

Project Report
Design of Forecasting Engine
Course Project in System Lab (IT 680)

Kaushal Mittal
(04329024)

Abhishek Seth
(04329001)

Amar Agrawal
(04329017)

April 19, 2005

Abstract

Most of the research work in forecasting domain concentrates on obtaining forecasts using multiple forecasting models rather than trusting a single model. In this project, we have designed a generic forecasting engine that can use a number of forecasting models provided by different forecasting systems and analyse the results for various research goals. Our experiments using this engine has revealed that a single forecasting model may not give the best predictions throughout the time series. A time series can be partitioned and different forecast models can be fitted for each of this segments to get the best forecast.

Contents

1	INTRODUCTION	3
2	MOTIVATION	3
3	REQUIREMENTS SPECIFICATION	4
4	DESIGN	5
4.1	Design Issues	5
4.2	Architechture	5
5	IMPLEMENTATION	7
5.1	Platform, Language and Tools	7
5.2	Assumptions and Dependencies	7
5.3	Deliverable	7
6	USAGE	8
7	EXPERIMENTAL RESULTS	9
8	FUTURE ENHANCEMENTS	10
9	APPENDIX	11
9.1	User Manual	11
9.1.1	Command Line Interface	11
9.1.2	Java Program Interface	14
9.1.3	Adding a new System	16
9.1.4	Accomodating new output format	18
10	Readings and References	19

1 INTRODUCTION

A time series can be defined as a sequence of observation taken sequentially in time[?]. The interval of observation is usually fixed. These observation can be monthly sales in a super market, hourly electricity load at power generators, and many more. What makes the time series interesting to analyze is the dependency of adjacent observations. This dependency of adjacent observations can be trivial or it can be complicated. Many real world time series has such complicated dependency, hence require sophisticated mathematics to explain them. Time Series analysis is concerred with fitting of mathematical models such as stochastic and dynamic models, to explain dependencies in observations.

The analysis of time series gives us the power of forecasting. Forecasting is concerned with the prediction of the future values of a given time series. Forecasting has many real world applications such as predicting of future sales, which will aid the businessmen in their planning.

Trivially one could think of using a forecasting expert that results in minimum error over a given time series. However, opinions from multiple experts can increase the confidence in the forecast values. A single expert cannot perform optimally over the entire series. Knowledge about which expert performs best at some segment of a time series can be used to improve the forecasting accuracy. We have designed an application that can be used to obtain this knowledge.

2 MOTIVATION

A number of systems like SAS, MATLAB etc. are available that can be used for forecasting. Each of these systems provides implementations for various kind of forecasting models. For a given series, using a combination of a number of experts over these systems can certainly improve the confidence in the forecast values.

We need a system, that allows to know the top-k experts for a given series, irrespective of the system that provides it. One of the ways for combining the results could be getting the votes from these top-k experts for a series. Another interesting enhancement possible is to get the best experts for different parts of the series. This will help us to know for what characteristics of a series, which expert will give better results. So the application should get forecast from different experts supported by various systems and find the best experts at each instance of time.

With the above aim, we have developed a prototype/design/implementation that combines these software systems in some useful way so as to predict the best forecast. This application is useful for various research goals in the forecasting domain, and practioners dealing with forecasting of time series.

3 REQUIREMENTS SPECIFICATION

The project was to develop a forecast engine that interacts with multiple forecasting back-ends such as SAS, MATLAB and other forecasting applications, to generate useful forecasts for a given time series. The main objective of the project was to provide support for analysing the forecast results from an ensemble of experts supported by different systems. Following requirements were identified for this project

- Support Multiple heterogenous forecasting backends for forecasting.
- Provide the support for analysis of results from an ensemble of forecasting experts to obtain -
 1. the top 'k' best experts for the entire series using an error criteria (eg. MAPE, RMSE, etc.) and use them to generate forecasts.
 2. the top 'k' best experts for each observation point in the series. This information can be used to identify if some experts perform better on some segment of the series which could help in selecting the best expert to use for each future forecast.
- Support adaptive forecasting i.e. generate forecasts on the series considering only the past values upto that point and not look into future values.

For example: If the series contains say 100 observation points then, use the first 30 points to predict the 31st, then include the 31st point(actual value) in the model to predict the 32nd next point and so on. This avoids the bias that occurs when we use all the points to build a model and then use it to calculate prediction errors on the series points.
- The application must facilitate the easy integration of new forecasting backends in the future. A developer must be able to do this with minimal efforts and no changes to the existing software.

4 DESIGN

4.1 Design Issues

Different forecasting backend systems differ in the format of input and outputs. They have their own scripting language. Few of them do not support adaptive forecasting directly. To provide integration support for such a variety of forecasting systems, following design decisions were considered.

- The backend system must have command line support.
- They must take the input time series in a flat file and should be able to generate the output forecast in a file.
- By default the system generate the reports containing the top-k models for the whole series and the top-k models for each observation. It uses the experts to forecast the value for each time instance and then calculate the error using the actual values for that time instance. Based on the error measure, it rank the experts.
- The template file is provided for defining the syntax and semantics of the scripts corresponding to the forecasting backend system. The template is an XML file with the simple programming features like parameter substitutions, looping. The system can generate the scripts for forecasting systems automatically using this file and the input parameters.

4.2 Architecture

The architecture of this application has following components:

- **Forecasting Engine:** This component takes in the input parameters and generate forecasts for a given time series. It act as a controller and coordinates among other components.
- **Script Generator:** This component generates forecasting backend specific scripts for the experts specified in the list of experts. This template file and the input parameters are used to generate the required script. The use of a template file makes the Script generator generic.
- **ScriptExecuter:** It interacts with the forecasting backends by executing the script files generated, using the appropriate command specified in the backend-definition file. Since the command can be different

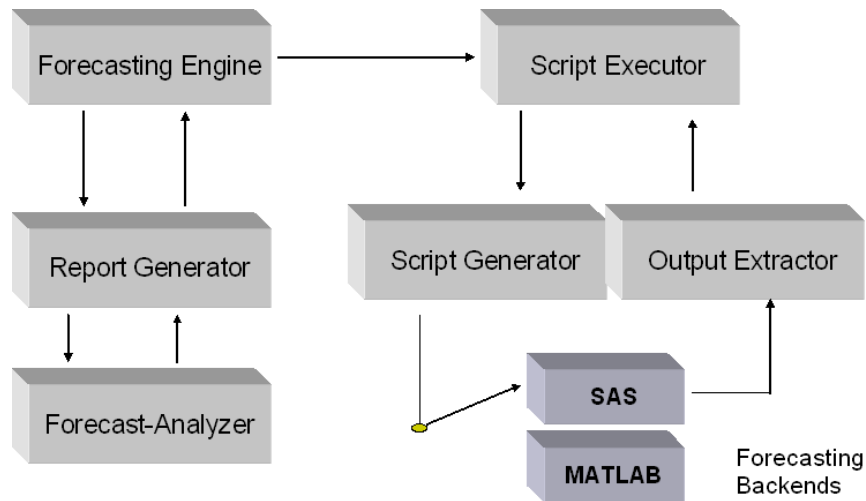


Figure 1: Architecture of Forecast Engine

for different forecasting backend systems, a new ScriptExecutor is required for different forecasting system.

- **Output Extractor:** This component extracts the results from the output file of forecasting backend. Since the forecasting backend systems have different output formats, a separate output extractor is required for each forecasting system.
- **Report Generator:** This component is responsible for presenting the statistics of various forecasting experts as a report in text and HTML format.
- **Forecast Analyzer:** This component is responsible for analyzing the forecasts obtained from various experts. The analysis consist of finding out the top-k expert on the overall time series, and finding out the top k experts for each observation along with various error measures.

5 IMPLEMENTATION

5.1 Platform, Language and Tools

1. Tested on windows.
2. Java used as implementation language.
3. Tools used are SAS, MATLAB for forecasts.

5.2 Assumptions and Dependencies

Currently the system runs on a single machine and assumes that the required forecasting software is installed on the same machine. The user is expected to make the required changes in the system properties file with path information.

5.3 Deliverable

The software is packaged as a JAR file and can be used directly from the command line or through JAVA programs.

6 USAGE

Following subsections provide an overlook of the basic usage of the software. A detailed user guide is provided in the appendix. The application can be used to:

- Generate the forecast using an expert.
- Generate the forecast with the best-k experts.
- Find the best-k experts at each time instance, for analysis purpose.
- Analyse the results from experts in his own way by using the API support.

A new forecast system can be added to the system through the following steps:

- Append a node in the backend-definition XML file. This contains information about the paths of scripts generated, output/log files, command to be executed and extensions of the script files.
- Write a general script to be executed on the new forecasting backend. Write a template XML file corresponding to this script.
- Write an expert definition file. This file contains the values of the parameters to be substituted in the template file for generating scripts.
- Make an entry of this new expert in the list of experts.
- If the output format is different from default format, write an output extractor and a customized engine. ¹.

¹Refer User Manual for details

7 EXPERIMENTAL RESULTS

In this section, we provide a summary of the results obtained on the seven different time series provided to us ². The series provides the monthly data. Table 7 gives the experts that appear in the results. This list is not exhaustive. The experiments were conducted using 73 different experts.

Table 7 gives the best model over the entire series and for the first three predictions i.e. for the forecast of observation 31, 32 and 33. The results shows that the best model found based on MAPE error criteria, for the overall series are different from the best models for individual observations. This, suggest for using different forecasting models for different part of the series.

Table 7 gives the MAPE over the entire series and APE error for the first three predictions i.e. for the forecast of observation 31, 32 and 33. The results shows that, accuracy would have increased tremendously, if the best model is used for prediction at each and every point.

Abbreviation	Full Form
E1	arima (0,2,2)(0,1,1)s
E2	arima (0,1,2)(0,1,1)s
E3	arima (2,0,0)(1,0,0)s
E4	arima (0,2,2)(0,1,1)s NOINT
E5	arima (0,1,2)(0,1,1)s NOINT
E6	arima (2,0,0)(1,0,0)s NOINT
E7	log arima (0,1,1)s
E8	log arima (0,2,2)(0,1,1)s NOINT
E9	log arima (1,1,1)
E10	log arima (2,0,0)(1,0,0)s
E11	log arima (2,0,0)(1,0,0)s NOINT
E12	log arima (2,1,0)(0,1,1)s NOINT
E13	Linear Trend AR4
E14	Linear Exponential
E15	log Linear Exponential
E16	log Winters Additive

Table 1: List of Experts

²Source: ABS

Series	BestModel	Obs31	Obs32	Obs33
1	E7	E7	E1	E16
2	E4	E4	E9	E4
3	E7	E5	E10	E6
4	E12	E2	E9	E13
5	E7	E14	E7	E7
6	E1	E3	E13	E8
7	E7	E15	E15	E11

Table 2: Best Models

Series	Best	Obs31	Obs32	Obs33	Optimal
1	3.21	1.26	0.36	1.68	0.45
2	8.62	0.35	2.49	0.57	2.32
3	7.72	2.35	0.19	1.15	1.76
4	7.87	1.66	3.25	0.60	1.55
5	5.34	0.40	0.78	0.02	1.10
6	9.51	0.19	0.04	0.01	2.08
7	14.72	0.58	0.12	3.95	2.27

Table 3: MAPE Errors

8 FUTURE ENHANCEMENTS

1. **For Developer:** The application currently uses a single execution thread on a single machine. A distributed parallel implementation might be supported in order to reduce the time for forecasts for a single series. This requires the scheduling of experts on multiple machines and combining their results.
2. **For User:** The implementation provides an extensive set of experts for SAS which can be used directly by the user. The forecasting system can be made more accurate/robust by incorporating additional forecasting experts into it. The application facilitate such extension through use of templates.

9 APPENDIX

9.1 User Manual

The user can use the forecasting package in two ways:

- Use command line to give a list of experts (property file) and general arguments (defined later).
- User writes his own java program using our class API's to interact with backend systems.

9.1.1 Command Line Interface

The first task is to update system properties file provided with the package, based on your requirement. An example of system properties file is follows

```
# System Options
BASEDIR                = C:/kaushal/sastest
ApplicationName        = STATS
LogFileNames           = /tests/Applog.log
SASApplication         = "c:/Program Files/SAS Institute/SAS/v8/sas.exe"
SASdatafiledir         = "./sastest/work/sas/datafiles"
SASscriptfiledir      = "./sastest/work/sas/scriptfiles"
inputfile              = "./sastest/tests/Testdata.txt"
SASForecastOutFile    = "./sastest/tests/forecastout.txt"
SASForecastLogFile    = "./sastest/tests/forecastlog.txt"
```

The command used to run the forecasting engine is shown below. ³. After running this command, system generates two reports output.best and output.opt. The specific examples are given below

Command:

```
java stats.metric.ForecastEngine -i <INPUT-DATA FILE>
-p <EXPERT-FILE> -r <OUTPUT-REPORT-FILE>
```

³ Assuming the stats.jar is in classpath of java

```
*****
An example of input data file
*****
```

```
23
45
56
68
```

```
*****
An example of expert file
*****
```

```
Expert1=c:/forecast_work/experts/sas/arima/arima/arima011s.exp
Expert2=c:/forecast_work/experts/sas/arima/arima/arima_111.exp
Expert3=c:/forecast_work/experts/sas/arima/arima/arima011_100s.exp
Expert4=c:/forecast_work/experts/sas/arima/arima/arima012_100s.exp
Expert5=c:/forecast_work/experts/sas/forecast/add_winters.exp
Expert6=c:/forecast_work/experts/sas/forecast/linear_expo.exp
Expert7=c:/forecast_work/experts/sas/forecast/linear_trend.exp
Expert8=c:/forecast_work/experts/sas/forecast/linear_trend_AR1.exp
Expert9=c:/forecast_work/experts/sas/forecast/linear_trend_AR2.exp
```

```
*****
Specific expert file c:/forecast_work/experts/sas/arima/arima/arima011s.exp
*****
```

```
# Expert 1
```

```
EXPERTNAME      = ARIMA(0 1 1 )s
```

```
# parameters to change
```

```
method          = ML
```

```
# parameters defined for the above expert
```

```
APPLICATION     = SAS
```

```
TEMPLATE       = ARIMA
```

```
LOG             =
```

```
difference     = (12)
```

```
nlag           = 24
```

```
p              = (0)
```

```
q              = (12)
```

```

lead          = 1
interval      = 1

```

```

*****
Report Generated report.opt
*****

```

TOP 3 Forecast values Using 3 best model at each Observation.

Summary

```

SeriesError      ModelName
2.2663234907377743  Optimum 1
2.8724923891518594  Optimum 2
3.470143675690086   Optimum 3

```

ObsId	Actual	Forecast	ErrorValue	ModelName
1	117.0	117.68	0.581196581196587	LOG_LINEAR_EXPO
	117.0	115.691	1.1188034188034166	LOG_WINTERS_ADDITIVE
	117.0	115.691	1.1188034188034166	LOG_WINTERS_METHOD
2	117.0	117.142	0.12136752136751787	LOG_LINEAR_EXPO
	117.0	117.33	0.2820512820512806	LOG ARIMA(0 1 2)(0 1 1)s
	117.0	115.951	0.8965811965812022	LOG_WINTERS_ADDITIVE
3	106.0	110.189	3.9518867924528234	LOG ARIMA(2 0 0)(10 0)s NOINT
	106.0	111.944	5.6075471698113235	ARIMA(0 1 2)(0 1 1)s
	106.0	113.066	6.666037735849059	LOG ARIMA(2 1 2)(0 1 1)s

```

*****
report.best
*****

```

TOP 3 Forecast series Using 3 best model.

Summary

```

SeriesError      ModelName
14.711904164062087  LOG ARIMA(0 1 1 )s
14.711904164062087  LOG ARIMA(0 2 2)(0 1 1)s
14.711904164062087  LOG ARIMA(0 1 1 )(1 0 0 )s

```

ObsId	Actual	Forecast	Error	ModelName
1	117.0	151.698	29.65641025641026	LOG ARIMA(0 1 1)s
	117.0	151.698	29.65641025641026	LOG ARIMA(0 2 2)(0 1 1)s
	117.0	151.698	29.65641025641026	LOG ARIMA(0 1 1)(10 0)s
2	117.0	153.981	31.607692307692304	LOG ARIMA(0 1 1)s
	117.0	153.981	31.607692307692304	LOG ARIMA(0 2 2)(0 1 1)s
	117.0	153.981	31.607692307692304	LOG ARIMA(0 1 1)(10 0)s
3	106.0	144.064	35.909433962264146	LOG ARIMA(0 1 1)s
	106.0	144.064	35.909433962264146	LOG ARIMA(0 2 2)(0 1 1)s
	106.0	144.064	35.909433962264146	LOG ARIMA(0 1 1)(10 0)s

The following command line options are supported by forecasting engine:

- -I Input file (data series)
- -p Property file (list of experts to be used)
- -r Report file
- -t Number of best experts in report. (3)
- -s Forecasting starts from this observation (31)
- -f Forecast upto this observation. (31)
- -a Flag actual values displayed or not. ;enabled;

9.1.2 Java Program Interface

The forecasting engine can also be used from custom Java program. An example test java program is shown below:

```
public class Test {

public Test(){

}

}
```

```

public static void main(String[] args) {
try{
ForecastEngine engine = new ForecastEngine();

Properties prop = new Properties();
//LOAD EXPERT FILE
prop.load(new FileInputStream("/forecast/add_winters.exp"));
HashMap inputParameters = new HashMap(prop);
inputParameters.put(ParameterConstants.BEGIN,"31");
inputParameters.put(ParameterConstants.END,"32");
inputParameters.put(ParameterConstants.INCR,"1");
inputParameters.put(ParameterConstants.INPUTFILE,
"C:/kaushal/sastest/tests/Testdata.txt");
engine.setInputParameter(inputParameters);
engine.execute();
ForecastAnalyzer analyser = engine.getForecastAnalyzer();

ForecastAnalyzer analyser = engine.getForecastAnalyzer();
ModelMetrics output[] = analyser.getOptimumForcast(ErrorMeasures.APE, 2);
for(int i = 0 ;i< output.length ; i++){
double [] forecastvalues = output[i].getForecastSeries();
System.out.println(forecastvalues.length);
for (int j = 0; j < forecastvalues.length ; j++)
System.out.println(forecastvalues[j]);
}
//Second method
ReportFormat format = new ReportFormat();
format.setFormat(ReportFormat.FORMAT_BESTK);
AbstractReportGenerator gen = engine.getDefaultReportGenerator();
gen.setReportFile("c:/Myoutput.txt");
gen.generateReport(format);
}
catch(Exception e){
System.out.println("Some error occured");
}
}
}

```

9.1.3 Adding a new System

Following steps are used to add a new system

- Write a XML template file for the scripts for the system.
- Modify the module-definition.xml to add the node for the new template file.
- Write a parameter file to define the forecasting expert.
- Add the entry for the expertfile in the list of experts property file.
- If the output format is different from forecasting system already supported. Create a new OutputExtractor implementing the interface provided.
- Write a custom forecast Engine.

An example of adding a forecasting backend, written in java application is presented below:

Forecasting backend to be added: TestForecast.class

1. Write a Usage script: For any forecasting backend, you need to write a usage script . Make the input/output files and other parameters hardcoded. The script program should take input file and produce the forecasting results in output file in preferably simple format.

An example of such script is as follows:

```
public class EXPERT100 {
public static void main(String[] args)
{
String INFILE = "/forecast_Work/preproc/abraham.txt";

String OUTFILE="/forecast_Work/testbackend/outputs/EXPERT100.out";

int START = 31 ;
int END = 40 ;
try{
TestForecast forecast = new TestForecast(INFILE ,
OUTFILE , START , END);
```

```

forecast.doForecast();
}catch (Exception e)
{
System.out.println(e.getMessage());
}
}
}
}

```

2. Write the backend XML file: Replace all parameters of the Usage script by variables. The XML file for the example Usage script is as follows:

```

<MODULE NAME="TESTFORCAST">
  <INPUTFILE>/home/abhishek/courses/syslab/test/abraham.txt</INPUTFILE>
  <SECTION NAME="PROGRAM">
    <STMT>public class #($(SCRIPTFILENAME)) {</STMT>
    <STMT> public static void main(String[] args) { </STMT>
    <STMT> String INFILE = "#($(INPUTFILE))"; </STMT>
    <STMT> String OUTFILE = "#($(OUTPUTFILE))"; </STMT>
    <STMT> int START = #($(BEGIN)) ; int END = #($(END)) ; </STMT>
    <STMT> try{ TestForecast forecast =
new TestForecast(INFILE , OUTFILE , START , END);
                forecast.doForecast(); }catch (Exception e) { System.out.p
    </SECTION>
  </MODULE>

```

3. Add an application node in script XML file provided with this package. This application node defines various parameters.

```

script.xml
-----

```

```

..other application nodes

```

```

<APPLICATION NAME="TESTJAVA1" COMMAND="/testbackend/javatest.sh"

```

```

SCRIPTDIR="/testbackend/scripts/"
OUTPUTDIR="/testbackend/outputs/"
LOGDIR="/testbackend/logs/"
EXTENSION=".java"
>
<TEMPLATE NAME="TESTFORCAST">resource/templates/testforcast.xml</TEMPLATE>
</APPLICATION>

```

```

javatest.sh
-----
#!/bin/bash

javafile=$1;
newfile='expr match "$javafile" '\([^.*]*\)'';
basename='basename $javafile';
classfile='expr match "$basename" '\([^.*]*\)'';

CLASSPATH=/testjava/:$CLASSPATH

echo "$javafile    $classfile\n\n";
javac -classpath $CLASSPATH $javafile;
java -classpath $CLASSPATH $classfile;

```

4. Write an expert file corresponding to the new backend.

```

EXPERTNAME      = TESTJAVA1
# parameters to change

# parameters defined for the above expert
APPLICATION     = TESTJAVA1
TEMPLATE        = TESTFORCAST

```

9.1.4 Accomodating new output format

If the output format is not supported by the forecasting engine, you need to write a new Java class for output extractor. To use the new output extractor, you need to write your a new forecasting engine. But writing this new forecasting engine is easier as follows:

```
MyForecastEngine extends ForecastEngine{
ForecastEngine(args []) {
Super();
OutputExtractorFactory.add(MyoutputExtractor ob);
}
public static void main(){
ForecastingEngine = new forecastingEngine(args []);
    engine.execute();
}
```

The next step is to run the forecasting engine equipped with new backend. To use the new backend add the expert file entry in the list of expert files, which is passed as command line option.

10 Readings and References

- SAS User Manuals. ETS Package.
- Matlab user Manual. Neural Network toolbox.