ANALYSING BRAND EQUITY IN THE DUTCH FLOWER AUCTIONS

Creating a performance measure for the flower grower

Jan Vrieler
YSS 82312

Student number: 950630-910-060
Chair group: MCB

Supervisors
Frans Verhees (MCB)
Cagatay Catal (INF)

Table of contents:

Table of figures:

Table of tables:

1 Introduction

1.1 The Dutch Flower Auctions

Holland is the world's biggest exporter of flowers. With a share of 59% of all cut flower exports in the world, it distributes more than half of the entire exported volume around the world (Batt, 2001). The flowers are not only distributed internationally, but are also sold at the Dutch flower auctions. The Aalsmeer Flower Auction, owned by Royal FloraHolland, is the biggest flower auction in the world. From this auction, exports attribute to a total sum of 794 million euros on a yearly basis. Their yearly report of 2016 shows a total turnover of 2,7 billion euros in cut flowers alone. While a physical auction place might seem outdated when everything can be done digitally, almost half of the turnover is still exchanged by regular auction (Royal FloraHolland, 2016).

The workings of the auction itself are very straightforward. The buyers can see most details of the flowers, like grower, type of flower, amount of flowers on a cart, amount of flowers in a bucket, length of the stems on their computer screens at the auction. The price of the flowers is shown on the auction clock. The price will gradually decrease until one of the buyers decides that the product is worth the shown price, and the auction will stop. The highest bidder will receive the auctioned flowers (Royal FloraHolland, 2017).

Like most other products, the price of auctioned flowers is dependent on supply and demand. If supply and demand were the only factors influencing the price of a particular product, it follows that two similar products on the same day will receive the same price. However, that is not the case. In reality, two carts with the same roses, that only differ in their suppliers, receive different prices. Occasional variance can be ascribed to the reaction-based auction system. In reality, some suppliers receive a structurally higher or lower price than their competitors, which rules out these random variations as a cause. With the only real difference between the received prices being the grower, it seems that the fact that a grower delivered the product influences prices.

1.2 Price differences

The difference in received prices might be caused by the name of the grower being perceived as a brand. Aaker (1997) describes brand as a name of, or association with, a product that differentiates it from its competitors. This differentiation makes a product recognisable, but it can also alter the market value of that product. Keller (1993) explains this effect as 'brand equity'. He explains that brand equity is the positive or negative effect brand has on the value of an associated product as perceived by the consumer. He continues by naming perceived quality an important contributor to brand equity. Steenkamp (1990) describes perceived quality as the fitness of a product to fulfil the expectations a consumer has before consuming a product. The better it fulfils those expectations, the higher the perceived quality. According to Steenkamp, a brand can influence perceived quality.

1.3 Research goal

When the aforementioned theories are combined, it can be reasoned that brand equity is a consequence of how a product is perceived by the customer. A product perceived as high in quality will likely have a higher brand equity, and will therefore receive a higher price. Likewise, a product that is perceived as low quality will likely have a negative brand equity, and receive lower prices. Because brand equity and perceived quality are related, it follows that a producer that invests more effort in the quality of his product will probably have a higher brand equity.

In the specific case of growers, high quality flowers will result in higher brand equity, and thus higher revenue. The goal of this research is to develop and provide performance measure for the grower. Knowing his performance in the market can trigger the grower to invest in the quality of his product, or cause him to maintain current production standards. This heightened focus on delivering higher quality products can act as a stimulant to flower quality in general.

1. 4 Problem statement

Unfortunately, currently there is no way of assessing this performance. Even though the grower can reap great benefits by monitoring the performance of his brand. This research aims to deliver a simple to implement performance monitoring tool for the grower. By developing this monitoring tool this research will alleviate the non-existence of monitoring tools, and provide monitoring tool for the grower.

1.5 Research questions

The aim of this research is to create a performance measure of the delivered product. The research question is therefore:

RQ1: How to measure the brand equity of a grower based on the differences in received price on the market?

Before trying to analyse brand equity in the flower auctions, it has to be clear if the theoretical background of this research holds in practice. In other words: can brand equity be derived from auction data? If brand equity can be derived from this data, how large is the impact it has on price? This leads to the following question:

RQ2: To what extent do (names of) growers influence the price of an auction product in the Dutch Flower Auctions?

2. Theoretical Framework

This chapter will be divided in two parts, the theories that support the existence of brand equity based on perceived quality, and the way this relation presents itself in the auctions. In the first part, brand equity and perceived quality will be further elaborated upon. Thereafter, these theories will be combined to show their applicability to the situation of the flower auctions.

2.1 Theory

2.1.1 Brand equity

In his book, Aaker (1997) describes a brand as follows: ‘A brand is a name and/or symbol (such as a logo, trademark or package design) intended to identify the goods or services of either one seller or a group of sellers, and to differentiate those goods or services from those of competitors. A brand thus signals to the consumer of the product the source of the product, and protects both the consumer and producer from competitors who would attempt to provide products that appear to be identical’ (Aaker, 1997). In other words, a brand distinguishes a product from its competitors based on a name or a logo. This is beneficial to the consumer because they can distinguish between similar products based on their brands.

According to Keller (1993), ‘a brand is said to have positive (or negative) customer-based brand equity if consumers react more (less) favourably to the product, price, promotion, or distribution of the brand than they do to the same marketing mix element when it is attributed to a fictitiously named or unnamed version of the product or service’. Essentially, this means that a brand can have a positive or negative effect based on how a product is perceived by the consumer. This effect is defined as the difference in reaction between the branded product, and a similar (or identical) product without that brand. This positive or negative effect on the price is called ‘brand equity’.

Brand equity is defined as follows: ‘brand equity is a set of brand assets and liabilities linked to a brand, its name and symbol, that add or subtract from the value provided by a product or service to a firm and/or to that firm’s customers’ (Aaker, 1997). This definition elaborates further upon the effect of a brand on the value of a product. Brand equity is the term for the added or lost value of a product when it is associated with a certain brand. Following the definitions above, brand equity is a consequence of a product having a brand associated to it. The brand can have a positive or negative brand equity, which in turn influences the value of the product associated with that brand in the eyes of the consumer. According to Aaker (1997) the perceived quality of a product associated to a brand is an important contributor to the brand equity of that brand.

2.1.2 Perceived quality

Steenkamp (1990) defines perceived quality as follows, 'Perceived product quality is an idiosyncratic value judgment with respect to the fitness for consumption which is based upon the conscious and/or unconscious processing of quality cues in relation to relevant quality attributes within the context of significant personal and situational variables'. In other words, perceived quality is the extent to which a product is judged to be able to fulfil a consumer's expectations of that product. This judgement is based on product characteristics that indicate the ability of the product to fulfil those expectations. These characteristics are better known as quality cues.

Quality cues and quality attributes

Steenkamp (1990) explains quality cues as indicators of (un)desired traits in a considered product. The quality attributes are the traits that are being evaluated. There is no way for the consumer to know the true state of a quality attribute before consumption, he can only estimate them by use of quality cues. In Steenkamps words 'Quality cues are what the consumer observes, quality attributes are what the consumer wants'. In his article, Steenkamp states that brand is a possible quality cue for a product.

Steenkamp (1990) states that the true state of an attribute cannot always be ascertained after consumption. For instance, even after experiencing a product, it might still not be clear if it has been produced sustainably. These attributes are called credence attributes. Other attributes can be regarded as experience attributes. The state of these attributes can be perceived after use of the product. Taste, or in case of roses: lifespan in the vase, cannot be ascertained by use of quality cues, as these cues are only used prior to consumption. After experience with the product however, that attribute of a product is clear to the consumer, and the experience can be a cue in subsequent consumption situations.

Ajzen and Fishbein (1975) described three ways a belief about a product attribute can be formed: descriptive, informational and inferential belief formation. Descriptive beliefs come from direct experience with product, informational beliefs can come from any informational source, and inferential beliefs are formed by using descriptive beliefs and inferring different beliefs from them. An example of inferential belief formation could be 'this product was made in Germany, Germans are known for durable production, so it must be durable'.

2.1.3 The influence of perceived quality on brand equity

Based on the last paragraph, the connection of brand to perceived quality can be characterised as an inferential belief. The product belief; 'this product was made by that specific company, so it must be of high quality' can be derived from a positive experience with that product, and the informational belief that it was produced by that specific company.

Yet this belief still plays a part as a quality cue. The assumption of quality is 'inferred' by connection previous experiences with the company to a product. Therefore, the perceived quality of the product based on the brand of that product is an inferential belief.

This belief can be used as a quality cue when assessing the product in the next consumption situation. This also works vice versa. Having a bad experience with a product, and in extension the producing company, might make the consumer reluctant to purchase a product, for fear of receiving low quality.

This can be linked to the connection between brand equity and product quality. The higher the quality of a product, the more likely it is to fulfil the expectations of the consumer. The more positive the consumer experience, the more highly will the consumer rate the brand of that product. When the brand is rated highly, the expected quality of the product is increased in subsequent consumption situations. In turn, this causes the brand equity of that product to grow, making it more likely that the consumer is willing to pay more. This rise in brand equity will increase the producing company's profits.

2.2 Research context

When the definition of brand is applied to the flower auctions, it seems that the name of a grower can indeed be seen as a brand. The buyers are able to see the name of the grower on the auction clock. This name distinguishes the grower's flowers from the others. This communicates a difference between the product of a specific grower and the products of the competition.

In the case of flower auctions, the theory of quality cues and attributes can be applied to the products. After direct experience, the buyer forms a descriptive belief about the quality of the flowers. The quality of the flowers cannot really be ascertained until the product has been consumed, which makes it an experience attribute. A descriptive belief of the quality is formed after use. This belief can be used to create an inferential belief about the quality of the grower's products. I.e. 'this grower has delivered a high quality in the past, so his products must be of high quality'. In that case the grower is a cue for the quality assessment of new products.

A positive view of the product brand can then function as a quality cue for the product, which in turn increases the perceived quality of the product. This increase will add to the brand equity of the grower. It follows that the higher brand equity increases the value of the product. When the products are sold at a higher price, then revenue will also increase.

3. Method

This chapter will take the theories given in the previous chapter, and combine them to provide a way to utilise and visualise the given data. First, a description of the data and its preparation will be given. Second, the applicability of the aforementioned theories on the data will be discussed. Finally, a description of the management tool creation process will be given.

3.1 Data description and preparation

The dataset available for research is an anonymised dataset consisting of auction data. The data contains details on 11.339 transaction records for one particular rose species named 'Red Naomi'. Each transaction contains information about; sale location, sold number of units, number of roses per unit (APE), product code (VBN), species, quality code, length, country of origin, color, sale price, grower and delivery date. There is quite some data missing that is vital to a complete analysis. The data only contains information on the demand for the product. Only the roses bought are recorded in the data, not the number of roses that were offered for sale. This means that the supply of the product cannot be taken into account during the analysis, and the effect of demand is measurable, but it cannot explain variance in price.

To make the data ready for use, redundant variables were omitted from the dataset. For instance, the colour of all the flowers is red, all flowers are of the species Red Naomi, and thus they all have the same product code. Theses variables are not relevant, and were therefore deleted.

To analyse brand equity, it is important to know which sales record relates to which grower. Unfortunately, for quite some records the grower was missing. These records have been deleted, because they cannot be used in this research. Being able to monitor brand equity while using total sales on a certain date, means that the date of a sale is an essential part of the analysis. A record with an unknown date is therefore of no interest to the analysis. These records have been deleted. Some records showed sales where no roses had been delivered, but a price was still recorded. These records have also been deleted. Last, some growers made very minimal deliveries. Analysing a grower with too few records will not result in reliable results because of a high standard error in the results. Therefore, all growers with less than 100 records have been omitted. All suppliers from other countries than the Netherlands had less than 100 records, as a consequence, the variable country was deleted because it now only featured the Netherlands as country of origin. The variable 'quality' has 5 different attribute levels; unknown, B (average), EX (extra quality), I (first harvest) and S (super quality). The quality of I is uncertain, because it is the first harvest. Apart from I, the quality levels are listed on increasing levels of quality.

3.2 Analysis

The effects of the theories mentioned in the previous chapter manifest themselves in the differences in price. Brand equity alters the value of a product, which alters price. To get a useable picture of brand equity, its effect on price will need to be quantified. An analysis that pinpoints the underlying factor that influence price will need to be conducted. Knowing the underlying variables that determine price will enable analysing these variables, and help isolate the effect of brand on price. To achieve this outcome, a multiple linear regression will be used.

3.3 Relevant variables

The analysis will use four independent variables, that predict the dependent variable price. The independent variables used are; length, amount bought on date per 10.000, grower and quality. The coefficients of the regression analysis explain the number of unique variance explained in the model by a variable. Therefore, length and quality were chosen as variables. When these variables are included, the fact that a grower delivers longer roses, or higher quality, will not be taken into account in the coefficient that represents that grower.

The grower variables and variables representing quality will be split into dummy variables so they can be entered into the analysis. Because these dummy variables were derived from a nominal variable, entering all into the regression analysis will result in perfect collinearity. Omitting one variable from the dataset will prevent this, but this will cause the coefficients of the other dummy variables to be relative to the deleted variable. The deleted variable thus serves as a point of reference for all other related variables. This is caused by the split into dummy variables. This split causes all dummy variables to be perfectly collinear to each other. For this reason, one needs to be omitted from the variables. This variable then automatically becomes the reference point for all other related dummy variables.

4 Building the tool

This chapter describes the development process of the analysis tool. It discusses the requirements, design, implementation and testing phases of the tool. The words tool and program are interchangeable in this chapter. This chapter makes use of UML (Unified Modelling Language) diagrams to enable easier interpretation.

4.1 Requirements

This chapter discusses the requirements set for the program in order to work on them properly. These requirements are divided into two types of requirements: functional and non-functional. Functional requirements describe what a program should do, while non-functional requirements describe how the program works.

The scope of the program is to provide an analysis tool that enables growers to monitor the performance of their brand. It is expected that these growers aren't experienced with statistical procedures and analyses.

Therefore, the program should be able to handle the statistical analysis without any choice by the user that requires knowledge of statistics. It follows that the general user will also not know how to interpret coefficients and F-statistics. Therefore, the program should output a simple and understandable artifact, that is easily interpretable for any user. Next, it is important that the tool should be able to stand on itself. There are many statistics packages that require payment or subscription in order to use them. The tool should be free to the grower. For this reason, it shouldn't rely on third party programs in order to be able to function properly. Instead, it should use a freely available software to run the analysis. An exception on this rule is the spreadsheet editor Excel, which is the format in which the data will be entered into the program. All this leads to the following requirements:

4.1.1 Functional requirements:

R1: The program shall analyse auction data, and perform a statistical analysis on them.

R2: The program shall take an Excel document as input format for its data.

R3: The program shall deliver its output in a separate window.

R4: The program shall not rely on third-party programs

R5: The program shall not ask the user for anything that requires knowledge on statistics.

R6: The program shall perform a statistical analysis of the data automatically.

4.1.2 Non-functional requirements:

R7: The program shall be written in the programming language Java.

R8: The program shall make use of the WEKA library for Java to run the analysis.

R9: The program shall be user friendly.

R10. The program shall apply best practices and design patterns whenever possible.

4.2 Design

This chapter highlights how the tool has been designed from a programmer's perspective. It contains multiple UML-diagrams that show how the tool functions in order to comply with the requirements.

4.2.1 Use case diagram

A program can be used in multiple ways, and can have multiple actors interacting with it. The use case diagram models the system from a user's perspective. In this case, there is only one actor that interacts with the tool. This paragraph contains the use cases for the program and their descriptions.

Use case:	enter file

Actor:	user
Description:	the user enters a file into the program
Exceptions:	the file has the wrong format

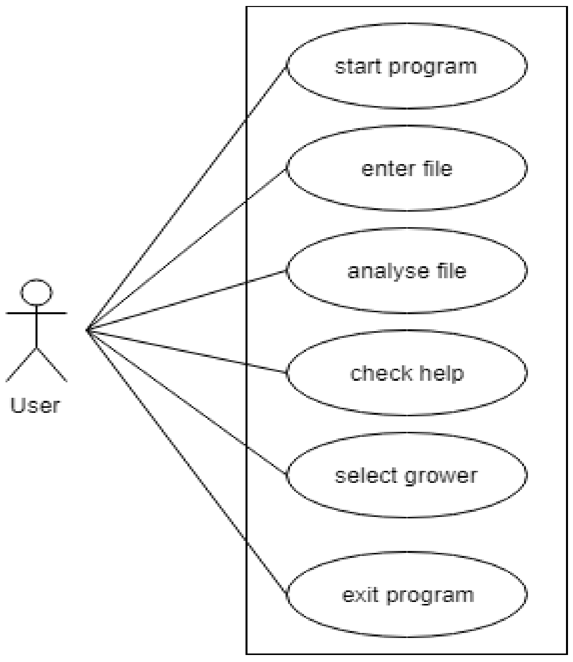
Use case:	select grower
Actor:	user
Description:	the user selects a grower
Exceptions:	the file is not available, the user doesn't understand the program

Use case:	ask help
Actor:	user
Description:	the user consults the 'help' screen
Exceptions:	none

Use case:	exit program
Actor:	user
Description:	the user exits the program
Exceptions:	none

Use case:	open program
Actor:	user
Description:	the user starts the program
Exceptions:	none

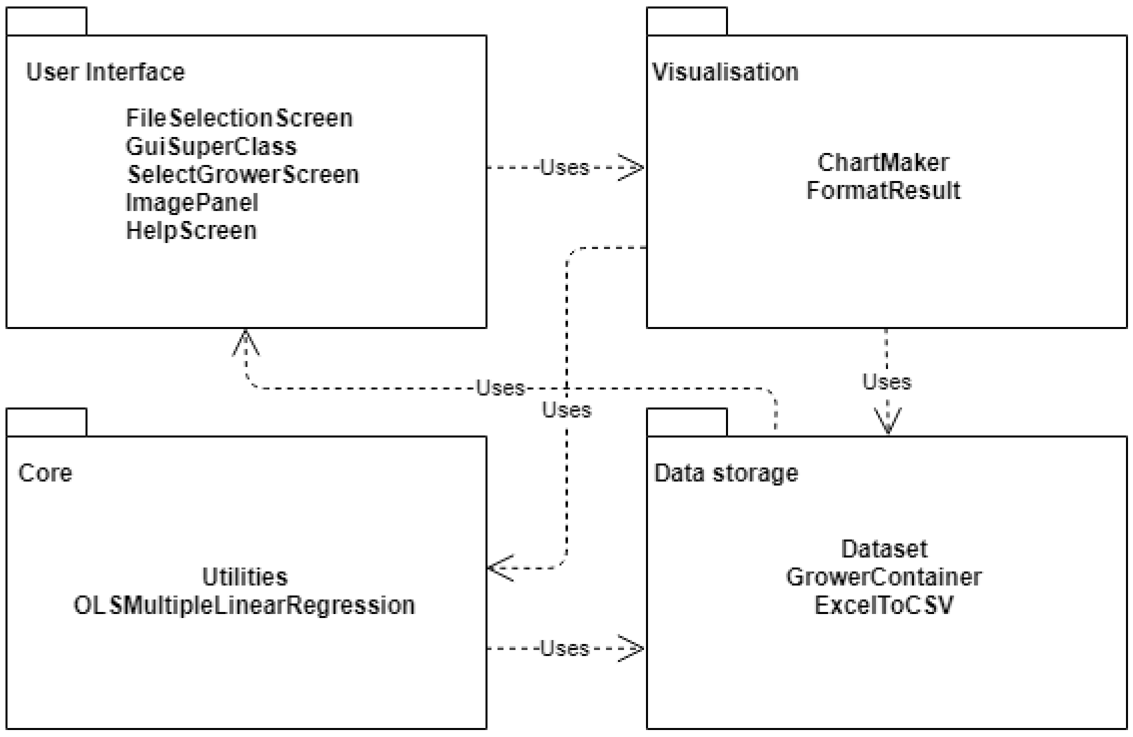
Figure 1: Use case diagram



4.2.2 Package diagram

The program consists of different 'blocks' of code that interact with each other. These interacting blocks are called packages. The package diagram shows the different packages in the program, and shows how they depend on each other. A package diagram in UML is used for the decomposition of the application and it might be different than the implementation-levels packages. However, software developers mostly prefer following the same decomposition during the implementation.

Figure 2: Package diagram



The program contains four main packages: Core, Data storage, User Interface and Visualisation. The User Interface package contains everything the user actually sees. The Core package contains the multiple linear regression, and different utilities used throughout the program, like being able to pad a sentence to a certain length, or finding the extension of a file. The Data Storage contains all information that needs to persist throughout the use of the program like the dataset and the name of the selected grower. The visualisation package prepares the data from the core package and makes it easily interpretable for the user. The user interface depends on the visualisation package to deliver what it needs to show to the user. The visualisation package in turn needs to know which grower has been selected from the data storage, and also needs the results of the analysis from the core package. The core package only needs the data stored in the Data storage. The Data storage receives the data to be stored from the user interface. This is how the different packages of the program work together.

4.2.3 Class diagram

The different names in the packages of figure two are called 'classes'. Classes are blocks of code that perform a specific action. This division into blocks keeps code easily maintainable and interpretable. The package diagram shows which classes belong to the same package, but it doesn't show which classes rely on each other. The class diagram shows which classes are related to each other, and in which way. Figures 3 and 4 show the relations between classes within the packages.

Figure 3: Class diagram of the user interface package

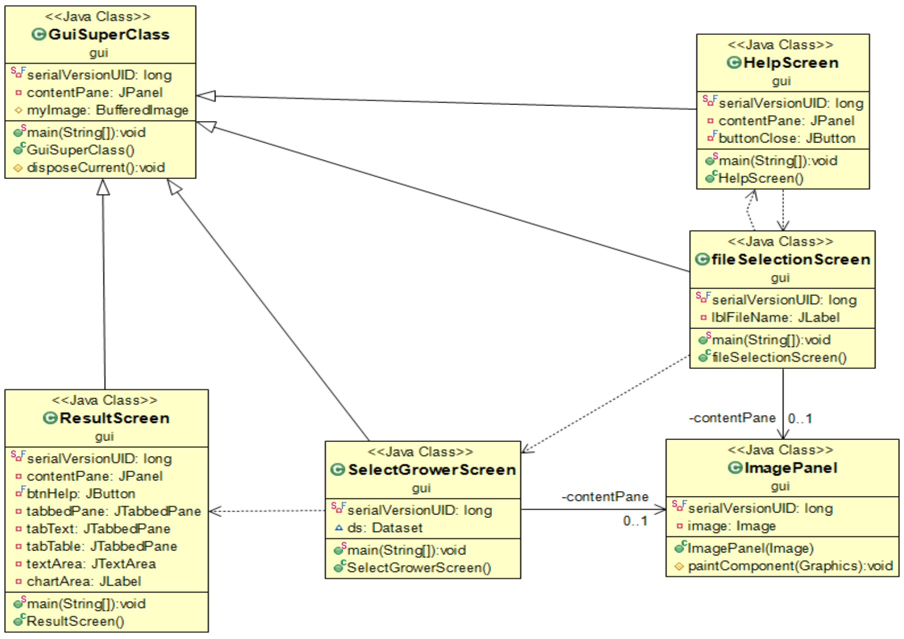


Figure 4: Class diagram of the main package

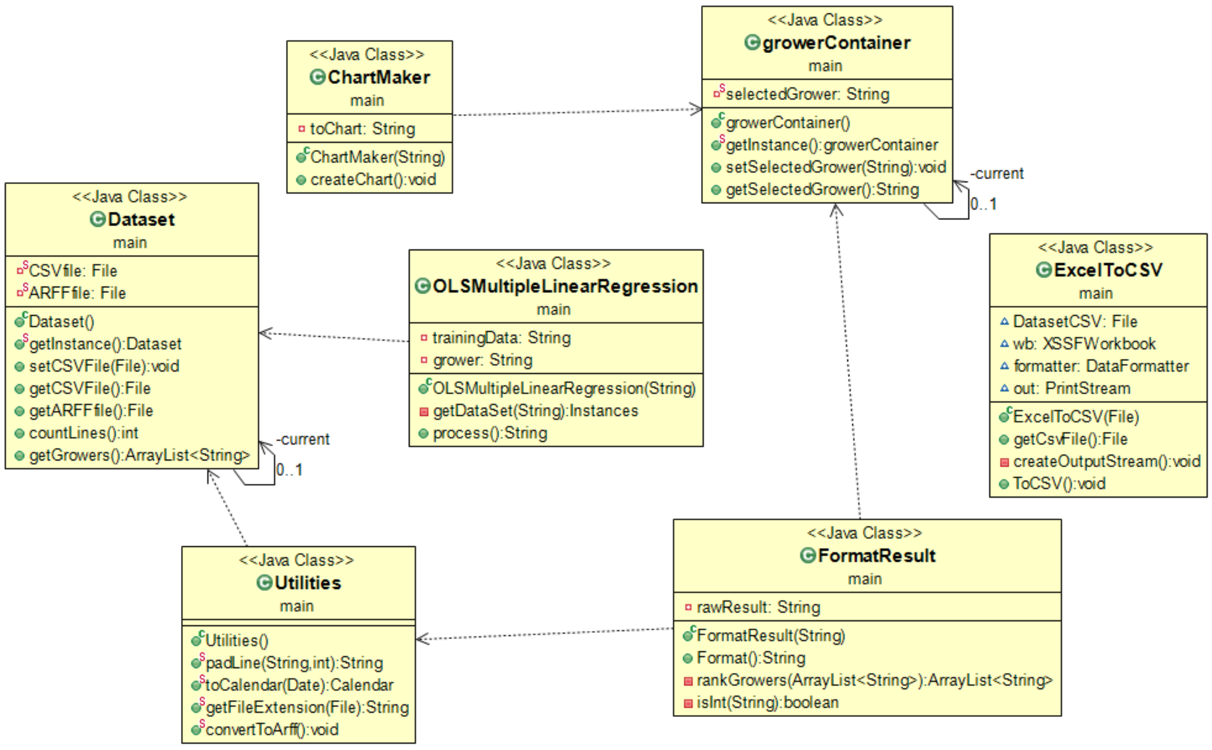


Figure 3 shows the relation between the different classes in the user interface package. These classes can be divided into screens, a super class, and a regular class. All screens (HelpScreen, FileSelectionScreen, SelectGrowerScreen, ResultScreen) are dependent on the superclass GuiSuperclass. This superclass contains the background image that can be used by all screens that inherit it, and it contains the method that disposes a screen when the program advances to the next screen.

The ImagePanel class allows the screens that need a background to display it. The ‘regular’ JFrame does not have a built-in method that allows setting a background. The ImagePanel is descended from the class JComponent, and overloads the function that ‘paints’ the background grey with a function that allows an image to be set as background instead.

Figure 4 shows significantly less relations within the package than Figure 3. This is explained by the fact that most classes in the main package are related to screens in the user interface classes. It can also be seen that growerContainer and Dataset have a relation to themselves. These classes make use of the ‘singleton’ design pattern. This pattern allows every instance of that particular class to store the same information. Because there is only one dataset, and only one selected grower, the singleton pattern was applicable to these classes.

Figure 5: Class diagram of the entire program

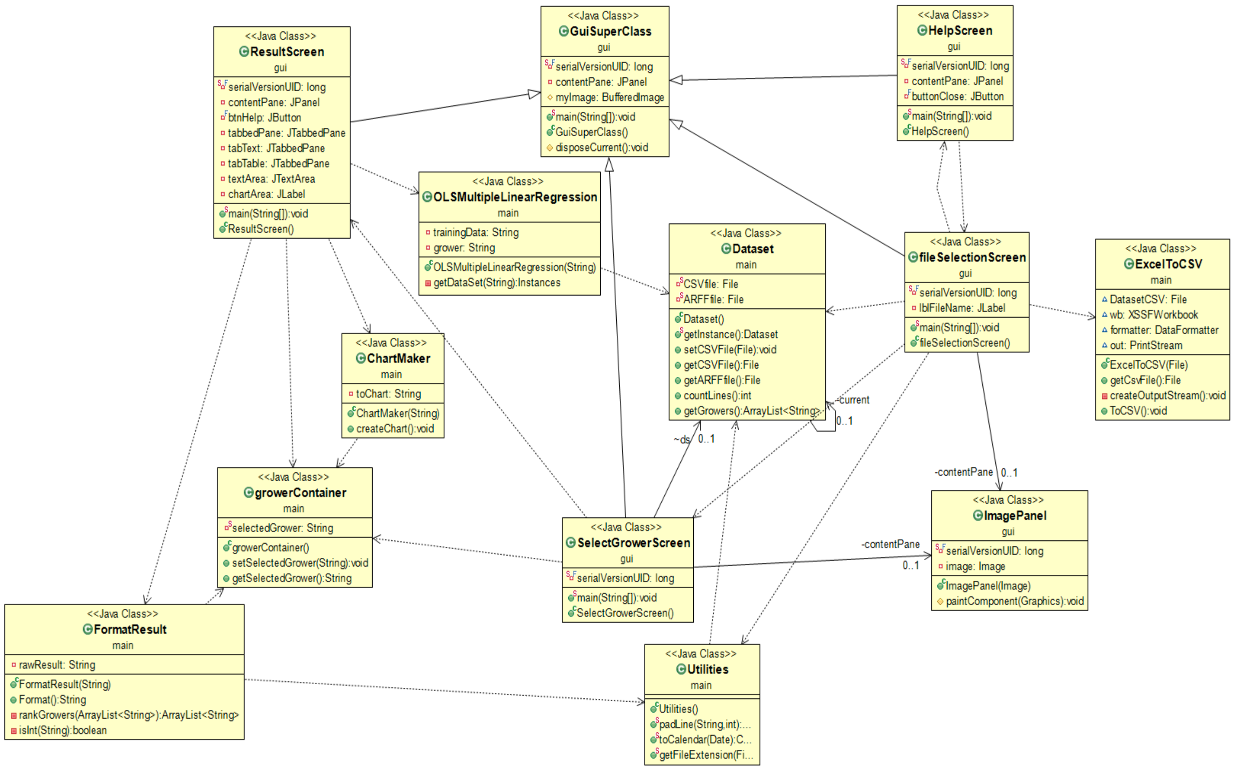


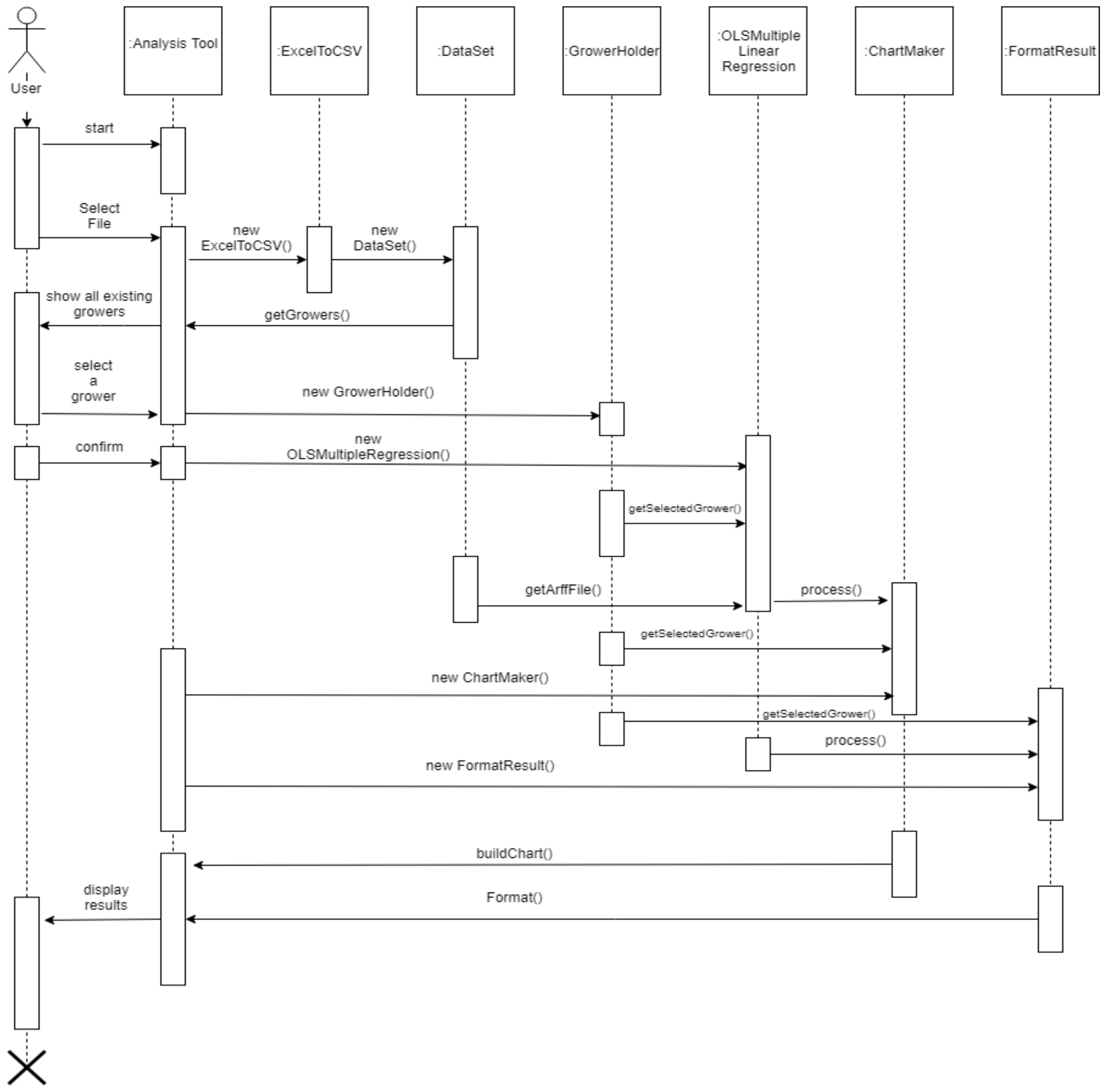
Figure 5 shows all the classes in the entire program and their relationships. It combines Figures 3 and 4, and visualises which parts depend on each other.

4.2.4 Sequence diagram

Whereas the class diagram depicts the static relations between different parts of the tool, the sequence diagram shows the dynamic interactions that take place while the program is being executed. The sequence diagram is used to elaborate upon one specific use case of a program. In this case, a complex use case is elaborated for further explanation. Figure 6 contains the sequence diagram in the case of using the program to analyse an excel file. The user enters the file, selects which grower to set the coefficients relative to, and gets the results.

While this sequence of actions might seem straightforward from the user's perspective, many interactions occur 'under the hood' of the program. The figure 6 walks through the process the program takes step by step.

Figure 6: Sequence diagram



4.3 Implementation

This chapter discusses different relevant technical aspects of the implementation of the design diagrams mentioned above. First the general specifications of the program are discussed, thereafter the classes and techniques are introduced.

4.3.1 Program specifications

The program was written in Java. Java is an object-oriented programming language that enforces the use of classes, packages and other structural elements. Because this structure is enforced, it inherently supports organisation in larger coding projects. Java can be written in multiple ways, one of those ways is coding it by using a text editor and compiling it with a command line tool, and the another option is writing the program in an IDE (Integrated Development Environment). The advantage of using an IDE is that steps that have to be taken when using the command line tools are automated and happen 'under the hood'. For this reason, the Eclipse Neon IDE was chosen to implement the project.

As mentioned in the requirements, the programme makes use of so called 'libraries'. Libraries are pieces of code that are ready to be used in any piece of code. These libraries are sometimes already included in the IDE, or they can be downloaded from several open source repositories. The program uses multiple external libraries that are essential to its functionality. The imported libraries and their roles in the program are discussed in the following sub sections.

WEKA

WEKA is an open-source, free to use collection of machine learning algorithms. It allows for use as a standalone program or as a software library. WEKA was used in multiple places in the program. Most importantly, its class for performing an ordinary least squares multiple linear regression was used. This is the 'engine' of the analysis in the program. Next to the analysis, the built-in converter from CSV to ARFF converter was used to make the data ready for analysis. (Frank, Hall, Witten, 2016)

Apache POI

The Apache POI was used to be able to read and process the Excel file entered by the user. It is used in the ExcelToCSV class, which consequently stores it into the dataset class. This functionality depends in the apache.commons.jar, which contains some basic operations used throughout the library (The Java API for Microsoft Documents, n.d.).

JFreechart

JFreechart was used to create the bar chart displayed in the ResultScreen. It depends on the library JCommon (JFreeChart, n.d.).

ObjectAid

The objectAID library automates the production of class diagrams, and is responsible for the creation of Figures 3, 4 and 5. This library can be considered as a reverse engineering tool which produces the class files using implementation-level artifacts (ObjectAID ,n.d.).

Windowbuilder

The Swing library was used in combination with the Windowbuilder library. This allowed easy building of GUI (graphical user interface) screens. Both of them are very well-known user interface libraries in Java, which are widely applied by Java developers. (Wren, n.d.)

4.3.2 Classes and techniques

This chapter describes all the classes used, and includes any noteworthy techniques.

Gui package

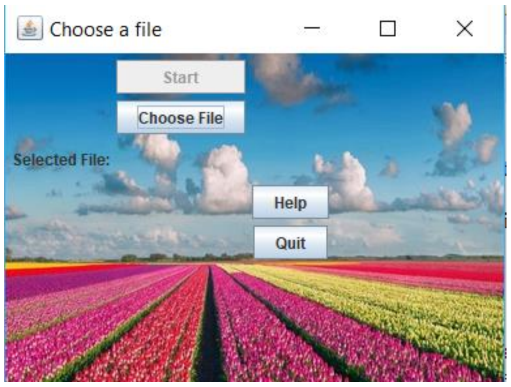
The Gui package mainly contains the visual aspects of the tool. For this reason, this sub section also contains screenshots for clarification purposes.

FileSelectionScreen:

This class produces the initial screen the user sees when the program is started. Figure 7 shows the layout of the screen. The 'choose file' button creates a pop-up that allows searching for a file in the operating system. The selected file label shows the name of the file when it is selected.

To support user friendliness, the 'selected file' label displays the name of a file with a correct format, and displays the message "please select an excel or CSV file", in addition the start button only becomes clickable when the right file format is selected. The help button creates the help screen, which provides extra information.

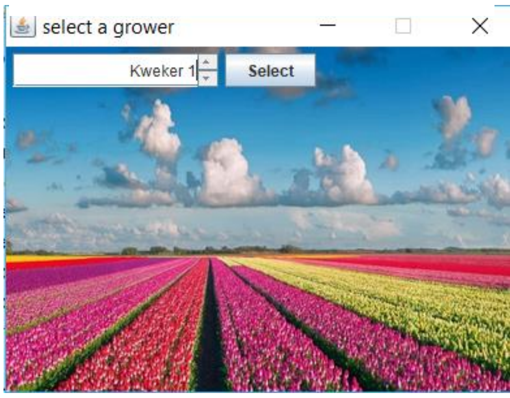
Figure 7: start screen layout



GuiSuperclass:

As its name indicates, the GuiSuperclass is the super class for all GUI screens. It provides shared variables and shared methods. It mainly provides one variable that all gui classes inherit the background image from.

Figure 8: grower selection screen



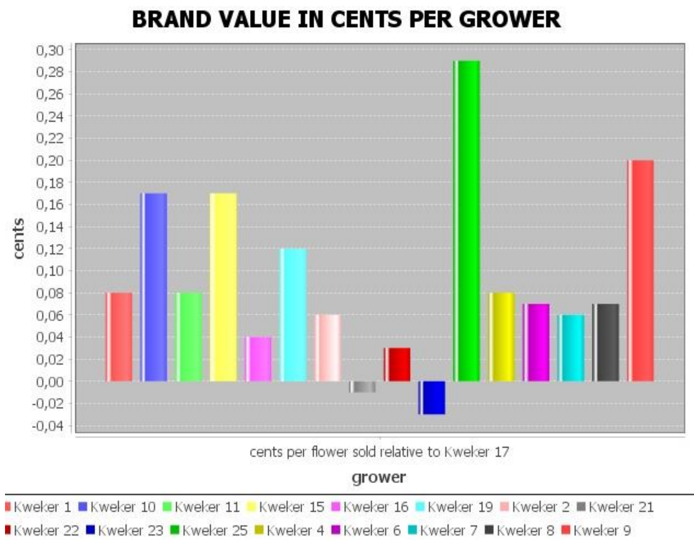
SelectGrowerScreen:

The grower selection screen facilitates the choice of which grower to use as contract for the other coefficients. It receives a list of growers from the dataset and puts them in a component. It passes on the selected grower to the growerContainer class.

Imagepanel:

The ImagePanel is used by the grower selection screen and the file selection screen, as they are in need of a background image. The standard JComponent class does not allow for the selection of a background image. The ImagePanel extends the JComponent class, but overloads its paintComponent() method. This allows the component to be created with a picture as a background while still retaining all regular functionalities as a JComponent.

Figure 9: bar chart visualising different growers



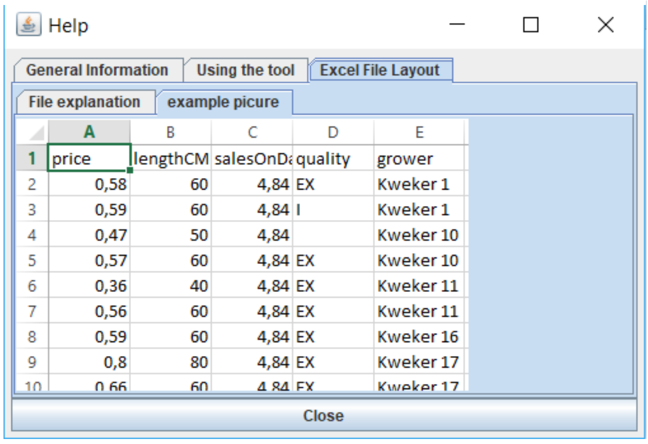
ResultScreen:

The results screen shows two different visualisations of the result. One simply contains the results in text, and ranks the growers in terms of their performance. The other provides a bar chart that better visualises the differences between growers. This screen makes use of tabbed components. The 'help' button provides help with interpreting the result.

HelpScreen:

The help screen can be accessed from the starting screen. It offers help with multiple issues the user can have while using the program. It provides general information, a guide on using the tool, and an explanation on how to format the input excel file along with a picture of how it should look. The close button returns the user to the start screen.

Figure 10: help screen



	A	B	C	D	E
1	price	lengthCM	salesOnD:	quality	grower
2	0,58	60	4,84	EX	Kweker 1
3	0,59	60	4,84	I	Kweker 1
4	0,47	50	4,84		Kweker 10
5	0,57	60	4,84	EX	Kweker 10
6	0,36	40	4,84	EX	Kweker 11
7	0,56	60	4,84	EX	Kweker 11
8	0,59	60	4,84	EX	Kweker 16
9	0,8	80	4,84	EX	Kweker 17
10	0,66	60	4,84	EX	Kweker 17

Main

The main package contains the part of the program that is invisible to the user. Therefore, this sub section makes use of CRC (Class responsibility) cards. A CRC card is a comprehensive overview of the responsibilities and dependencies of a class. The left side denotes the responsibilities of the class, and the right side shows what other classes it relies on in order to function. Any noteworthy techniques are again added in the description.

ChartMaker:
This class makes use of the JFreeChart library

Chartmaker	
Responsibilities	Collaborators
Creates bar chart from regression output Saves bar chart as image	GrowerContainer OLSMultipleLinearRegression

Dataset:
This class is built with the singleton pattern, meaning that every time this class is called, it returns the same instance of the class

Dataset	
Responsibilities	Collaborators
Contains CSV File Contains ARFF File Isolates all unique growers in the dataset	fileSelectionScreen

ExcelToCSV:

The ExcelToCSV class takes the selected file from the file selection screen and converts it to a csv file, which can in turn be converted to an arff file. This class makes use of the Apache POI library.

ExcelToCSV	
Responsibilities	Collaborators
Converts Excel to CSV file	fileSelectionScreen

FormatResult:

FormatResult	
Responsibilities	Collaborators
Outputs the regression result into a readable format	Resultscreen OLSMultipleLinearRegression

GrowerContainer:
This class makes use of the singleton pattern.

GrowerContainer	
Responsibilities	Collaborators
Contains the grower that was selected in the SelectGrowerScreen	SelectGrowerScreen

OLSMultipleLinearRegression:

This class makes use of the WEKA library.

OLSMultipleLinearRegression	
Responsibilities	Collaborators
Performs a OLS multiple linear regression on the dataset	GrowerContainer ResultScreen DataSet

Utilities:

OLSMultipleLinearRegression	
Responsibilities	Collaborators
Pads strings to a desired length Detects file path extensions Converts csv to arff files Swaps the deprecated type Date to Calendar type	

4.4 Testing

This chapter contains the tests that check if the program meets certain set of requirements. First, the functional and non-functional requirements are evaluated. Thereafter, acceptance tests are discussed.

4.4.1 Software requirements

Table 1 : Functional requirements

Requirement	Description	Completed	Explanation if needed
R1	The program shall analyse auction data, and perform a statistical analysis on them.	Yes	
R2	The program shall take an Excel document as input format for its data.	Yes	The program also takes CSV formatted files
R3	The program shall deliver its output in a separate window.	Yes	The program outputs bar chart, and the contribution of brand per flower sold in cents in a separate screen
R4	The program shall not rely on third-party programs	Yes	As mentioned before, the program makes an exception for Excel
R5	The program shall not ask the user for anything that requires knowledge on statistics.	Yes	
R6	The program shall perform a statistical analysis of the data automatically	Yes	

Table 2 : non-functional requirements

Requirement	Description	Completed	Explanation if needed
R7	The program shall be written in the programming language Java	Yes	
R8	The program shall make use of the WEKA library for Java to run the analysis	Yes	
R9	The program shall be user friendly	Yes	The program offers help for using it, and has minimised room for error during use.
R10	The program shall apply best practices and design patterns whenever possible.	Yes	

As can be seen from tables 1 and 2, all functional and non-functional requirements have been met.

4.2.2 Acceptance tests

To test if the program functions properly, it is important to know if it passes certain acceptance tests. Acceptance tests check if the program has certain functionalities or if these functionalities have been implemented correctly by the programmer. The next table contains the acceptance tests for the program and shows if they have been met.

Table 3 : acceptance tests

Acceptance test	**Steps**	**Passed**	**Additional information**
The program should have a background image on the screens so it is visually more appealing	Start program, look at background	Yes	Only in places where the background wasn't completely blocked by other components like tabs etc
When the help button is clicked in the start screen, a new screen should pop up that provides help on how to use the program.	Start program, click 'help' button	Yes	The help screen contains help on how to use the program, what it does, and on which file should be entered.
The grower names in the document should be provided to the user to prevent mistakes	Start program, select excel file, click 'start' button, select a grower	Yes	The grower selection screen only provides the growers that are in the dataset
The program shouldn't allow the user to progress when the wrong file is selected	Start program, select any non-excel or csv file, try to click start button	Yes	Start button only becomes clickable when the right file is selected
The program shouldn't allow the user to progress when no file is selected	Start program, try to click start button	Yes	Start button only becomes clickable when the right file is selected
The program should provide a bar chart to analyse results	Start program, select file, click start button, select a grower, click select button, look at chart	Yes	

These tables show that all acceptance tests have passed, and the described functions are therefore implemented correctly.

5. Results

The regression was carried out with stepwise entry of different blocks of variables. Four blocks were entered, each with one of the variables. All dummy variables that originated from one variable were entered in one block. This stepwise entry allowed to check the F-statistic for each variable. the F-value increased significantly for each block entered ($p < 0.000$). A significant change in the F-value shows that the entered variables cause a significant increase of the predictive power of the model.

Table 4 shows the coefficients of all entered variables. In this case, grower 23 was chosen as a reference point. Quality rating 'B' has been chosen as a reference point for the quality ratings. It also shows the coefficients of all variables in the model. It is important to keep in mind that all grower coefficients are relative to grower 23, and all quality ratings are relative to quality rating b.

Table 4: Model summary without grower 23 and quality B

Variable	**Coefficient**	**Standard error**
Constant	-,782	,035
Total sales on date per 10.000	,038	,001
Length in cm	,014	,000
Grower 1	,057	,025
Grower 4	,053	,021
Grower 6	,038	,023
Grower 7	,040	,030
Grower 8	,047	,023
Grower 9	,197	,019
Grower 10	,137	,027
Grower 11	,045	,032
Grower 15	,146	,032
Grower 16	,008	,026
Grower 17	-,027	,027
Grower 19	,088	,026
Grower 21	-,038	,029
Grower 25	,296	,027
Quality EX	,151	,017
Quality I	,125	,022
Quality S	,270	,022
Quality unknown	,103	,020

Statistic	**Value**
N	4266
R Square	0.485
F	199,914 + df 20
F – significance	0.00

6. Conclusion

As the dependent variable in the model is measured in cents, all independent coefficients are expressed in cents as well. For instance, each centimetre of length added to the flower, adds 3.8 cents on average to its price. This same logic can be applied to the growers. The difference between grower 23 and grower 25 is 29.6 cents per flower. As the amount of flowers sold can reach into the hundreds of thousands per year, this difference per flower can amount to significant increases in revenue over time. The combination of the significance of the f-statistic, and the differences found between different growers shows that there is indeed a significant, and sizable impact of brand. This impact is specific to the product that has been analysed, and doesn't necessarily say anything about other products associated with the same brand.

This shows that it is possible to isolate the brand equity of a grower based on the differences of price to a certain extent. The analysis results show that the grower has a significant influence on price, and shows the relative difference between the different competitors. This confirms that the brand equity of a grower can be measured by use of a multiple linear regression analysis.

7. Discussion

This research has shown that the brand equity of a brand for a specific product can be measured by use of a multiple linear regression. It has also produced an analysis tool that presents the results of this analysis in an easily interpretable way. The results of the analysis tool show the performance of the grower when compared to his competitors. It is free to use, and can also be used for different species of flowers as long as the variables used in this research can be entered into the program. This analysis tool can be used by the auction as well as the grower to monitor market performance. This can function as an incentive for growers on the lower end of the performance ranking to improve their quality, or be a confirmation of performance for growers on the higher end.

Furthermore, this research has confirmed that there is indeed an effect of grower brand on price. This knowledge in itself could incentivise the grower to take more care about his brand image. A relative difference of 29 cents per flower can accumulate to a substantial amount of revenue gained.

This research has applied pre-existing theories on a real-world situation. It has taken the definitions of brand, brand equity and perceived quality, and applied them to the situation of the Dutch flower auctions. The definition of brand was easily applied to the name of the grower, and brand equity and perceived quality's application on the product were supported by the literature. The literature suggested that there should be a difference in price based on the grower. The outcome of the analysis confirms this hypothesis. This research has confirmed the applicability of the used theories from literature on a specific real-world case.

However, a few things should be taken into account when interpreting the results of this research. The analysis has confirmed that there is an effect of grower on price, but it doesn't explain why. Literature suggests that quality is one of the main contributors, but the outcome of the analysis cannot confirm this. Furthermore, the outcome now only shows the brand equity associated to a specific product, it does not show the performance of the brand over all markets.

The current output that is delivered by the tool shows the performance of the growers relevant to each other. It doesn't show the actual amount of cents gained by the grower used as baseline. Therefore, the actual amount of money received by differences in grower cannot be ascertained.

The multiple regression does not take changes over time into account. A grower could score highest in one period, score lowest in the next and be represented as an 'average' grower overall. In order to truly see the development of brand equity over time, changes over time should be taken into account.

Further research should be focused removing the limitations of the current analysis. It will be very beneficial to the grower not only to know that he is receiving a different price for his product than his competitors, but also why this is the case. Clarifying the reasons why a competitor is performing differently on his brand equity presents a clearer picture of where the grower needs to improve. Currently, one of the reasons given for difference in brand equity is product quality. However, there are more contributors, and identifying these will be very beneficial to the applicability of the this research.

Furthermore, the analysis tool should be further improved upon. Next iterations should provide a clear picture of the development of brand equity over time. This could provide the grower with an idea with what makes his brand equity shrink or grow, as well as alert him in time when his brand equity is declining.

This research has provided a basis on which can be expanded and improved. It has proven the existence of the effect of brand on price, and created an analysis tool that can isolate this effect. However, further research and improvement is needed if the grower is to get the most value out of the tool.

Literature

Aaker, D. (1997). *Managing brand equity*. New York, NY: Free Press [u.a.]

Eibe Frank, Mark A. Hall, and Ian H. Witten (2016). *The WEKA Workbench. Online Appendix for "Data Mining: Practical Machine Learning Tools and Techniques",* Morgan Kaufmann, Fourth Edition, 2016.

Field, A. (2009). *Discovering statistics using SPSS*. 3rd ed. London: SAGE.

Fishbein, M. and Ajzen, I. (1975). *Belief, attitude, intention and behavior*. Reading, Mass.: Addison-Wesley.

JFreeChart (n.d.). Retrieved June 25, 2018, from http://www.jfree.org/jfreechart/

Kevin Lane Keller (1993) Conceptualizing, Measuring, and Managing Customer-Based Brand Equity, Journal of Marketing, Vol. 57, No. 1 (Jan., 1993), pp. 1-22

ObjectAID (n.d.). Retrieved June 25, 2018, from http://www.objectaid.com/installation

Peter J. Batt (2001) Strategic Lessons to Emerge from an Analysis of Selected Flower Export Nations, Journal of International Food & Agribusiness Marketing, 11:3, 41-54, DOI:10.1300/J047v11n03_03

Royal FloraHolland (2016) Annual Report 2016, retrieved from: https://publish.folders.eu/en/fixed/1057037?token=c7e5b912bfa008cb7957f8331c1477b8&pageMode=single

Royal FloraHolland. (2017, April 12). Retrieved March 09, 2018, from https://www.youtube.com/watch?v=FUJoRMD0B-Y&feature=youtu.be

Seber, G. (1977). *Linear regression analysis*. New York: John Wiley & Sons.

Software sources:

Steenkamp, J. (1990). Conceptual model of the quality perception process. *Journal of Business Research*, 21(4), pp.309-333.

The Java API for Microsoft Documents. (n.d.). Retrieved June 25, 2018, from https://poi.apache.org/

Wren, J. (n.d.). WindowBuilder | The Eclipse Foundation. Retrieved June 25, 2018, from https://www.eclipse.org/windowbuilder/