UNIVERSITY OF TROMSØ

Initial system's implementation for Analysis and Display data 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

April 2017

Declaration of Authorship

I, Andrea Spreafico, declare that this thesis titled, 'Initial system's implementation for Analysis and Display data 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

List of Figures

List of Tables

Abbreviations

LAH	**L**ist **A**bbreviations **H**ere
OWL	**O**rdinary **W**izarding **L**evel

Physical Constants

Speed of Light $c = 2.997\ 924\ 58 \times 10^8$ ms^{-S} (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 away, 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 statistcs, 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 Dataset Structure

- Cages:

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

Period: Each single month from 2005 to 2016.

Location: Norway.

- Localities:

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

Period: Each month from 2005 to 2016.

Location: Norway.

- Consumption of feed

Content: Reported feed consumption for Salmon.

Units: Figures in tonnes.

Period: Each month from 2005 to 2016.

Location: Norway.

- Fish at the end of the month (Salmon)

Content: Reported number of Salmon.

Units: Figures in 1000 pcs.

Period: Each month from 2005 to 2016.

Location: Norway.

- Fish restock (Salmon)

Content: Fish restock reported for Salmon.

Units: Figures in 1000 pcs.

Period: Each month from 2005 to 2016.

Location: Norway.

- Withdrawals (Salmon)

Content: Withdrawals of Salmon for slaughter.

Units: Figures in tonnes.

Period: Each month from 2005 to 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.

Period: Each month from 2005 to 2016.

Location: Norway.

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

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.

In the computer science area in particular 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.

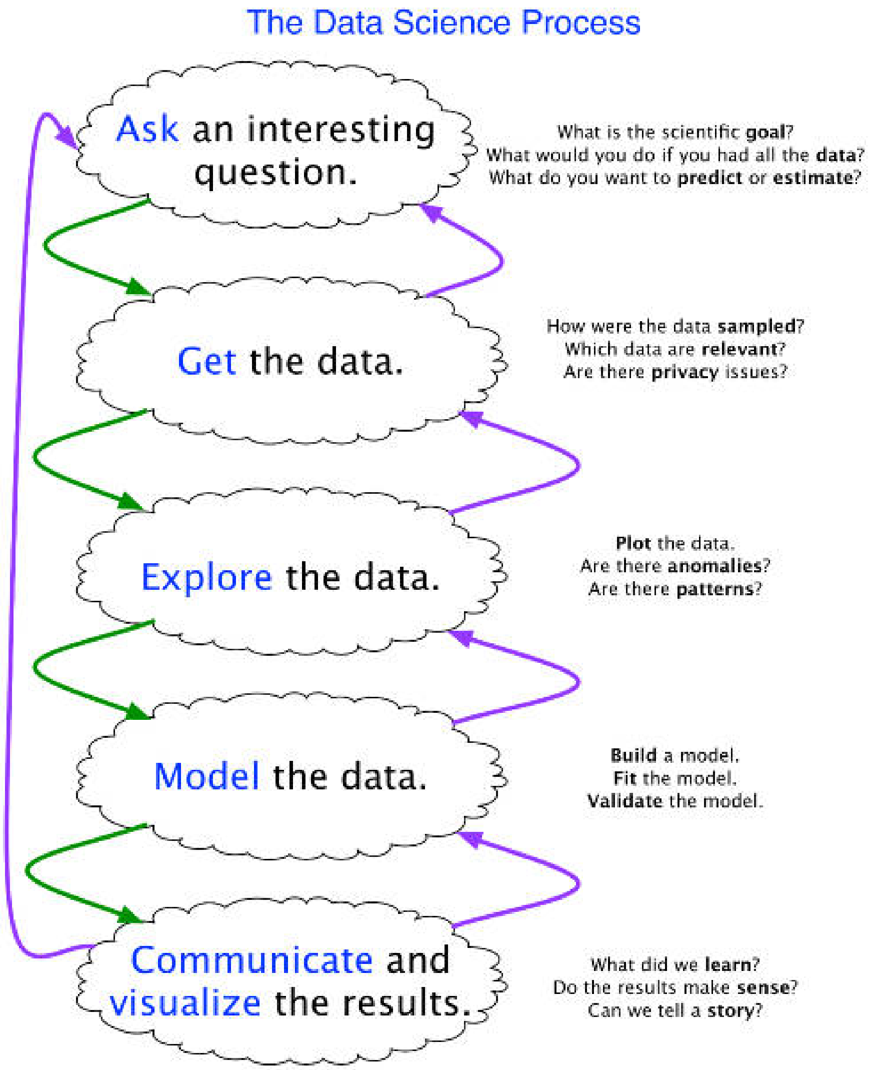
The Data Science Process
Ask an interesting question.
What is the scientific goal?
What would you do if you had all the data?
What do you want to predict or estimate?
Get the data.
How were the data sampled?
Which data are relevant?
Are there privacy issues?
Explore the data.
Plot the data.
Are there anomalies?
Are there patterns?
Model the data.
Build a model.
Fit the model.
Validate the model.
Communicate and visualize the results.
What did we learn?
Do the results make sense?
Can we tell a story?

Figure 2.1: 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,[2] 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.

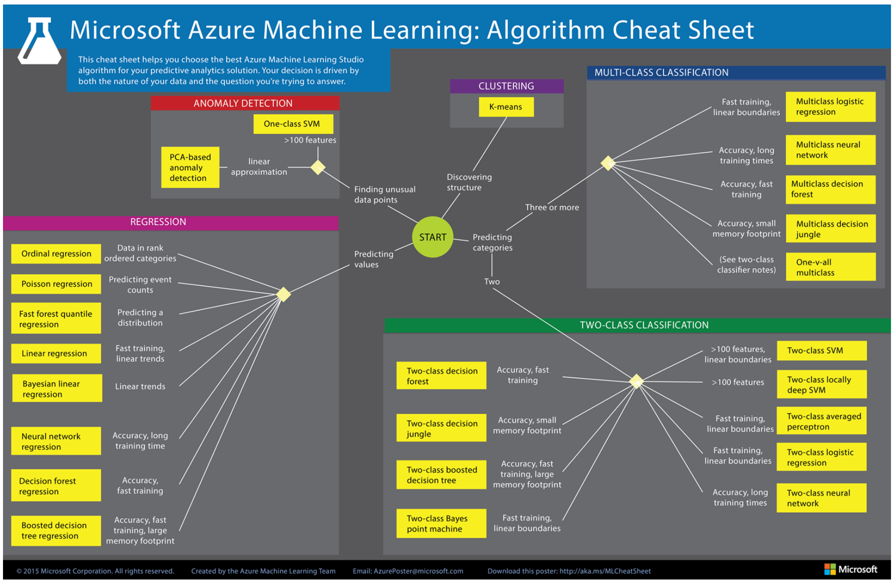
Microsoft Azure Machine Learning: Algorithm Cheat Sheet
This cheat sheet helps you choose the best Azure Machine Learning Studio algorithm for your predictive analytics solution. Your decision is driven by both the nature of your data and the question you're trying to answer.
ANOMALY DETECTION
One-class SVM
>100 features
PCA-based anomaly detection
linear approximation
Finding unusual data points
CLUSTERING
K-means
Discovering structure
START
Predicting categories
Three or more
Two
Predicting values
MULTI-CLASS CLASSIFICATION
Fast training, linear boundaries
Multiclass logistic regression
Accuracy, long training times
Multiclass neural network
Accuracy, fast training
Multiclass decision forest
Accuracy, small memory footprint
Multiclass decision jungle
(See two-class classifier notes)
One-v-all multiclass
REGRESSION
Ordinal regression
Data in rank ordered categories
Poisson regression
Predicting event counts
Fast forest quantile regression
Predicting a distribution
Linear regression
Fast training, linear trends
Bayesian linear regression
Linear trends
Neural network regression
Accuracy, long training time
Decision forest regression
Accuracy, fast training
Boosted decision tree regression
Accuracy, fast training, large memory footprint
TWO-CLASS CLASSIFICATION
Two-class decision forest
Accuracy, fast training
Two-class decision jungle
Accuracy, small memory footprint
Two-class boosted decision tree
Accuracy, fast training, large memory footprint
Two-class Bayes point machine
Fast training, linear boundaries
>100 features, linear boundaries
Two-class SVM
>100 features
Two-class locally deep SVM
Fast training, linear boundaries
Two-class averaged perceptron
Fast training, linear boundaries
Two-class logistic regression
Accuracy, long training times
Two-class neural network
© 2015 Microsoft Corporation. All rights reserved.
Created by the Azure Machine Learning Team
Email: AzurePoster@microsoft.com
Download this poster: http://aka.ms/MLCheatSheet
Microsoft

Figure 2.2: Machine learning algorithms

2.4 Python

Chapter 3

Development Method

In order to achieve the problem reported in the introduction, has been developed an easy but clear plan (how you can see in the follow picture 3.1).

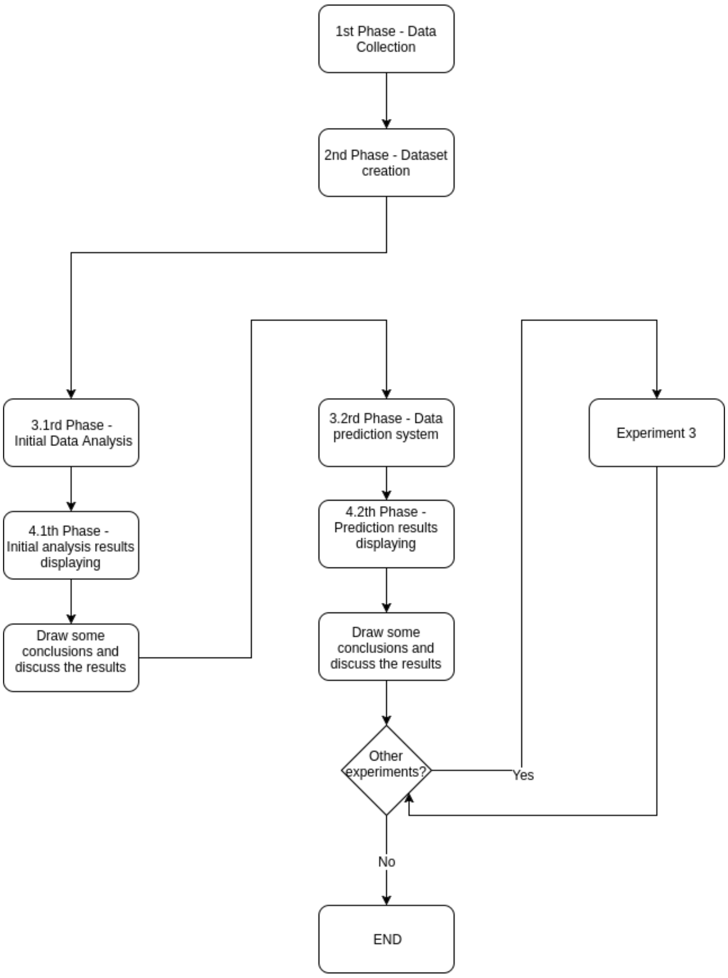
1st Phase - Data Collection
2nd Phase - Dataset creation
3.1rd Phase - Initial Data Analysis
4.1th Phase - Initial analysis results displaying
Draw some conclusions and discuss the results
3.2rd Phase - Data prediction system
4.2th Phase - Prediction results displaying
Draw some conclusions and discuss the results
Experiment 3
Other experiments?
Yes
No
END

Figure 3.1: Plan flow chart

1st Phase: Data collection 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 they are going to be indispensable for the next phases and in particular the final result and conclusion. The data's reliability and utility mainly depend by the kind of sources where you're able to mine.

2nd Phase: Dataset creation During this phase you should customize the unstructured data that you collected in the first phase. This data's customizing has the main purpose to let the data structure follow some kind of setting and standard needed in the system that will be implemented later on. Part of this phase has to be done meanwhile you're working on the successive phase, since you're going to discover a lot of needed setting and standard once you're trying to feed the system.

3rd Phase: Data analysis 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 then for the predictions system.

3.1rd Phase – Initial data analysis system:

3.2rd Phase – Data prediction system:

4th Phase: Data displaying This is the last but not least phase since displaying some information in a easy-read mode sometimes is very difficult. During this phase the main purpose will be to find a way to display the analysis results in a way that could be easily understand.

4.1th Phase - Initial analysis results displaying

4.2th Phase - Prediction results displaying

Chapter 4

Implementation

Total implementation link for Experiment 1 :
https://github.com/Sprea22/Data_Analyzer_Python

Total implementation link for Experiment 2 :

(INSTRUCTION FOR DOWNLOAD AND USE FROM GITHUB)

4.1 1st Phase: Data collection

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).

4.1.1 Data sources

4.1.2 Data description and validation

4.2 2nd Phase: Dataset creation

4.2.1 Processing the data

Once we have all the ”row” data we can start to setup the dataset that we will need for this system. First of all we will use the format CSV for the new datasets instead of XLSX. Then we need to create three different ”standards” of each input dataset. The reason of this requirement is that using the python library for some kind of analysis is easier to read the data in a particular format instead of reading the data always in the normal CSV standard and then edit and handle it later through the code. Following are reported the needed standards.

- Normal CSV standard (Standard_N)

Month	Input_Name
January_2005	Value1
February_2005	Value2
March_2005	Value3
...	...
November_2016	Value143
December_2016	Value144

Table 4.1: Dataset Normal Standard

- Month standard (Standard_M)

Month	2005	2006	2007	...	2016
January	value1	value13	value25	...	value133
February	value2	value14	value26	...	value134
March	value3	value15	value27	...	value135
...	...	...	...	...	...
December	value12	value24	value36	...	value144

Table 4.2: Dataset Month Standard

- Year standard (Standard_Y)

Year	January	February	March	...	December
2005	value1	value2	value3	...	value12
2006	value13	value14	value15	...	value24
2007	value25	value26	value27	...	value36
...	...	...	...	...	...
2016	value133	value134	value135	...	value144

Table 4.3: Dataset Year Standard

The last step of this phase is to setup the final dataset, creating for each data input three different CSV file that are following the three standard reported above.
The following strucutre show how our final dataset will looks like once we apply the standards on each one of our data inputs.

1. Cages
    - Cages.csv : Contains the data about "Cages" following the Stand_N
    - Cages_M.csv : Contains the data about "Cages" following the Stand_M
    - Cages_Y.csv : Contains the data about "Cages" following the Stand_Y
2. Localities
    - Localities.csv : Contains the data about "Localities" following the Stand_N
    - Localities_M.csv : Contains the data about "Localities" following the Stand_M
    - Localities_Y.csv : Contains the data about "Localities" following the Stand_Y

    Continue and do the same thing for all the other data input
3. Monthly_salmon_price
4. Salmon_biomass_end_month
5. Salmon_consumption_of_feed
6. Salmon_number_end_month
7. Salmon_restock
8. Salmon_withdrawals

And in the end we will create also a total dataset of all the different inputs, that we will call it Total_Dataset.csv and looks like:

Input	January_2005	February_2005	March_2005	...	November_2016	December_2016
Input1	value1.1	value1.2	value1.3	...	value1.143	value1.144
Input2	value2.1	value2.2	value2.3	...	value2.143	value2.144
Input3	value3.1	value3.2	value3.3	...	value3.143	value3.144
...	...	...	...	...	...	
Input7	value7.1	value7.2	value7.3	...	value7.143	value7.144
Input8	value8.1	value8.2	value8.3	...	value8.143	value8.144

Table 4.4: Total Dataset

Part I

Experiment 1: Data analysis system

4.3 3nd Phase: Analyzer system implementation

Total implementation link for Experiment 1 :
https://github.com/Sprea22/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 that we are going to implement is mainly used for a generic analysis of the data under 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.
- Comparison of the graphics for each single year of the input data from 2005 to 2016.
- Correlation matrix between different months of the same input.
- Correlation matrix between different years of the same input.

It's important to remind that this phase can be implemented in different ways and with different programming language, in this case the programming language choosen is Python, so be sure to have installed all the necessary for compile and execute Python code on your platform.
(PYTHON REQUIREMENTS)
During this experiment we are going to implement a system that is basically divided in two subsystems, that are:

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

4.3.1 Imported libraries

During the implementation of both the analysis systems we will need the following Python libraries.

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 coefficent between two series.

```
import numpy as np
```

The ”pyplot” library it’s used for basic graphic displaying and customization, easy to use but very efficent.

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

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
```

4.3.2 Single Input Analyzer

- SIA part I: Generate and display araphic 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.

4.3.2.1 SIA section I: Total graphic for all the years

Goal:

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

Requirements:

- Input dataset: CSV format following the Stand_N

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

Code implementation:

During this section of the code we will use the ”pandas” library for read the dataset.

```
series = pd.read_csv("DATASET_DIRECTORY", header=0)
```

Then using the ”pyplot” library we can setup the plot of the input data.

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

Thera 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")
```

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

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

Results:

With this first part of the code we are able to display and save the basic graphic about the current input from 2005 to 2016, that looks like this example:

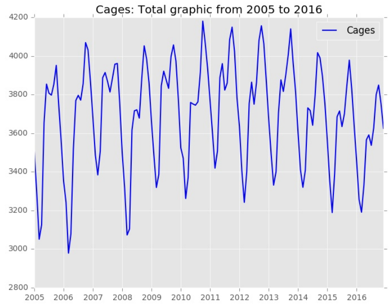

FIGURE 4.1: Total graphic about current input with total data from 2005 to 2016.

4.3.2.2 SIA section II: Single graphics for each year

Goal:

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

Requirements:

- Input dataset: CSV format following the Stand_M
- Data content: 144 values, 1 value for each month from 2005 to 2016

Code implementation:

During this section of the code we will use the ”pandas” library for read the dataset.

```
series2 = pd.read_csv("DATASET_DIRECTORY",
        index_col=['Month'],
        header=0, usecols=[0,1,2,3,4,5,6,7,8,9,10,11,12])
```

Then using the ”pyplot” library we can setup the plot of the input data.

```
series2.plot()
```

Adding the title at the graphic that we are going to display.

```
pyplot.title("Single year's graphic from 2005 to 2016")
```

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

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

Results:

With this second part of the code we are able to display and save the graphics about the current input for each single year from 2005 to 2016, that looks like this example:

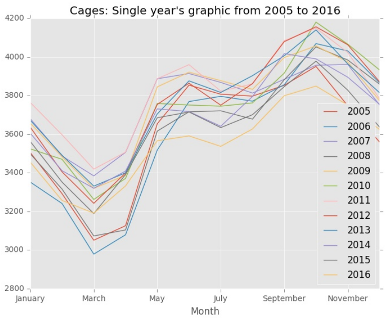

Figure 4.2: Graphics for each single year of the input data from 2005 to 2016

4.3.2.3 SIA section III: Correlation matrix between years

Goal:

Generate and display the correlation matrix about current input between each single year from 2005 to 2016

Requirements:

- Input dataset: CSV format following the Stand_Y
- Data content: 144 values, 1 value for each month from 2005 to 2016

Code implementation:

During this section of the code we will use the ”pandas” library for read the dataset.

```
series3 = pd.read_csv("DATASET_DIRECTORY",
         header=0, usecols=[1,2,3,4,5,6,7,8,9,10,11,12])
```

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

```
test = np.corrcoef(series3.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 coefficents.

```
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

```
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)
```

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 years")
pyplot.colorbar(cax)
```

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

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

Results:

With this second part of the code we are able to display and save the correlation matrix about current input between each single year from 2005 to 2016, that looks like this example:

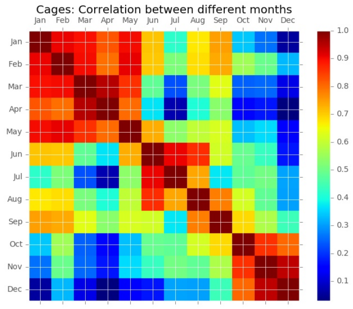

Figure 4.3: Correlation matrix between different months of the same input

4.3.2.4 SIA section IV: Correlation matrix between months

Goal:

Correlation matrix about current input between each single month from 2005 to 2016

Requirements:

- Input dataset: CSV format following the Stand_M

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

Code implementation:

During this section of the code we will use the "pandas" library for read the dataset.

```
series4 = pd.read_csv("DATASET_DIRECTORY", header=0,
        usecols=[1,2,3,4,5,6,7,8,9,10,11,12])
```

With the library "numpy" is possible to calculate the correlation coefficents 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 coefficents.

```
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.

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

Results:

With this second part of the code we are able to display and save the correlation matrix about current input between each single month from 2005 to 2016, that looks like this example:

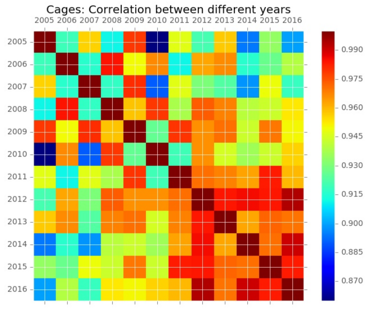

FIGURE 4.4: Correlation matrix between different years of the same input

4.3.3 Multiple Inputs Analyzer

Goal:

This analyzer is mainly used for show the correlation coefficent 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

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 coefficents 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 coefficents.

```
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")
pyplot.show()
```

Results:

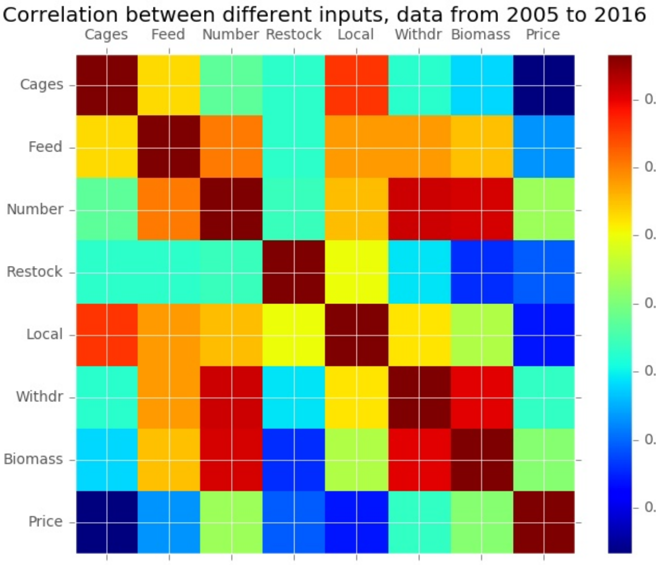

FIGURE 4.5: Correlation matrix between different inputs with data from 2005 to 2016

4.4 4th Phase: Data displaying

Part II

Experiment 2: Values prediction system

4.5 3th Phase: Data prediction system implementation

4.6 4th Phase: Prediction results displaying

Chapter 5

Results and Discussion

Chapter 6

Conclusion

Appendix A

An Appendix