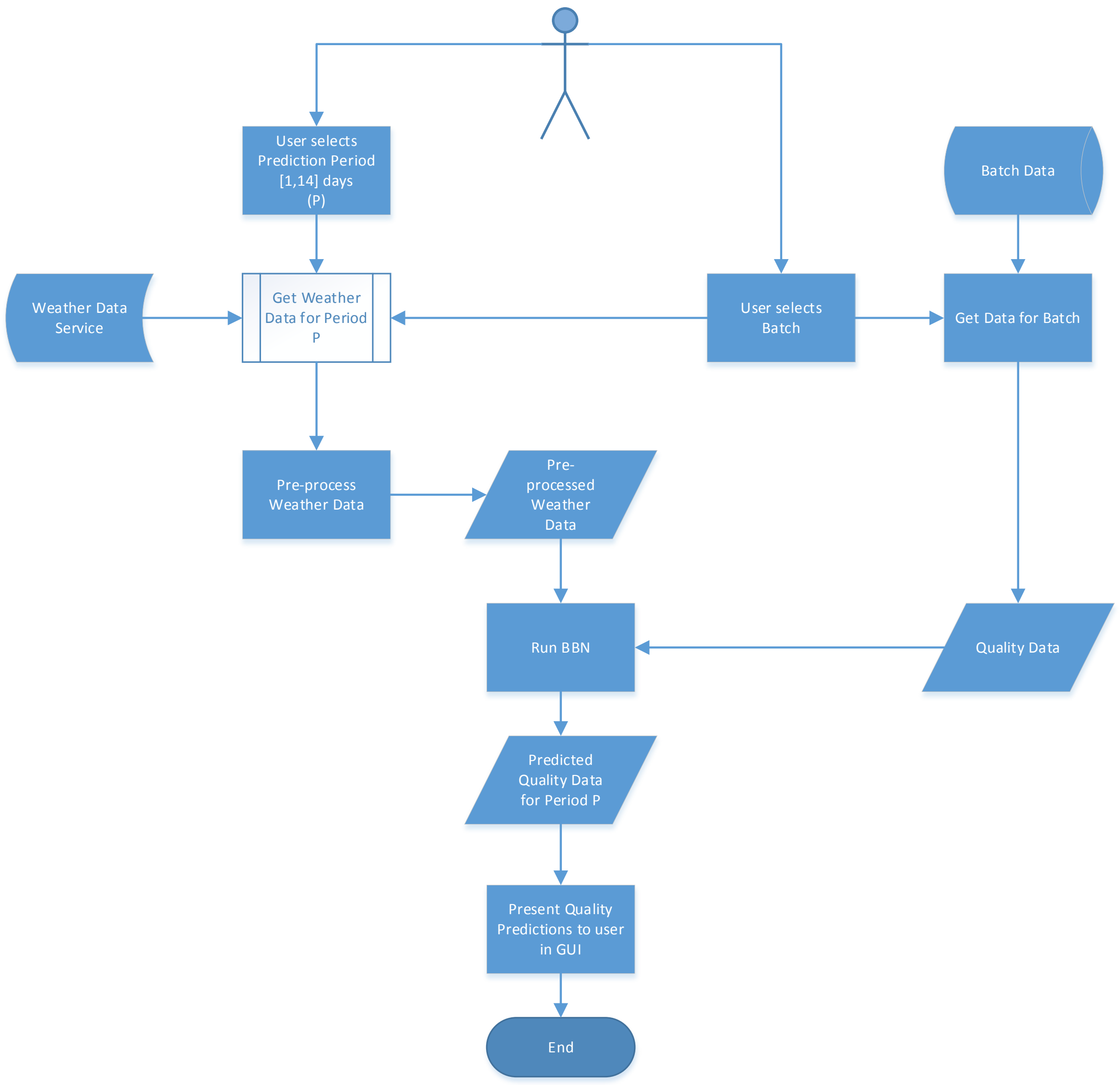
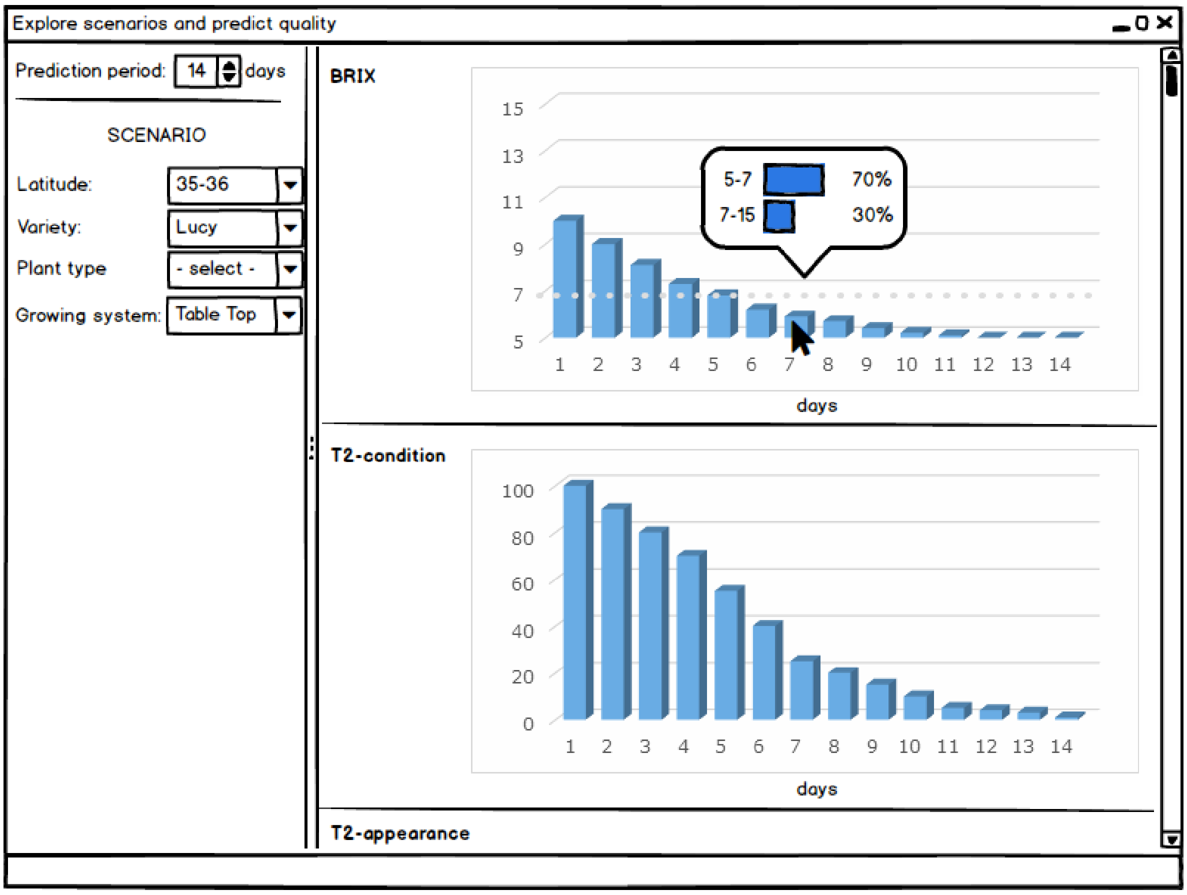

User selects Prediction Period [1,14] days (P)
Batch Data
Weather Data Service
Get Weather Data for Period P
User selects Batch
Get Data for Batch
Pre-process Weather Data
Pre-processed Weather Data
Run BBN
Quality Data
Predicted Quality Data for Period P
Present Quality Predictions to user in GUI
End

Figure 2 *Flow chart associated to the user interface in Figure 1. The user (top) selects a prediction period and batch number. The batch number is used to look up quality data associated to the batch in a database, and passed to the model as input. Weather data regarding the period entered by the user is obtained from a weather service database. Weather data is pre-processed and stored, and then passed to the model as input. Resulting predictions (output) from the model are stored and visualized in the user interface.*



Explore scenarios and predict quality

Prediction period: 14 days

SCENARIO

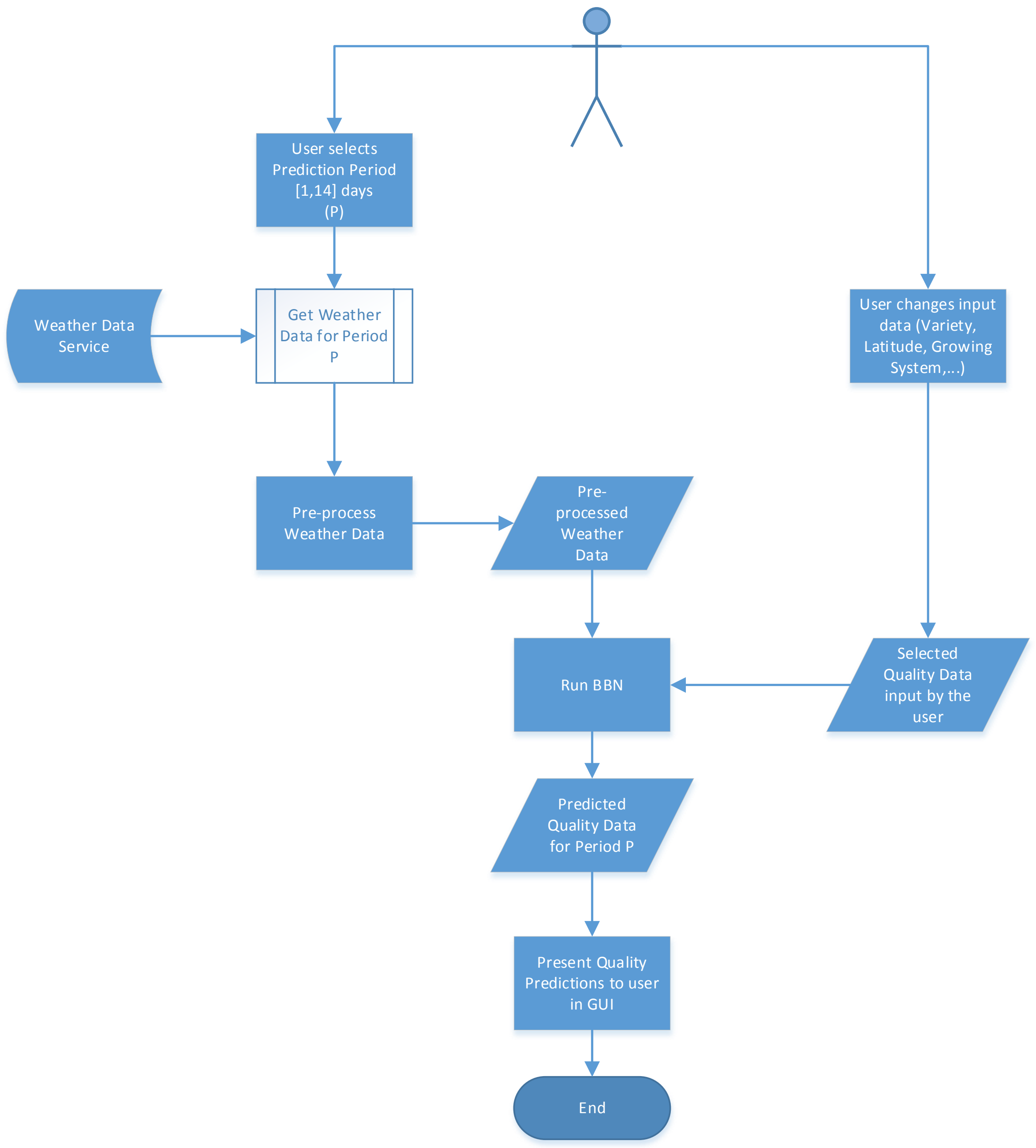
Latitude: 35-36

Variety: Lucy

Plant type - select -

Growing system: Table Top

Figure 3 *Mock-up of a screen in which the user is enabled to run 'what-if' scenario's with the model, to explore conditions that result in the desired fruit quality. The screen shows how the user is allowed to construct various scenarios by entering different values for a set of input parameters (e.g. latitude, variety, plant type and growing system), and generate predictions from these.*



User selects Prediction Period [1,14] days (P)
Weather Data Service
Get Weather Data for Period P
User changes input data (Variety, Latitude, Growing System,...)
Pre-process Weather Data
Pre-processed Weather Data
Run BBN
Selected Quality Data input by the user
Predicted Quality Data for Period P
Present Quality Predictions to user in GUI
End

Figure 4 *Flow chart associated to the user interface in Figure 3. The user (top) selects a prediction period and can freely play around with various input parameters to compose a scenario. The prediction period data is used to look up detailed weather data from a weather service, which is pre-processed and stored before being fed to the model as input. The parameters comprising the scenario are also passed as input to the model. Resulting predictions (output) from the model are stored and visualized in the user interface.*

3 Back end – Architecture

In this chapter we discuss the back end that implements the model, and the software architecture around it (see Figure 5), that allows the model to connect to the user interface, as well as to available sources of data (if not all data is provided via the user interface). The back end allows the model to be provided with parameters obtained from the user interface, allows the model to be executed, and hands over the resulting predictions back to the user interface for display (for example via the API specification in Annex 3). We will assume that a user interface as described in Chapter 2 (or a similar interface) is in place. Whether this user interface is implemented as an extension to existing enterprise software, or as a standalone (browser) application, does not matter for the back end architecture.

The heart of the back end consists of two parts:

1. The BBN.dne file that contains information on both structure and content (values resulting from training) of the Bayesian Belief Network developed for this project[4].
2. The Netica API environment[5] that uses the BBN.dne file to generate predictions on fruit quality. A special Netica license is required for this.

Surrounding this heart, software needs to be developed for turning the model into a web service and allow it to run on a server. This facilitates multiple instances of the model to be approached from user interfaces on different remote computers at the same time (e.g. different users) via a web API. This approach also brings the freedom to run the web service - with the model inside it - on premise, or out-source it in a private cloud.

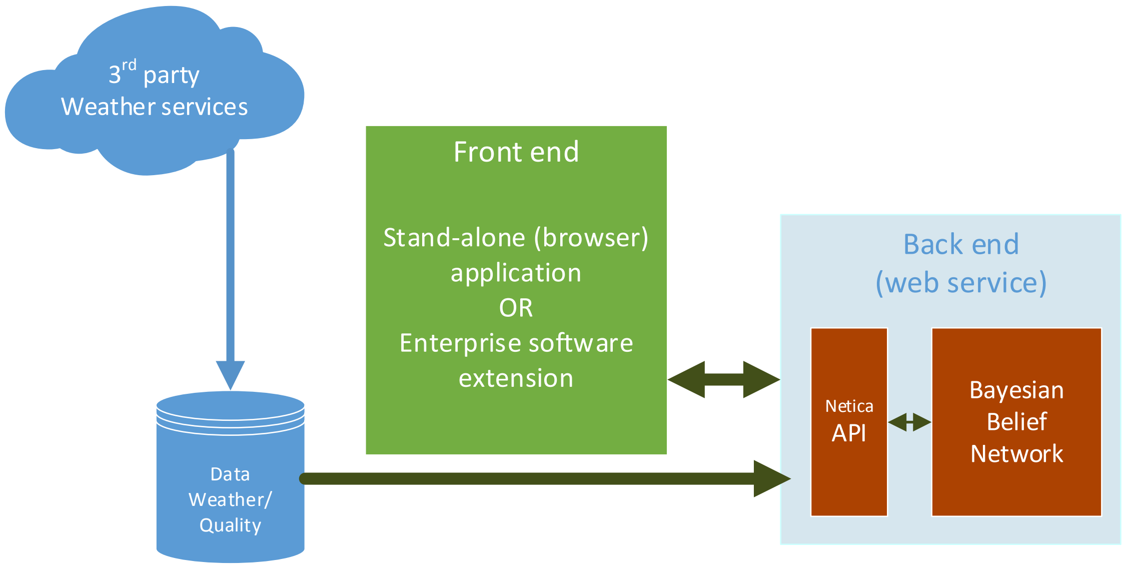


Figure 5 *Visual representation of the proposed architecture.*

[4] The intellectual property of this model will reside under a shared license between Wageningen Research and Driscoll's B.V.

[5] https://www.norsys.com/netica_api.html ; Versions of the Netica API are available for most popular platforms, including Microsoft Windows 95/98/Me/NT4/2000/XP/Vista, Linux, Sun Sparc, Mac (68000, PPC: OS 6 to 9 & OS-X), Silicon Graphics and DOS.

The web service can fulfil several other purposes as well:

1. It can handle interfacing to existing sources of data to be used as input to the model, in case not all data is entered via the user interface.
2. It can take care of pre-processing of data, for instance calculating the sum over multiple values, before feeding it to the model.
3. It can store resulting predictions of the model in a database for later recall and/or push them to the user interface for directly presenting them to the user.

Wageningen Food & Biobased Research has in fact developed web services and interfaces for similar situations. License costs to both Wageningen Food & Biobased Research and the provider of the Netica library would be incurred in this case.

Alternative architectures are of course possible, but seem less suitable for the intended use of the model. For the remaining chapters we will therefor assume the proposed architecture is used.

4 Back end - Requirements

4.1 User interaction

The back end should facilitate a connection from and to a user interface (see Chapter 2). The back end will receive data from the user interface and return the resulting model predictions to it. In Annex 3, an example of an API definition is provided for communication between front end and back end. Alternatively, or additionally, a database approach to input and output for the model could be considered. This is recommended as an addition in case quality predictions generated in the past need to be recalled. These alternatives pose a different requirement to the back end, namely a connection for writing model predictions to these destinations.

4.2 Model interaction

The back end should interact with the model via the Netica API, for which detailed documentation and license information can be found online[6]. After the model file (BBN.dne) is loaded into Netica, it requires initialisation (Netica command) before quality predictions can be generated. After each prediction the model state should be reset (purging content from previous predictions). Depending on the specified workflow for the user interface, the model can be invoked in two ways:

1. The model can automatically recalculate predicted fruit quality triggered by changes of its input parameters.
2. All input parameters can be changed freely (without recalculations), and a separate command can trigger recalculation of the predicted fruit quality using all current parameters.

In order to run the model, specific input parameters will have to be communicated to the model via the Netica API. Annex 1 (Table 1) provides an overview of the input parameters the model is capable of accepting. For instance, climate conditions (e.g. weather or greenhouse climate) during cultivation of the product should be provided to the model for making a quality prediction. Many variables, including climate conditions, require a form of pre-processing to turn them into suitable input for the model (for details see Annex 2). Pre-processing typically consists of calculating an average, minimum or maximum value over a certain period of time or growing degree hours. Pre-processing should also address the units of measurement for each variable, and provide conversion if required. It is critical that pre-processing in the back end takes place according to the same algorithms involved when the model was trained (see Annex 2). These algorithms are developed by Wageningen Research and are available on request.

Depending on decisions made in the design process of the final user interface, the model's output variables need to be interpreted for visualisation in the user interface (e.g. a graph or pop-up with details on likelihood of a prediction). This interpretation can take place in the back end, or in the user interface. Annex 1 (Table 2) provides a description of the model's raw output variables, for this purpose.

4.3 Data sources

When data sources with relevant input for the model exist (e.g. weather, grower, supply chain, and quality data) it seems preferable to access these directly, rather than require the user to enter these data via the user interface. Data in these sources can be dynamic in the following ways:

1. Automatically updated by a measurement system, for instance in case of weather data

[6] https://www.norsys.com/netica_api.html

2. Automatically updated by other business processes, for instance data/measurements from the supply chain
3. Manually updated, by providing data in an Excel sheet or by entering data in a user interface, for instance by quality officers who use an app to provide their quality assessment for a specific batch of fruit.

If databases for these types of data do not exist yet, it is recommended to create them.

Weather data is probably most easily pulled from an external weather data web service (not applicable in greenhouse conditions). It is recommended that a copy of relevant weather data is stored in a client owned database, to be able to recall weather parameters if required. Raw weather data obtained this way requires pre-processing to obtain a calculated parameter for a specific period, before it can be fed to the model (see also Section 4.2). This should take place in the back end on raw weather data that is already stored in a local database. The result can be used with the model to generate quality predictions.

In certain cases – i.e. when the product has not yet been harvested - quality predictions can be made beyond the (presumed) harvest date, using a mixture of *measured* (up to current date) and *predicted* climate data (e.g. weather service forecast from current date). Pre-processing should take place on these mixed data, to be able to provide the model with one climate parameter for the time span of the requested prediction. When time passes and *measured* data becomes available where *predicted* data was previously used, these can be pre-processed and used to update model predictions, hereby improving prediction accuracy.

4.4 Model learning and improvements

Up to this point the model has been described as a static part within the architecture. When used statically, the model in its current state - as supplied by Wageningen Research - is already capable of predicting quality of strawberries. However, it can be improved via a learning mechanism. Presenting new data to the model allows it to learn. These data may be new batches of the given varieties from the given countries. In this case a growing variation of measured values will be obtained – both in the field of quality variables and climate conditions. Hence, when these data will be fed to the BBN, the predictions will be more accurate and cover a broader domain (for example, more extreme climate conditions). Netica supports this learning functionality – new data can be loaded into the model using the library of the Bayesian network tool. In this case, re-validation of the model is not necessary, but may be interesting as to find out how much performance was gained in the learning process. If learning is desired, the back end web service should support this, by facilitating means to supply a file with training data to Netica. Netica learning requires training data - new cases with quality data - to be provided in a specific file format[7].

More extensive changes, such as extending the model for use with a different fruit (e.g. raspberry), or incorporating new types of data, requires more scientific efforts. Such changes require the model to be re-validated, especially for the concerned quality variables, as the changes and new data affect the performance. Reassurance is needed that performance on different quality variables is still sufficiently high. At the end of this scientific process a new version of the model would be obtained that can replace a previous version of the model in the back end of the client.

[7] The training data file should be formatted as a tab-delimited text (ASCII) file. The file requires a header opening at the first line with // ~->[CASE-1]->~, followed by column names for each input and output variable. Following the header, each column can contain multiple values (in rows) for a variable. A * symbol can be used in place of missing values. More details (e.g. how to invoke learning) are described on the Netica website: https://www.norsys.com/netica_api.html.

5 Conclusion

A model for predicting quality in soft fruit, specifically strawberries, has been developed. In the previous chapters various aspects of a front end and back end for this model have been discussed. This information combined allows to take the next step in deploying the model at the client’s company. To summarize, the following resources are required to incorporate the model in daily workflows:

- Netica or other BBN implementation to run the model
- The model itself (BBN.dne)
- Databases that supply input parameters for the model (environmental conditions, growing conditions)
- Database that collects model output (predictions)
- Front end software containing user interface
- Back end software to invoke model, connect to data sources, perform pre-processing, connect to user interface
- IT support / maintenance for front and back end as well as data sources
- Personnel trained to operate the model via the user interface

Some of these resources are readily available, others - namely front and back end, and (some) databases - need to be developed. This requires involvement of a software developer, and possibly a UI designer, and software tester. Detailed decisions about connecting model input parameters to available data at the client company are best made with Wageningen Research involved, as are detailed decisions about interpretation and visualisation of the model output in the user interface.

Annex 1 Model input and output parameters

Table 1 Model input parameters.

Type	Parameters	Indication of pre-processing
Dynamic growing parameters		
	Temperature, rel. humidity, vapour-pressure deficit (VPD)	Maximum/minimum/average during day/night/both for several periods of growing degree hours. VPD needs to be calculated, based on temperature and humidity.
	Temperature difference (between min and max)	For several periods of growing degree hours
	Radiation, CO2	Average for several periods of growing degree hours
Static growing parameters		
	Growing system, growing medium, plant density	-
	Variety	-
	Grower	-
	Planting type	-
	Water source	-
	Latitude	-
Time related parameters		
	Duration of growth	Production date minus planting date
	Planting date	-
	Day number, week number, year (all of production date)	Date to day/week number conversion
	Average time between picking and cooling	Average difference
Quantity parameters		
	(Corrected) Week production	Week production including culls
	Uncorrected week production	Week production excluding culls
	Cumulative week production	Cumulating

Table 2 Model output parameters.

Parameter
T2-condition
T2-appearance
PFQ-score
PFQ-score (calculated)
BRIX
Dry bruising
Wet bruising
Rot
Not fully coloured
Cracked skin
Overripe
Severity score
Severity score (calculated)
Visibility score (8th day)
Percentage Rot Fruit
Percentage Wet Fruit
Percentage Medium Defects
Percentage NTL (none to light)

Annex 2 Pre-processing of climate data

Author: dr. S.K. Schnabel
Affiliation: Mathematical and Statistical Methods, Wageningen Plant Research

Goal
The goal is to summarize the climate related variables (temperature, relative humidity, radiation, CO_2 level, water damp deficit, difference between the lowest night temperature and the highest day temperature) into mean/minimum/maximum values during daytime/night/whole day for different periods of (cumulative) growing degree hours (GDHs) preceding the harvest date: 500 GDHs, 2000 GDHs, 5000 GDHs, 7500 GDHs, 10000 GDHs, 12500 GHDs, 15000 GDHs.

Remark
Some of these steps might be obsolete or different depending on the software implementation. We did some data cleaning, including making sure that data from different sources follow the same date/time format, etc.

Additional data
In addition to the sensor data from the locations we also needed the daily sunrise/sunset moment of the respective location (or the closest place available).

Calculations

A. ***Temperature/Relative Humidity/Radiation/CO2***:

1. Determine per batch based on the production date (= harvest date) the GDHs respective period.
2. Determine which climate measurements are falling into this period.
3. Use the sunrise/sunset variable to sort out which measurements are taken during daytime and which are during night.
4. Summarize these subsets of the data into the mean (all climate variables above), minimum and maximum (only temperature and relative humidity) for the given period and part of the day.

For one batch this will result in 7*3*3 variables for temperature/relative humidity alone: we used seven GDHs periods, we summarize into mean/minimum/maximum and for daytime/night/whole day.
For radiation/CO_2 we limited the creation of variables to only the mean for the whole day, resulting in seven variables per batch each.

B. ***Vapour pressure deficit (VPD)/Water Damp Deficit (WDD):***

As a first step for every time point with data for temperature and relative humidity we calculated the vapour pressure deficit/water damp deficit as:

Xsat=exp(23.4795-(3990.56/(T+233.833)))/162
X=Xsat*RH/100
WDD = Xsat –X

With T the temperature in °C and relative humidity RH measured in g(water)/kg(air) (formula from an internal communication).
Then we proceeded with the same steps as mentioned for temperature above resulting in as mentioned for temperature above, resulting in 7*3*3 variables per batch.

C. ***Temperature difference (night/day):***

1. Determine per batch based on the production date the respective period.
2. Determine per day that falls into the respective GDHs period the difference between the maximum day temperature and the minimum night temperature.
3. Summarize these subsets of the data into the mean for the given period.

This will result in seven variables per batch for temperature difference.

Alternative approaches are also possible, e.g. creating all variables for all dates in a time period of production and then assign to the respective batches.

Annex 3 Example API

The following API definition gives two possible definitions for API methods needed for communication between front end (GUI) and back end (running the BBN). Additional API methods will be needed to request the climate variables from the database containing weather measurements and predictions, and from the database containing data for each batch.

`GET` `predictQuality`

Request body; a JSON map with the following key values:

- `startPeriod`: Date – **optional** - The first date of the period for which predictions are requested. If no start date is specified, the current date is used.
- `period: Int` – **optional** - The length of the period in number of days (between 1 and 14 days). If no period length is specified, the maximum length (14 days) will be used.
- `batch: String` – **optional** - A string identifying the batch for which the quality prediction is requested. The batch number is used to get the growing conditions (e.g. latitude, variety,...) that should be used as input to the model.
- `inputVariables: [variableID : Any]` – **optional** - If no batch is specified, input variables such as latitude and variety can be defined and set using the API. The format should be a map containing the IDs of the variables as keys and the value of the variable. The format of the value can be either a Float, Double, Integer, or String (as stateID).
- `outputVariables: [variableID]` – **optional** - An array containing the variable identifiers for which the predicted value is requested. If no output variables are specified, the predictions for all output variables will be returned.

Result:

A map with the following structure:

- `day` – The day number (after the start date).
- `outputValues` – A map with the following structure:
    - The identifier of the variable as key with the following map as value:
        - `value` – The prediction when the value is a number. This is the average of the normal distribution fitted onto the probability distribution.
        - `states` - A map with the state identifier as key and the likelihood (a percentage) as value.

Each day in the specified period will have an entry containing its predicted values.

Example:

Request Body:

```
{
        "startPeriod": "2018-12-04",
        "period": 3,
        "batch": "XX=XXXX=XX",
        "outputVariables": ["BRIX", "T2Condition"]
}
```

Result:

```
{
        "day": 1,
        "outputValues": {
                "BRIX": {
                        "value": 5.8,
                        "states": {
                                "5-7": 70,
                                "7-15": 30
                        }
                },
                "T2_Condition": {
                        "value": 40,
                        "states": {
                                "0-60": 70,
                                "60-80": 30
                        }
                }
        },
        "day": 2,
        "outputValue": {
                "BRIX": {
                        "value": 5.8,
                        "states": {
                                "5-7": 70,
                                "7-15": 30
                        }
                },
                "T2_Condition": {
                        "value": 30,
                        "states": {
                                "0-60": 93,
                                "60-80": 7
                        }
                }
        },
        ...
}
```

GET `inputVariables`

Returns a list of variables containing the identifier of the variable, its human-readable name, and the possible states of that variable. The request does not accept parameters, it just returns a list of the input variables.

Result:

The result consists of an array of a map for each input variable. The map has the following keys:

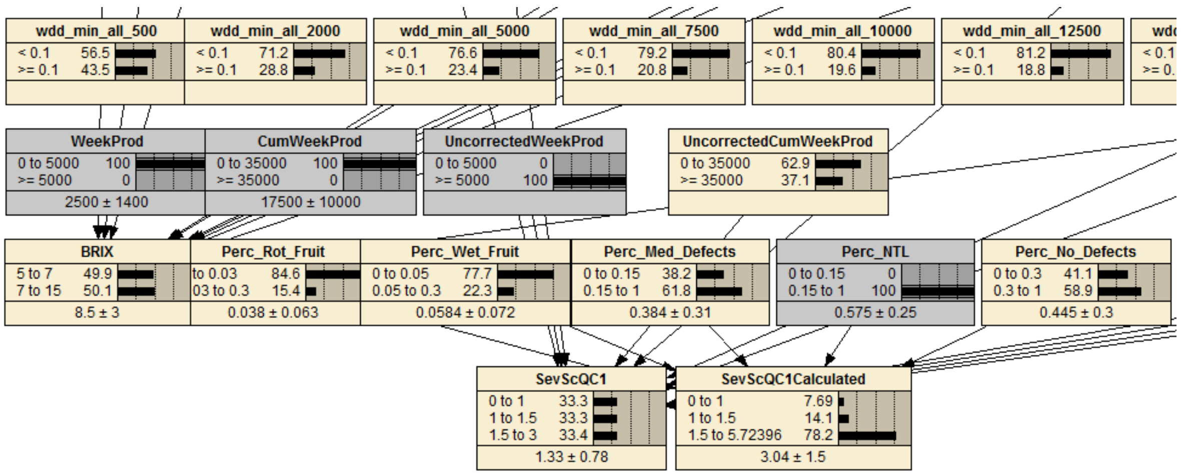
- `id: String` – The identifier of the variable.
- `name: String` – The human readable name of the variable.
- `unit: String` - The unit of the variable if applicable.
- `states: [String]` – An array containing the identifiers of the different states possible for the variable.

Example:

```
[
        {
                "id": "Latitude",
                "name": "Latitude",
                "unit": "degrees north",
                "states": ["35-36","36-50","50-51.55","51.55-51.6"]
        },
        {
                "id": "Variety",
                "name": "Variety",
                "states": ["Lusa","Scarlet"]
        },
        ...
]
```

Annex 4 A glance at the model

The complete BBN model is too large and complex to visualize easily, but a small portion of it is displayed in the figure below. It shows various nodes and the interconnections (arrows) between them. It shows how input parameters determine the value (state) of some nodes, for instance the parameter for week production named *WeekProd*, and how the state of some nodes is determined by their connections to others. It also shows several output nodes containing the resulting prediction, for instance the one predicting the BRIX value (bottom left).



wdd_min_all_500 < 0.1 56.5 >= 0.1 43.5
wdd_min_all_2000 < 0.1 71.2 >= 0.1 28.8
wdd_min_all_5000 < 0.1 76.6 >= 0.1 23.4
wdd_min_all_7500 < 0.1 79.2 >= 0.1 20.8
wdd_min_all_10000 < 0.1 80.4 >= 0.1 19.6
wdd_min_all_12500 < 0.1 81.2 >= 0.1 18.8
WeekProd 0 to 5000 100 >= 5000 0 2500 ± 1400
CumWeekProd 0 to 35000 100 >= 35000 0 17500 ± 10000
UncorrectedWeekProd 0 to 5000 0 >= 5000 100
UncorrectedCumWeekProd 0 to 35000 62.9 >= 35000 37.1
BRIX 5 to 7 49.9 7 to 15 50.1 8.5 ± 3
Perc_Rot_Fruit to 0.03 84.6 03 to 0.3 15.4 0.038 ± 0.063
Perc_Wet_Fruit 0 to 0.05 77.7 0.05 to 0.3 22.3 0.0584 ± 0.072
Perc_Med_Defects 0 to 0.15 38.2 0.15 to 1 61.8 0.384 ± 0.31
Perc_NTL 0 to 0.15 0 0.15 to 1 100 0.575 ± 0.25
Perc_No_Defects 0 to 0.3 41.1 0.3 to 1 58.9 0.445 ± 0.3
SevScQC1 0 to 1 33.3 1 to 1.5 33.3 1.5 to 3 33.4 1.33 ± 0.78
SevScQC1Calculated 0 to 1 7.69 1 to 1.5 14.1 1.5 to 5.72396 78.2 3.04 ± 1.5

To explore the potential of nature to improve the quality of life

Wageningen Food & Biobased Research
Bornse Weilanden 9
6708 WG Wageningen
The Netherlands
www.wur.eu/wfbr
E info.wfbr@wur.nl

Report 1879

Confidential