

UNIVERSITY OF TROMSØ

Initial system's implementation for data
Analysis and values forecast about
Aquaculture in Norway

by

Andrea Spreafico

A thesis submitted in partial fulfillment for the degree of Computer Science
Computer Science

in the

Faculty of Computer Science
Department of Computer Science

May 2017

Declaration of Authorship

I, Andrea Spreafico, declare that this thesis titled, ‘Initial system’s implementation for data Analysis and values forecast about Aquaculture in Norway ’ and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed:

Date:

*"We did it, we bashed them, wee Potter's the one,
and Voldy's gone moldy, so now let's have fun!"*

- Peeves

UNIVERSITY OF TROMSØ

Abstract

Faculty of Computer Science
Department of Computer Science

Computer Science

by [Andrea Spreafico](#)

Moonstone (also known as the wishing stone[1]) is found in a variety of colors. Its supposed magical effects include helping a person gain emotional balance. Since Harry spent much of book five emotionally unbalanced, it is perhaps fitting that he was forced to write an essay on the stone's use in Potions-making. It is a gemstone of medium value. Moonstones are a milky colour and shine very brightly, almost as though they are a source of their own light. They are a useful potion ingredient; powdered moonstones are used as an ingredient for the Draught of Peace and in several Love Potions. Powdered Moonstone is also an ingredient in in Potion No. 86 which is likely an experimental potion. Moonstones were also known to be among the gems set into Muriel's tiara.

Acknowledgements

I would like to express my very great appreciation to Susan Bones for the . . .

I would also like to offer my special thanks to Cedric Diggory for. . .

My special thanks are extended to the staff of the Matron for. . .

My special thanks goes to Pomona Sprout for taking on this thesis work.

I am particularly grateful for the support and good times given by my friends, for. . .

To my family, for. . . , I am particularly grateful.

Advice given by Helga Hufflepuff has been a great help in. . .

To my beloved Ernie Macmillan for all the. . .

Contents

Declaration of Authorship	i
Abstract	iii
Acknowledgements	iv
List of Figures	viii
List of Tables	ix
Abbreviations	x
Physical Constants	xi
Symbols	xii
1 Introduction	1
1.1 Aim of the study	1
1.2 Initial Goals	2
2 Background Theory	4
2.1 Data science	5
2.2 Aquaculture in Norway	6
2.3 Machine learning	7
2.3.1 Time Series analysis and predictions	7
2.4 Python	8
3 Development Method	9
3.1 Development Flow	9
3.2 Github Usage	11
4 Implementation	12
4.0.1 Implemented Systems repositories	12

I	Data collection and validation	14
4.1	Data collection	15
4.1.1	Data sources	15
4.1.2	Data description and validation	16
4.2	Increase accessibility and availability of data	17
II	Data Analysis and Displaying	18
4.3	Analysis of the data	19
4.3.1	Single Input Analyzer	20
4.3.1.1	SIA: Requirements for reusability	20
4.3.1.2	SIA section I: Total graphic for all the years	21
4.3.1.3	SIA section II: Single graphics for each year	22
4.3.1.4	SIA section III: Correlation matrix between years	23
4.3.1.5	SIA section IV: Correlation matrix between months	24
4.3.1.6	SIA section V: Single overview	25
4.3.2	Multiple Inputs Analyzer	26
4.3.2.1	MIA: Requirements for reusability	26
4.3.2.2	MIA: Implementation	26
4.4	Extract information from data	28
III	Data Prediction	29
4.5	Prediction of values about the data	30
4.5.1	Evaluating System	31
4.5.2	Training System	32
4.5.3	Future Prediction System	34
4.6	Requirements for reusability	36
IV	Future Works	37
4.7	Visualization of the data	38
4.8	Prediction system as a service	38
5	Results	39
6	Discussion	40
7	Conclusion	41
8	Bibliography	42
A	SIA Implementation code	43
A.1	SIA: Imported libraries	43
A.2	SIA section I: Total graphic for all the years	44
A.3	SIA section II: Single graphics for each year	45
A.4	SIA section III: Correlation matrix between years	46

A.5	SIA section IV: Correlation matrix between months	48
A.6	SIA section V: Single overview	49
B	MIA Implementation code	51
B.1	MIA: Imported libraries	51
B.2	MIA: Implementation	51
C	Prediction System Implementation code	53
C.1	Evaluating System	53
C.2	Training System	53
C.3	Future Prediction System	53

List of Figures

2.1	Data science concept	5
2.2	Data science process	5
3.1	Plan flow chart	10
4.1	Total graphic about current input with total data from 2005 to 2016. . . .	21
4.2	Graphics for each single year of the input data from 2005 to 2016	22
4.3	Correlation matrix between different months of the same input	23
4.4	Correlation matrix between different years of the same input	24
4.5	Example of "Single Input Overview Image"	25
4.6	Correlation matrix between different inputs with data from 2005 to 2016. .	26
4.7	Normalized angular coefficients of each input's trendline.	27
4.8	Graphic that displays different MAPE values for each ARIMA order. . . .	31
4.9	Graphic that displays the predicted values from a particular ARIMA machine configuration and the historic real values.	33
4.10	Graphic that display historic, future and predicted values of a input. . . .	35

List of Tables

Abbreviations

LAH List **A**bbreviations **H**ere

OWL Ordinary **W**izarding **L**evel

Physical Constants

$$\text{Speed of Light } c = 2.997\,924\,58 \times 10^8 \text{ ms}^{-\text{S}} \text{ (exact)}$$

Symbols

a	distance	m
P	power	W (Js^{-1})
ω	angular frequency	rads^{-1}

For/Dedicated to/To my...

Chapter 1

Introduction

1.1 Aim of the study

Every single day in the world is produced a huge amount of data: some of this data, if they are analyzed and interpreted in the right way, could provide useful informations. If we watch for example at the Aquaculture business in Norway is produced a big amount of data about every single locality or about national statistics, but most of the time this data are not analyzed and difficult to understand.

The main purposes of this thesis are basically to test and show:

- Data potential in Aquaculture business in Norway through a system for analyzing and displaying data, in order to help the companies related with Aquaculture to improve their operations thanks to the analysis results.
- Python potential in data science, in order to show the people how it works and what you could do using it.

For achieve the goals reported above, this thesis will provide:

- Implementation and description of a procedure that can be used for make a Python system able to do an initial analysis of big datasets and also to display the obtained results.
- Implementation and description of a procedure that can be used for make a system implemented in Python able to predict future's values using a regression model.
- List of possible implementation for further uses of machine learning algorithms and regression model.

1.2 Initial Goals

1) Collect as much data about aquaculture in Norway as possible.

- Is possible to obtain data about aquaculture general statistics in Norway?
- Is possible to obtain data about aquaculture of single locations in Norway?

2) Increase accessibility and availability of the data.

- Is that possible to create a unique dataset that contains all the data previous collected?
- Is the total dataset easier to access and read the new dataset structure compared with original single sources?

3) Analyze and display the data.

- Is that possible to provide a general analysis and displaying of data about Norwegian aquaculture reported in the dataset?
- Is that possible to use Python for analyzing and displaying the data?
 - Is Python a good programming language for data analysis and displaying?
 - Does Python give the possibility to create analysis systems in easy way?
- Are the obtained graphics allowing to comprehend information quickly?
- Are the obtained graphics allowing to identify relationships and patterns in a easy way?
- Are the obtained graphics allowing to identify historic and future trend line?
- Are the obtained informations (such as correlation coeff, trend line equations,..) reported in documents ready to be reused in the future?

4) Extract information from the data.

- Is the aquaculture business in Norway growing?
- More in the specific, which parameters are increasing and how fast?
- Is that possible to have a trend line about the data?
- Is that possible to find some correlations between the data in input and display them?
- Are that data influenced from external parameters? Such as weather, news, people ideas..

5) Prediction of values about the data.

- Is that possible to predict some of the data that we own?
 - Would it be useful to have the possibility of forecasting some future data?
 - Which kind of data might be the most useful to know for people into the Aquaculture field?
- Test and describe time series predictions utility in Python.
 - Is Python a good way for time series prediction systems implementation?
 - Does it give the possibility to have accurate results?
 - Is that good for let the people get some experience with machine learning field?

6) Recommendations to future work and extra ideas.

- Is it possible to let the informations collected before be available like a service?
 - Is that possible to provide the analysis and prediction systems like a service?
 - Is that possible to let the data displaying dynamic instead of static?

Chapter 2

Background Theory

The background theory required for implement this thesis work could be basically divided into 4 main areas:

- Data science
- Aquaculture in Norway
- Machine Learning
- Python

Could be very useful to give some basic definitions and explanations about the topics written just above and then try to get some more specific informations during the course of this thesis.

2.1 Data science

It's really important to have a general idea about what "Data Science" means since this thesis procedure is strongly based on the classic Data Science Process.

We can define Data Science like a "concept to unify statistics, data analysis and their related methods" in order to "understand and analyze actual phenomena" with data.

It includes theories drawn from many field within the broad areas of mathematics, statistics, information science and computer science.

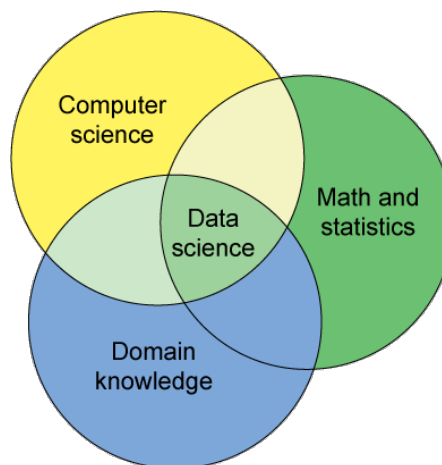


FIGURE 2.1: Data science concept

In the computer science area are particular important the subdomains of machine learning, classification, cluster analysis, data mining, databases, and visualization.

The follow image represents the "Blitzstein and Pfister's framework" and provides a clear overview of the topic.

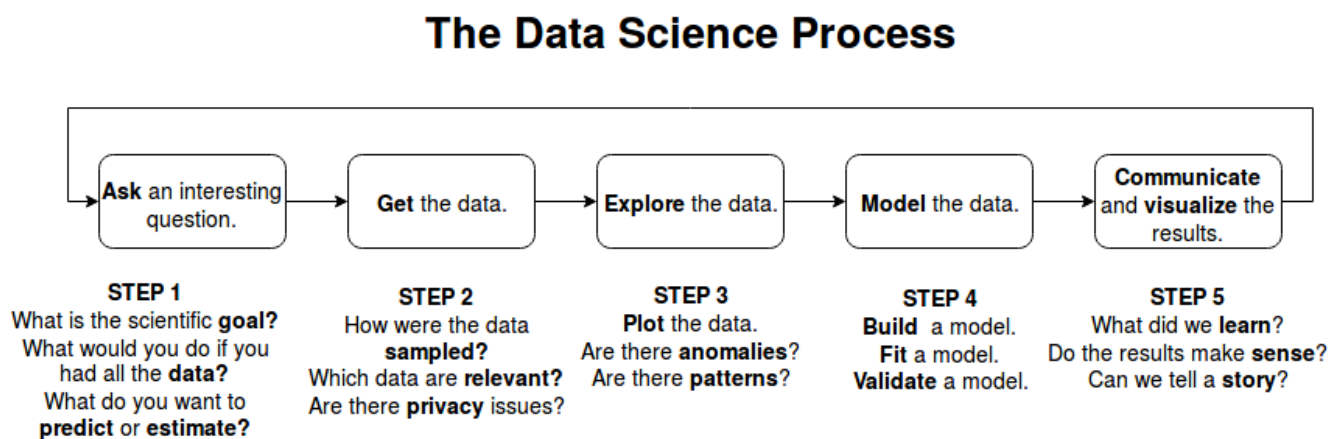


FIGURE 2.2: Data science process

2.2 Aquaculture in Norway

Aquaculture, also known as aquafarming, is the farming of fish, crustaceans, molluscs, aquatic plants, algae, and other aquatic organisms.

Aquaculture would be the future of fish: In 2030, according to the World Bank, aquaculture will supply:

- 93.6 Million tonnes of fish per year
- 25 percent less wild fish will be available
- 62 percent of the fish we eat will come from farms

2.3 Machine learning

This subfield of computer science gives "computers the ability to learn without being explicitly programmed".

Evolved from the study of pattern recognition and computational learning theory in artificial intelligence, machine learning explores the study and construction of algorithms that can learn from and make predictions on data.

There are several machine learning algorithm, each one of them is used for a different purpose. The following picture gives a general idea about which categories of algorithms are used and some specific types.

2.3.1 Time Series analysis and predictions

Time Series forecasting is an important area of machine learning, but that is often neglected.

Is that important mainly because there are so many prediction problems that involve a time component, and these problems are neglected because it is this time component that makes time series problems more difficult to handle.

" A time series is a sequence of observations taken sequentially in time. " Quoted — Page 1, Time Series Analysis: Forecasting and Control.

Classic example of a time series dataset:

Time #1, observation

Time #2, observation

Time #3, observation

There are different goals depending on whether we are interested in understanding a dataset or making predictions.

Understanding a dataset is called time series analysis and it can help to make better prediction, but sometimes it's not required and can result in a large of technical investment in time and expertise.

Making predictions could be called time series forecasting and it involves taking models fit on historical data and using them to predict future observations.

Information Source: <http://machinelearningmastery.com/time-series-forecasting/>

2.4 Python

Chapter 3

Development Method

3.1 Development Flow

1st Phase: Data collection and validation

During this phase the most important thing is to gather as much as possible data, but they must be as much as possible reliable and useful since they are going to be indispensable for the next phases and in particular for the final results and conclusions. The data's reliability mainly depend by the kind of sources where you're able to mine. Then you should customize the unstructured data that you collected.

This data's customizing has the main purposes of:

- Let the data structure be a summarize of all the data inputs previous collected.
- Let the new data structure be easier to access and read.
- Follow some kind of setting and standard needed in the system that will be implemented.

2nd Phase: Data Analysis and Displaying

During this phase the first thing that you're going to do is to decide some kind of analysis results that you would like to have.

Once you decided which kind of results you might reach, you will start with the analysis system implementation and meanwhile saving evidences of it.

Once the general analysis of the data is finished, and evidences have been collected, it's time to analyze it and try to extract information about it.

3rd Phase: Data Prediction

During this phase the main purpose is to predict some kind of useful data about the current dataset. To reach this goal, is first of all indispensable to choose a prediction system to implement.

Once the prediction system has been implemented, it's time to apply it on the current data and try to get as much evidences as possible.

4th Phase: Future Work ideas

The last but not least phase is to watch at the future: try to figure out some other extra implementations about this thesis.

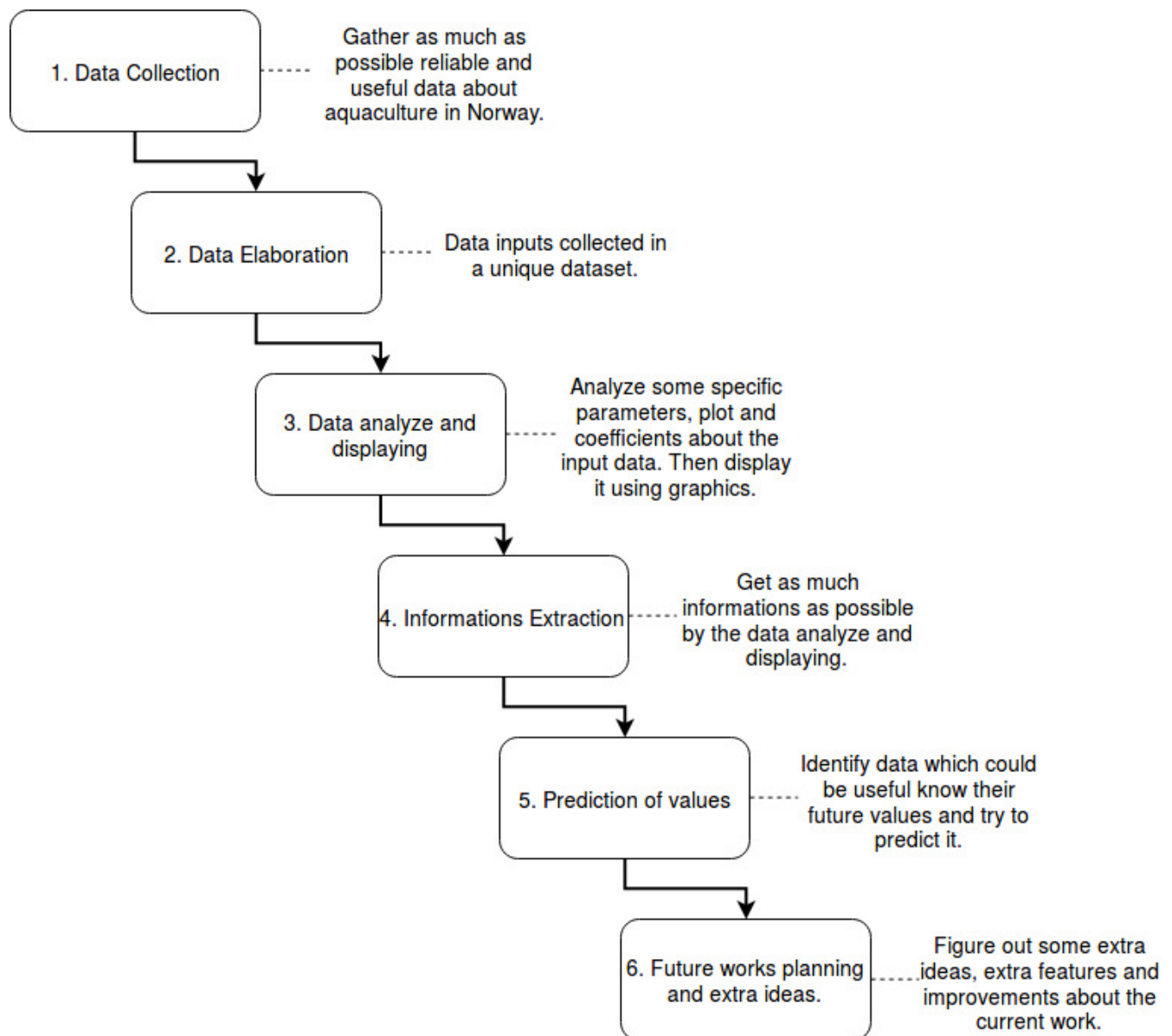


FIGURE 3.1: Plan flow chart

3.2 Github Usage

GitHub is a code hosting platform for version control and collaboration. It lets you and others work together on projects from anywhere.

In this thesis it will be used like "version control" since I'm the only person working on it, and I created on myself three repositories that will be very useful for better understand the work done with this thesis.

Repository that contains the Latex document about my thesis.

https://github.com/Sprea22/Thesis_Latex_Doc

Repository that contains the Data Analyzer implemented in python.

https://github.com/Sprea22/Data_Analyzer_Python

Repostory that contains the System for data forecasting implemented in python.

https://github.com/Sprea22/Forecasting_System_Python

Chapter 4

Implementation

4.0.1 Implemented Systems repositories

You can easily download my already implemented system for test it and better understand how it works.

Link for Total implementation Experiment 1 :

https://github.com/Sprea22/Data_Analyzer_Python

Link for Total implementation Experiment 2 :

https://github.com/Sprea22/Forecasting_System_Python

How to download and test it:

There are two ways for download it:

- Download the ZIP files from the following links:

Direct link for the already implemented Data Analyzer in Python.

https://codeload.github.com/Sprea22/Data_Analyzer_Python/zip/master

Direct link for the already implemented Forecasting System in Python.

https://codeload.github.com/Sprea22/Forecasting_System_Python/zip/master

- If you have already installed github on your computer, you can easily download it creating a folder and then inside that folder open a terminal shell and execute the following commands:

```
git init
git remote add origin https://github.com/Sprea22/
                        Data\_Analyzer\_Python.git
git pull origin master
```

Otherwise:

```
git init
git remote add origin https://github.com/Sprea22/
                        Forecasting\_System\_Python.git
git pull origin master
```

Once you downloaded it you will find a readMe inside both the repository that will explain your how to execute and how to test it.

Part I

Data collection and validation

4.1 Data collection

4.1.1 Data sources

During this phase is important to be sure to have all the data that we are going to need. During this particular implementation we need the 7 input dataset written above in the "Dataset structure" section, and below here you can find the link of the website where you can download all the needed datasets for this example.

Datasets downloads website:

<http://www.fiskeridir.no/Akvakultur/Statistikk-akvakultur/Biomassestatistikk>

The datasets that you can download on this website are in a different format than the one we will need, but for better understand the meaning of the data are useful since are in XLSX format, splitted in clear tables and well commented (only in norwegian).

4.1.2 Data description and validation

- Cages:

Content: Reported number of cages with salmon and rainbow trout.

Observation Frequency: Once per month

Period: From January 2005 to December 2016.

Location: Norway.

- Localities:

Content: Reported number of localities with salmon and rainbow trout.

Observation Frequency: Once per month

Period: From January 2005 to December 2016.

Location: Norway.

- Consumption of feed

Content: Reported feed consumption for Salmon.

Units: Figures in tonnes.

Observation Frequency: Once per month

Period: From January 2005 to December 2016.

Location: Norway.

- Fish at the end of the month (Salmon)

Content: Reported number of Salmon.

Units: Figures in 1000 pcs.

Observation Frequency: Once per month

Period: From January 2005 to December 2016.

Location: Norway.

- Biomass at the end of the month (Salmon)

Content: Reported biomass of Salmon.

Units: Figures in tonnes.

Observation Frequency: Once per month

Period: From January 2005 to December 2016.

Location: Norway.

- Fish restock (Salmon)

Content: Fish restock reported for Salmon.

Units: Figures in 1000 pcs.

Observation Frequency: Once per month

Period: From January 2005 to December 2016.

Location: Norway.

- Withdrawals (Salmon)

Content: Withdrawals of Salmon for slaughter.

Units: Figures in tonnes.

Observation Frequency: Once per month

Period: From January 2005 to December 2016.

Location: Norway.

- Monthly Salmon Price (US Dollar)

Content: Fish Salmon, Farm Bred Norwegian Salmon, export price, US Dollar per kg.

Units: US Dollars per Kilogram.

Observation Frequency: Once per month

Period: From January 2005 to December 2016.

Location: Norway.

4.2 Increase accessibility and availability of data

Quite complicated datasets structure rebuilt in a easy readable way, provided an accurate description (in English, since it was available just in Norwegian) and collected in a unique big dataset.

It allows to access to different kind of values about aquaculture in Norway in a much easier way.

Part II

Data Analysis and Displaying

4.3 Analysis of the data

Total implementation link for data analyzer :

https://github.com/Spree22/Data_Analyzer_Python

During this part the main purpose is to analyse the whole dataset in order to find some kind of useful informations later on.

The Python system implemented in this section is mainly used for a generic analysis of the data from different point of views.

The output of this phase will basically be for each single data input:

- Total graphic of the input data from 2005 to 2016.
- Graphic of the input data for each single year from 2005 to 2016.
- Correlation matrix between different months of the same input.
- Correlation matrix between different years of the same input.

And then it also provides:

- General correlation matrix between all the different inputs.
- Graphic of the normalized angular coefficients of all the inputs.

It's important to remind that this phase can be implemented in different ways and with different programming language, in this case has been choose Python, so be sure to have installed all the necessary for compile and execute Python code on your platform.

Current development environment:

Python version: 2.7.12

Linux kernel version number: Linux Asus 4.4.0-71-generic SMP

During this experiment has been implemented a system that could be divided in two subsystems, that are:

- Single Input Analyzer (SIA): Used for analyze a single data input.
- Multiple Inputs Analyzer (MIA): Used for analyze multiple data inputs.

4.3.1 Single Input Analyzer

- SIA imported libraries.
- SIA part I: Generate and display a graphic about current input with total data from 2005 to 2016.
- SIA part II: Generate and display a graphic about current input for each year from 2005 to 2016.
- SIA part III: Generate and display a graphic that contains the correlation matrix between each single year from 2005 to 2016 of the current input.
- SIA part IV: Generate and display a graphic that contains the correlation matrix between each single months of the year of the current input.
- SIA part V: Generate and display a single overview image for the current input.

4.3.1.1 SIA: Requirements for reusability

The system that is going to be implemented in this phase of the work could be used for other data inputs as well, but there are of course some kind of requiriments about the dataset that are necessary for let it works in a proper way.

The aalysis system need in input a dataset structure that:

- Data from January 2005 to December 2016
- One single value for each month

It means that the dataset must contains 144 values for each single input.

4.3.1.2 SIA section I: Total graphic for all the years

Goal:

Generate and display the total graphic about current input from 2005 to 2016.

Requirements:

- Data content: 144 values, 1 value for each month from 2005 to 2016

Implementation:

It's possible to check out the full ccommented code in the appendice: [A.2](#)

Results:

With this first part of the code has been reached the first goal of to displaying and save the basic graphic about the current input from 2005 to 2016, with also the trendline displayed, that looks like this example:

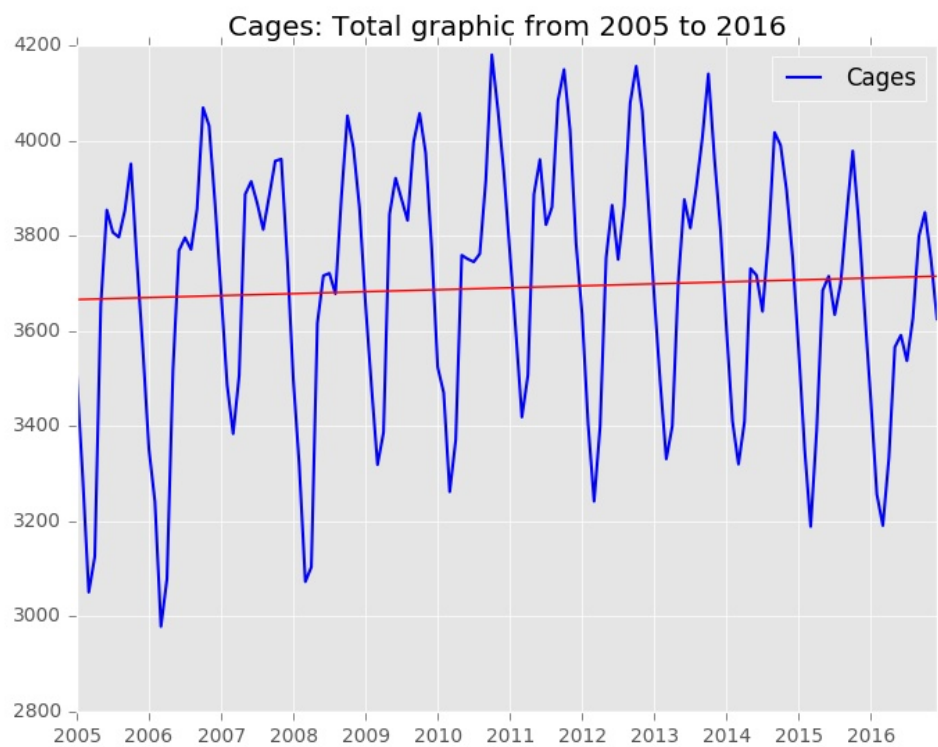


FIGURE 4.1: Total graphic about current input with total data from 2005 to 2016.

4.3.1.3 SIA section II: Single graphics for each year

Goal:

Generate and display graphics about current input for each year from 2005 to 2016

Requirements:

- Data content: 144 values, 1 value for each month from 2005 to 2016

Results:

With this second part of the code has been reached the goal of displaying and save the graphics about the current input for each single year from 2005 to 2016, that looks like this example:

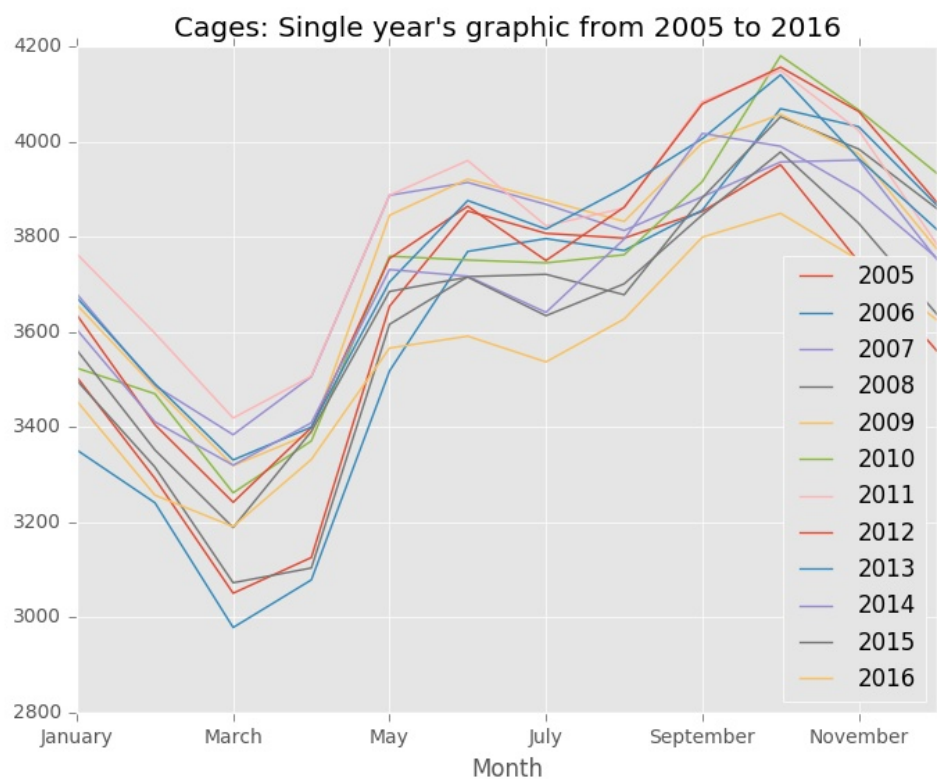


FIGURE 4.2: Graphics for each single year of the input data from 2005 to 2016

4.3.1.4 SIA section III: Correlation matrix between years

Goal:

Calculate the correlation coefficients between each single year from 2005 to 2016 of the current input and then display it with a correlation matrix.

Requirements:

- Data content: 144 values, 1 value for each month from 2005 to 2016

Results:

With this third part of the code have been calculated the correlation coefficients between each single year from 2005 to 2016 for each single inputs, saved the values in a document and then also displayed and saved the correlation matrix about it, that looks like this example:

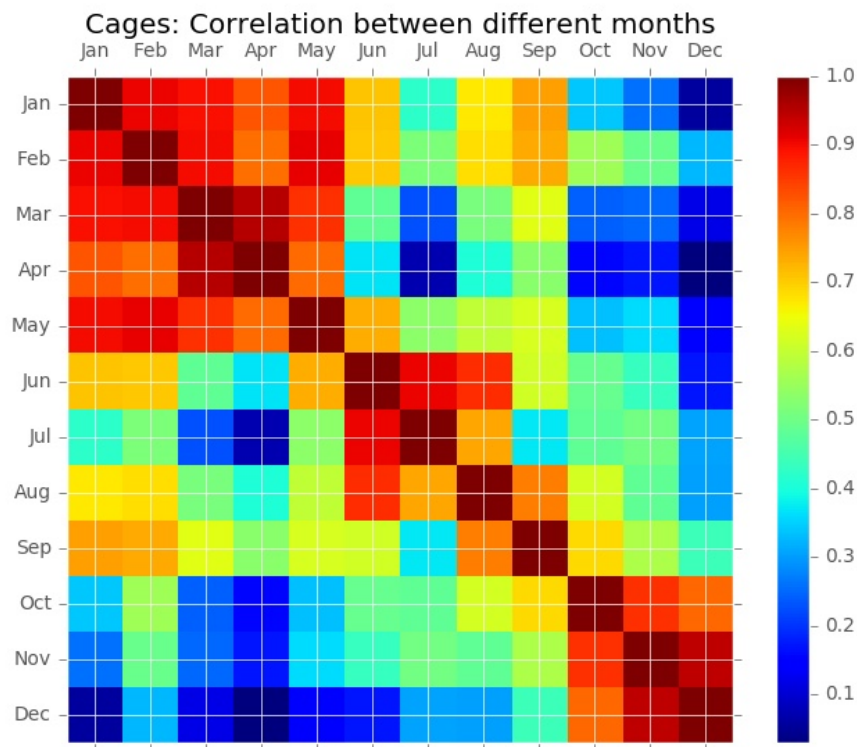


FIGURE 4.3: Correlation matrix between different months of the same input

4.3.1.5 SIA section IV: Correlation matrix between months

Goal:

Calculate the correlation coefficients between each single month from 2005 to 2016 of the current input and then display it with a correlation matrix.

Requirements:

- Data content: 144 values, 1 value for each month from 2005 to 2016

Results:

With this part of the code have been calculated the correlation coefficients between each single month from 2005 to 2016 of all the inputs and saved it in a document. Then has been also displayed and saved the correlation matrix about it, that looks like this example:

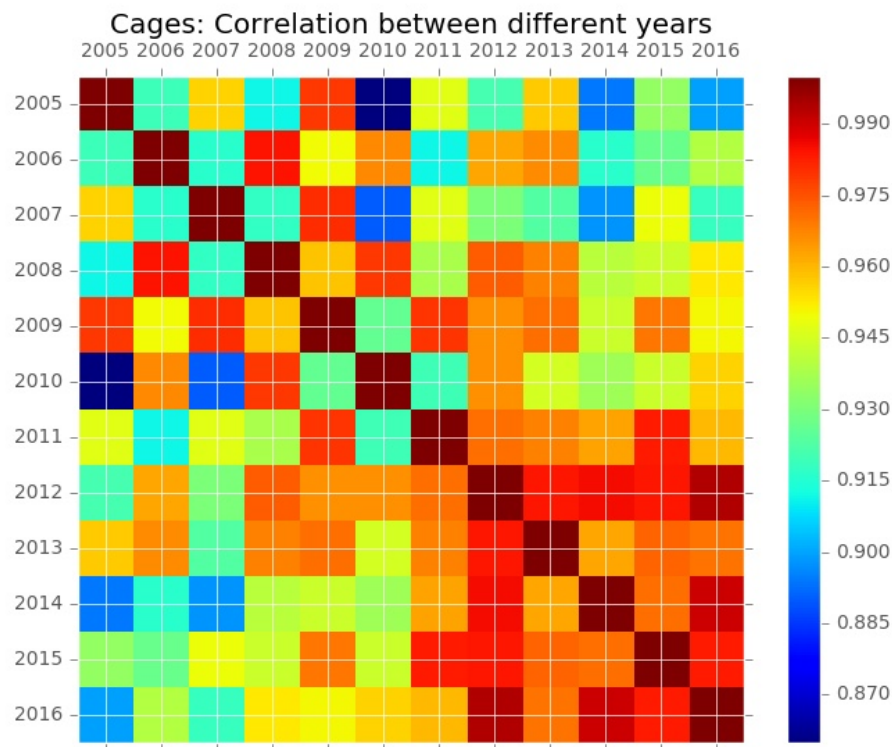


FIGURE 4.4: Correlation matrix between different years of the same input

4.3.1.6 SIA section V: Single overview

Goal:

Generate and display a single overview image for the current input.

Requirements:

- All the images that contain graphic about current input.

Results:

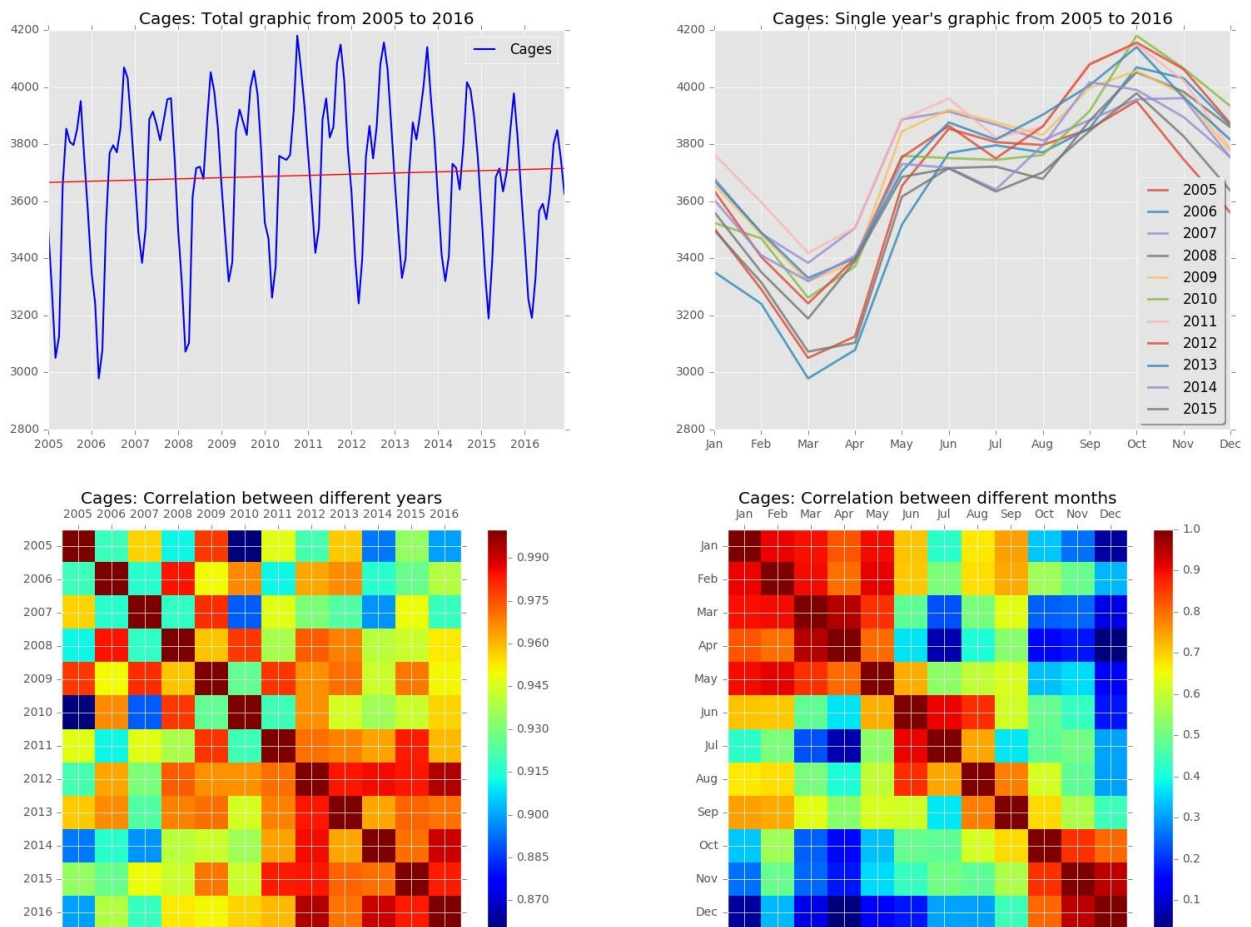


FIGURE 4.5: Example of "Single Input Overview Image"

4.3.2 Multiple Inputs Analyzer

4.3.2.1 MIA: Requirements for reusability

The system that is going to be implemented during this phase it's not reusable at all. It means that has been implemneted just for analyze the current dataset composed by 8 different inputs about Aquaculture.

4.3.2.2 MIA: Implementation

Goal:

This analyzer is mainly used for show the correlation coefficient between the diffent inputs along the total period (from 2005 o 2016), that it will be important to have a general about which kind of relation there is between different inputs and how much strong it is.

Requirements:

- Input dataset: Total_Dataset required, structure already reported here: Total Dataset

Results:

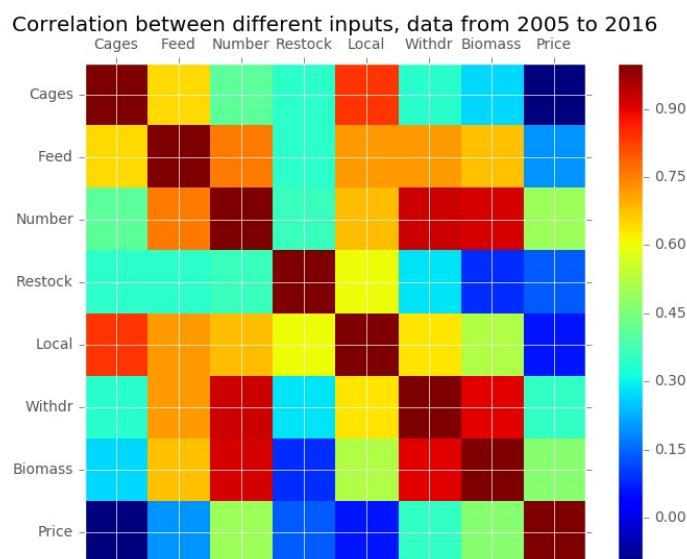


FIGURE 4.6: Correlation matrix between different inputs with data from 2005 to 2016.

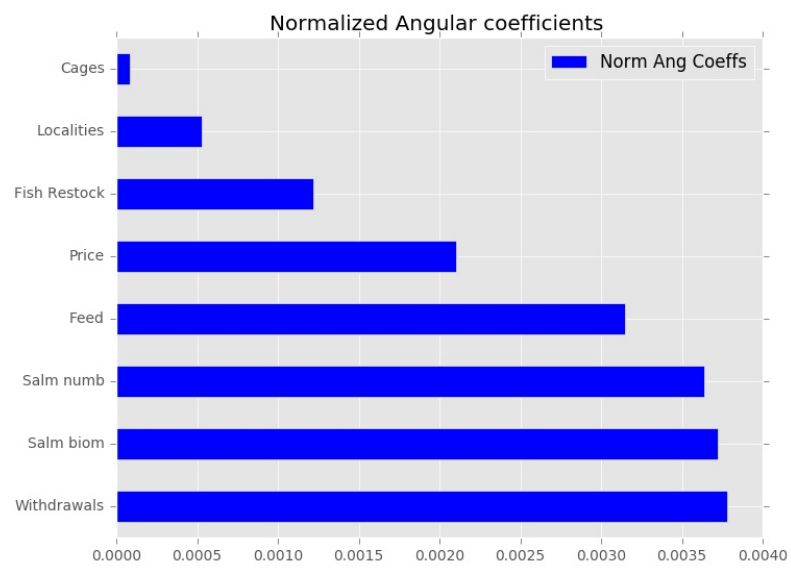


FIGURE 4.7: Normalized angular coefficients of each input's trendline.

4.4 Extract information from data

Part III

Data Prediction

4.5 Prediction of values about the data

Some basic and general goals were defined before starting this phase, with the idea of "doing more if it's possible". Basically the main purpose was the one of, after the previous analysis, predict some values and evaluate the quality of the results. This prediction system was not defined with some specific requirements, so the first main problem was to find a reliable, accurated and user-friendly way to predict and display prediction of values.

Since the current dataset can be considered like a time series, in this phase we will develop the data prediction system using an ARIMA machine implemented in python.

The ARIMA machine can be configured with several configurations, it allows you to have more accurated results; so the first thing was to find the right configuration of the ARIMA machine of each single input which we are interested to forecast.

During this phase of the work have been implemented 3 different subsystems for different purposes:

1. Evaluating System
2. Training System
3. Future Prediction System

4.5.1 Evaluating System

Goal:

Used for evaluate different configurations of ARIMA machine.

It tests 112 different configurations for each single input that we would like to forecast and report the results with each MAPE (Mean Average Percentage Error) values.

Requirements:

There are not strict requirements needed. There are no type of restriction neither about the length or about the type of data.

Code implementation:

The most important part of the code about the Evaluating System is the following.

Basically the method `ARIMA()` allows to train a model based on historic values (history) and a specific order (p,d,q). After that it's possible to call the method `forecast()` through the trained model and having some predictions like result.

```
model = ARIMA(history, order=arima_order)
model_fit = model.fit(dispatch=0)
yhat = model_fit.forecast()[0]
```

This system will provide 112 different ARIMA configurations results for each single input, and in particular it will display the best ARIMA configuration, that is the one with the lower MAPE.

Results:

The system will display the MAPE between real value and predicted values for each single tested ARIMA machine, in particular the configuration that gives the best result. All these results have been reported in a document and then also displayed with a 3D graphic that allows to see the MAPE value for each different order in input.

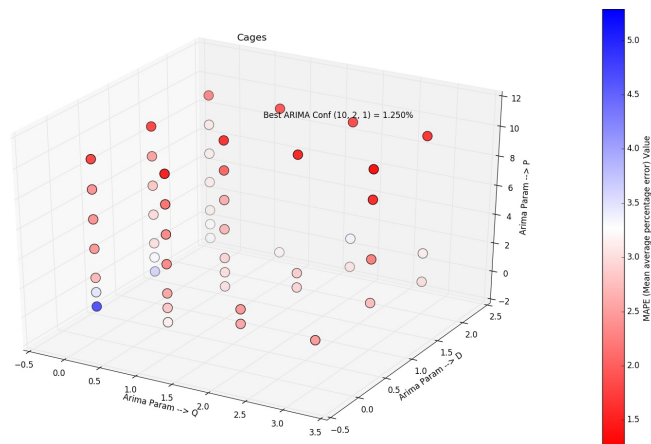


FIGURE 4.8: Graphic that displays different MAPE values for each ARIMA order.

4.5.2 Training System

Goal:

This system has the goal of training/testing a particular specific ARIMA configuration about a data input, and see how much accurate it is.

Requirements:

Since this Training System has been used mainly for train and test the current dataset, it need to have like input a dataset that follows the same format: - Data content: 144 values, 1 value for each month from 2005 to 2016

Code implementation:

First of all this system it's going to split the input data in two part:

- Train data: values which the ARIMA model is going to use for training.
- Test data: values which are hided by the forecasting model.

Once the ARIMA model has been created, the system will try to predict the future values, that are actually the "Test data".

Once the predictions have been calculated it's possible to display the "Test data" (that are the real values) and the predicted values, just to see how much the ARIMA configuration is accurate.

```
model = ARIMA(history, order=arima_order)
model_fit = model.fit(dispatch=0)
yhat = model_fit.forecast()[0]

series = pd.read_csv("Dataset.csv", usecols=[sys.argv[1]])
series.plot(color="blue", linewidth=1.5,
            label="Series: "+sys.argv[1])

output = Series.from_csv('Output_Files/predictions.csv')
output.plot(color="red", linewidth=1.5,
            label="Prediction test:")
```

Results:

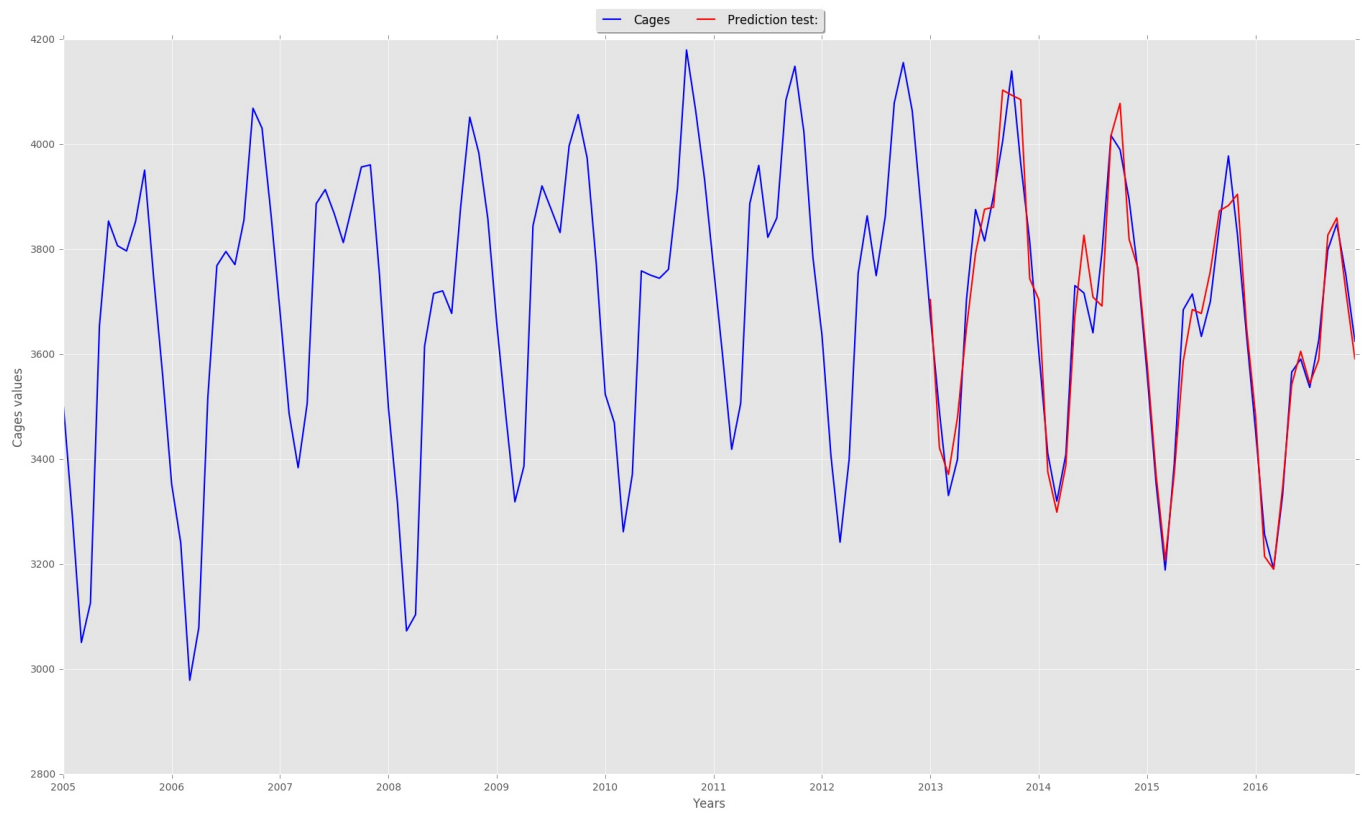


FIGURE 4.9: Graphic that displays the predicted values from a particular ARIMA machine configuration and the historic real values.

4.5.3 Future Prediction System

Goal:

This part of the work has the goal of display some real prediction of values in the future. It basically collects real future values, in this particular are values about 2017, of each single input and then, once calculated also the prediction values, it's going to display on the same graphic:

- Historic values
- Predicted future values
- Real values (if are available)

Requirements:

This system has to be as much reusable as possible, so there are not that strict requirements. You can reuse this Future Prediction System with any kind of dataset with no restrictions about length.

The only requirement to let it works in a proper way is that you have to set the historic and real values datasets in the right way; it means that you have to write down the historic values in the dataset in this way:

Index — Input

1 — Value1

2 — Value2

3 — Value3

.. — ..

120 — Value120

121 — Value121

122 — Value122

And then, if you want to compare the predicted values with some real values that are already available, you have to set the real values dataset in this way:

Index — Input

123 — Value123

124 — Value124

125 — Value125

126 —

127 —

128 —

Code implementation:

```

model = ARIMA(history, order=arima_order)
model_fit = model.fit(dis=0)
yhat = model_fit.forecast()[0]

# 1) Historic values
series = pd.read_csv("HISTORIC DATASET",
                    usecols=[sys.argv[1]])
series.plot(color="blue", linewidth=1.5)

# 2) Predicted future values
series = Series.from_csv("PREDICTIONS DATASET")
series.plot(color="red", linewidth=1.5,
            label="Prediction Results")

# 3) Real future values
series = pd.read_csv("REAL VALUES DATASET")
pyplot.plot(series["Index"], series[sys.argv[1]],
            color="green", linewidth=1.5, label="Real values")

```

Results:

The system implemented during this phase allows to predict future for as many months as you want in the future and to display it, compared also with the historic values and real values once are available.

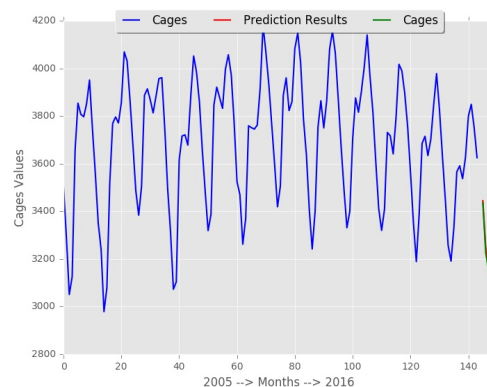


FIGURE 4.10: Graphic that display historic, future and predicted values of a input.

4.6 Requirements for reusability

The subsystems implemented during this phase of the work are almost completely reusable.

The reusability this systems allows to get some prediction of values about different kind of dataset, in particular:

- The "Evaluating System" is actually 100% reusable, and you can use it for evaluate any kind of dataset.
- The "Future Prediction System" is completely reusable as well, you should only modify the historic values and the real values inside the dataset for it works in a proper way.
- The "Training System" has been implemented for testing the current dataset, so it's not completely reusable but could be changed very easily and let it works also for other data input.

Part IV

Future Works

4.7 Visualization of the data

4.8 Prediction system as a service

Chapter 5

Results

Chapter 6

Discussion

Chapter 7

Conclusion

Chapter 8

Bibliography

- [1] <http://www.fiskeridir.no/Akvakultur/Statistikk-akvakultur/Biomassestatistikk>
- [2] https://www.quandl.com/data/ODA/PSALM_USD-Fish-Salmon-Price
- [3] <http://machinelearningmastery.com/time-series-forecasting/>
- [4] https://github.com/Sprea22/Thesis_Latex_Doc
- [5] https://github.com/Sprea22/Data_Analyzer_Python
- [6] https://github.com/Sprea22/Forecasting_System_Python
- [7] https://en.wikipedia.org/wiki/Data_science

Appendix A

SIA Implementation code

A.1 SIA: Imported libraries

The library "os" is really important since provides a waay of using operating system dependent functionality.

```
import os
```

Also the library "sys" would be very useful for test and execute the program, mainly because it allows to input directly from terminal.

```
import sys
```

The "pylab" library will be useful for plot data.

```
import pylab
```

The "pandas" library will be very useful for read the data from CSV dataset and setup the plot abut it.

```
import pandas as pd
```

The "numpy" library it's used for mathematic purpose, such as calculating the correlation coefficient between two series.

```
import numpy as np
```

The "pyplot" library it's used for basic graphic displaying and customization, easy to use but very efficient.

```
import matplotlib.pyplot as pyplot
```

The library "PIL" supports many file formats, and provides powerful image processing and graphics capabilities.

```
from PIL import Image
```

The library "fpdf" allows to generate and use PDF file.

```
from fpdf import FPDF
```

A.2 SIA section I: Total graphic for all the years

Code implementation:

During this section of the code was used "pandas" library for read the dataset.

```
series = pd.read_csv("Dataset.csv", usecols=[1,sys.argv[1]])
```

Then using the "pyplot" library has been possible to setup the plot of the input data.

```
series.plot(color="blue", linewidth=1.5)
```

There are some settings about the axis x just to display the data in the right format, are easy to change and to costume.

```
years = ["2005","2006","2007","2008","2009","2010",
         "2011","2012","2013","2014","2015","2016"]
x = range(144)
pyplot.xticks(np.arange(min(x), max(x)+1, 12.0), years)
pyplot.title("Total graphic from 2005 to 2016")
```

Once setted up the plot of the current data, the next step was to display the trendline of the current graphic.

The following code represent the method for calculate and display it.

```
def trendline(x, y, col):
    z = np.polyfit(x, y, 1)
    p = np.poly1d(z)
    pylab.plot(x,p(x), c=col)
    # print "y=%.6fx+(%.6f)"%(z[0],z[1])
```

At this point the current data values have been read again and passed to the method just implemented above for calculating the trendline.

```
series2 = pd.read_csv("Dataset.csv", usecols=[sys.argv[1]],
                      squeeze=True)
trendline(x, seriesV.values, "red")
```

There is the possibility to save the graphic like an image and/or display it.

```
saveFigure("_Total.jpg")
```

A.3 SIA section II: Single graphics for each year

Code implementation:

During this section of the code was used "pandas" library for read the dataset.

```
series2 = pd.read_csv("Dataset.csv", index_col=['Month'],
                      usecols=[0,1,sys.argv[1]])
```

Some initialization of variables that are going to be useful.

```
months = ["Jan", "Feb", "Mar", "Apr", "May", "Jun",
          "Jul", "Aug", "Sep", "Oct", "Nov", "Dec"]
x_pos = np.arange(len(months))
test = []
j = 0
```

The following code allows the system to split the values and display them in the right way: that means that are going to be splitted for each single year and then plotted on the same graphic.

```
for i in range(len(series2.values)):
    if j in range(12):
        test.append(series2.values[i][1])
        j = j + 1
    else:
        pyplot.plot(x_pos, test, linewidth=2,
                    alpha=0.8,
                    label = int(series2.values[i-1][0]))
        test = []
```

```
test.append(series2.values[i][1])
j = 1
```

These are some personalization settings that could be easily changed as you want.

```
ax.legend(loc=4, ncol=1, fancybox=True, shadow=True)
pyplot.xticks(x_pos, months)
pyplot.xlim(0, 11)
pyplot.title(sys.argv[1]+
              ": Single year's graphic from 2005 to 2016")
```

There is the possibility to save the graphic like an image and/or display it.

```
saveFigure("_Years.jpg")
```

A.4 SIA section III: Correlation matrix between years

Code implementation:

During this section of the code was used "pandas" library for read the dataset.

```
series3 = pd.read_csv("Dataset.csv", index_col=['Year'],
                      usecols=[0, sys.argv[1]])

corr = []
test = []
j = 0
# Collecting the correct values to elaborate.
for i in range(len(series2.values)+1):
    if j in range(12):
        test.append(series2.values[i][1])
        j = j + 1
    else:
        corr.append(test)
        test = []
        if i in range(144):
            test.append(series2.values[i][1])
            j = 1
```

With the library "numpy" is possible to calculate the correlation coefficients between all the variables in the series just read.

```
test = np.corrcoef(corr)
```

Setup the figure that will display the correlation matrix using the library "pypot".

```
fig2 = pyplot.figure()  
ax = fig2.add_subplot(111)
```

Creating the correlation matrix using the already calculated correlation coefficients.

```
cax = ax.matshow(test, interpolation='nearest')
```

Settings for display the matrix in the right way, in particular for the values to display on both the axis x and y, in this case every single year from 2005 to 2016

```
pyplot.title(sys.argv[1]+ ": Correlation between different years")  
years = ["2005", "2006", "2007", "2008", "2009", "2010",  
         "2011", "2012", "2013", "2014", "2015", "2016"]  
x_pos = np.arange(len(years))  
y_pos = np.arange(len(years))  
pyplot.yticks(y_pos, years)  
pyplot.xticks(x_pos, years)  
pyplot.colorbar(cax)
```

Adding a title to the graphic that we are going to display and also a bar that works like a legend for the colors of the matrix, allowing the reader to better understand the values reported inside the matrix.

```
saveFigure("_Years_Matrix.jpg")
```

There is the possibility to save the correlation matrix like an image and/or display it.

```
pyplot.savefig("OUTPUT_DIRECTORY", format="jpg")
pyplot.show()
```

A.5 SIA section IV: Correlation matrix between months

Code implementation:

During this section of the code was used "pandas" library for read the dataset.

```
series4 = pd.read_csv("Dataset.csv", usecols=[0,1,sys.argv[1]])

corr = []
for Month, Year in series4.groupby(["Month"], sort=False):
    corr.append(Year[sys.argv[1]].values)
```

With the library "numpy" is possible to calculate the correlation coefficients between all the variables in the series just read.

```
test = np.corrcoef(series4.values)
```

Setup the figure that will display the correlation matrix using the library "pypot".

```
fig2 = pyplot.figure()
ax = fig2.add_subplot(111)
```

Creating the correlation matrix using the already calculated correlation coefficients.

```
cax = ax.matshow(test, interpolation='nearest')
```

Settings for display the matrix in the right way, in particular for the values to display on both the axis x and y, in this case every single months of the year.

```
months = ["Jan", "Feb", "Mar", "Apr", "May", "Jun",
          "Jul", "Aug", "Sep", "Oct", "Nov", "Dec"]
x_pos = np.arange(len(months))
```

```
y_pos = np.arange(len(months))
pyplot.yticks(y_pos, months)
pyplot.xticks(x_pos, months)
```

Adding a title to the graphic that we are going to display and also a bar that works like a legend for the colors of the matrix, allowing the reader to better understand the values reported inside the matrix.

```
pyplot.title("Correlation between different months")
pyplot.colorbar(cax)
```

There is the possibility to save the correlation matrix like an image and/or display it.

```
saveFigure("_Months_Matrix.jpg")
pyplot.show()
```

A.6 SIA section V: Single overview

Code implementation:

`create_single_overview()` : this method will use the "Image" library for autogenerate a collage of the current input's graphics and save it like an overview image. The content of the params will basically decide how the "Current input overview image" will look like.

It uses each single "current input overview image" of all the inputs and the "correlation matrix between all the inputs image" for combine them in a unique "total overview" and save it using the PDF format.

```
listofimages=["CURRENT_INPUT_TOTAL_GRAPHIC",
              "CURRENT_INPUT_YEARS_MATRIX",
              "CURRENT_INPUT_YEARS_GRAPHIC",
              "CURRENT_INPUT_MONTHS_MATRIX"]

create_single_overview(params1, listofimages)
create_single_overview(params2, listofimages)
```

The "create_single_overview" method has basically this structured, and then its configuration depends from the input data and from the preferences.

```
def create_single_overview(cols, rows ,
                           width, height, listofimages):
    thumbnail_width = width//cols
    thumbnail_height = height//rows
    size = thumbnail_width, thumbnail_height
    new_im = Image.new('RGB', (width, height))
    ims = []
    for p in listofimages:
        im = Image.open(p)
        im.thumbnail(size)
        ims.append(im)
    i = 0
    x = 0
    y = 0
    for col in range(cols):
        for row in range(rows):
            new_im.paste(ims[i], (x, y))
            i += 1
            y += thumbnail_height
        x += thumbnail_width
        y = 0
    new_im.save(SINGLE_OVERVIEW_IMAGE")
    new_im.show()
```

Appendix B

MIA Implementation code

B.1 MIA: Imported libraries

The "pandas" library will be very useful for read the data from CSV dataset and setup the plot about it.

```
import pandas as pd
```

The "numpy" library it's used for mathematic purpose, such as calculating the correlation coefficient between two series.

```
import numpy as np
```

The "pyplot" library it's used for basic graphic displaying and customization, easy to use but very efficient.

```
import matplotlib.pyplot as pyplot
```

B.2 MIA: Implementation

Code implementation:

First of all, we are going to use the "pandas" library for read the dataset.

```
series3 = pd.read_csv("TOTAL_DATASET_DIRECTORY",  
                      index_col=['Input'], header=0)
```

Then with the library "numpy" is possible to calculate the correlation coefficients between all the variables just read above.

```
test = np.corrcoef(series3.values)
```

Setup the figure that will display the correlation matrix using the library "pyplot".

```
fig2 = pyplot.figure()
ax = fig2.add_subplot(111)
```

Creating the correlationg matrix using the already calculated correlation coefficients.

```
cax = ax.matshow(test, interpolation='nearest')
```

Settings for display the matrix in the right way, in particular for the values to display on both the axis x and y, in this case every single input.

```
inputs = ["Cages", "Feed", "Number", "Restock",
          "Local", "Withdr", "Biomass", "Price"]
x_pos = np.arange(len(inputs))
y_pos = np.arange(len(inputs))
pyplot.yticks(y_pos, inputs)
pyplot.xticks(x_pos, inputs)
```

Adding a title to the graphic that we are going to display and also a ba that works like a legend for the colors of the matrix, allowing the reader to better understand the values reported inside the matrix.

```
pyplot.title("Correlation between different inputs
             about data from 2005 to 2016")
pyplot.colorbar(cax)
```

In the end, using again the library "pyplot", there is the possibility to save the correlation matrix graphic like an image and/or display it.

```
pyplot.savefig("OUTPUT_DIRECTORY")
```

```
series = pd.read_csv("TRENDLINES_VALUES_DOCUMENT",
                    header=0, usecols=["Norm Ang Coeffs"])
series.plot(kind="barh")
pyplot.savefig("OUTPUT_DIRECTORY")
```

```
create_total_overview()
```

```
pyplot.show()
```

Appendix C

Prediction System Implementation code

C.1 Evaluating System

C.2 Training System

C.3 Future Prediction System