Predistock: Applying ML principles to build retail stock prediction AIs

T. Hollis (August 29, 2017)

1. Introduction

The ComCleaver group is venturing into the world of Big Data, Machine Learning (ML) and Artificial Intelligence (AI) by the development of its Predistock project.

The Predistock project aims to harness the latest scientific research developments in ML to build a competent AI capable of predicting optimal stock levels using a web-collected dataset.

This AI would then be able to advise ComClever customers on various metrics which may help liquidity and prevent excessive assets tied up in stock. This would be an alternative to the Just-In-Time approach.

2. Requirements outline

The AI needs to be able to process a comma-separated value (CSV) datasets and return the relevant analysis in a human readable form.

The main algorithm must be written in a single piece of software code which can be deployed on any modern OS, platform independently, as long as the required software is installed.

The AI also needs to have a clear sequence of instructions for it to be reproducible for client back-end.

Finally a mock dataset must be produced using reasonable guesses to demonstrate and evaluate the performance of the proposed AI solution.

These instructions and all the relevant files are present in this report and in its attached appendices.

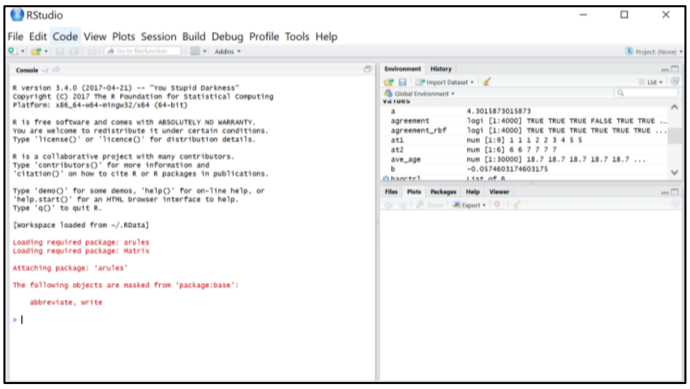
It is worth noting that the entire AI must be developed using a mock dataset as a Proof-Of-Concept as the data will only be collected once the feasibility of this AI has been demonstrated. This dataset must also be created using reasonable values.

3. Choosing the language & IDE

The first step in the design process is to decide on a language to write the AI in. The R language[1] offers a wide range of advantages over other programming languages. The most important benefits include a remarkable library of pre-build packages (CRAN[2]), an easy to use and simple syntax and a heavy connection to scientific literature.

In fact, many modern-day AIs are written in R simply because of this final advantage. As AI research is mostly done in R and documented as such, it makes intuitive sense to build on the work of others rather than rewriting certain fundamental functions from scratch.

The favoured R integrated development environment (IDE) of the software developer of this project is RStudio.



```
R version 3.4.0 (2017-04-21) -- "You Stupid Darkness"
Copyright (C) 2017 The R Foundation for Statistical Computing
Platform: x86_64-w64-mingw32/x64 (64-bit)

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

[Workspace loaded from ~/.RData]

Loading required package: arules
Loading required package: Matrix

Attaching package: 'arules'

The following objects are masked from 'package:base':

    abbreviate, write

> |
```

```
a                4.3015873015873
agreement        logi [1:4000] TRUE TRUE TRUE FALSE TRUE TRUE ...
agreement_rbf    logi [1:4000] TRUE TRUE TRUE TRUE TRUE TRUE ...
at1              num [1:9] 1 1 1 2 2 3 4 5 5
at2              num [1:6] 6 6 7 7 7 7
ave_age          num [1:30000] 18.7 18.7 18.7 18.7 18.7 ...
b                -0.0574603174603175
bagctrl          List of 6
```

Figure 3.1 - RStudio IDE

RStudio is versatile yet simple enough for the scope of this project. However, it is worth noting that the same AI can also be built using the default R console.

4. Choosing the ML algorithm

In order to make the best choice regarding the algorithm to be used for the Predistock project, the first aspect to consider is the desired output of the AI.

In this case, the desired output is to predict the optimal stock level. This is a quantitative output.

This means we must chose a predictive ML model, based on classification, rather than a descriptive ML model, based on regression.

Another criterion to choose is whether or not this learner should be supervised or unsupervised. In our case our learner must be supervised as we can provide it with inputs and build a cost function for the outputs, helping the algorithm learn from past mistakes. In addition, the algorithm's output can be easily verified and corrected, thus supervised learning is chosen.

The other ML algorithm options considered are summarised on the following page.

[1] The R Foundation, *What is R?* (2017) Available at: https://www.r-project.org/about.html

[2] The R Foundation, *Comprehensive R Archive Network* (2017) Avaible at: https://cran.r-project.org/

Model	Learning task
Supervised learners	
k-Nearest-Neighbour	Classification
Naïve Bayes	Classification
Decision Trees	Classification
Classification Rule Learners	Classification
Linear Regression	Numeric prediction
Regression Trees	Numeric prediction
Model Trees	Numeric prediction
Neural Networks	Both
Support Vector Machines	Both
Unsupervised learners	
Association Rules	Pattern detection
k-Means Clustering	Clustering

Table 4.1 - ML algorithms considered

At this stage Black Box Method algorithms (neural networks and support vector machines) stand out as being the most well suited to the Predistock AI, with neural networks having a slight advantage over support vector machines.

Neural networks have been around since the 1990s and were kickstarted with I-Cheng Yeh's pioneering work. However, processing resources and artificial intelligence research has only recently become advanced enough to put the theory into practice. The modern R package "*neuralnet*" will be the backbone of our Predistock AI and was first published in 2016 based on many scientific research papers from previous years.

The main strengths and weaknesses of this algorithm are summarised below.

Strengths	Weaknesses
• Can be adapted to classification or numeric prediction problems	• Extremely computationally intensive and slow to train, particularly if the network topology is complex
• Capable of modelling more complex patterns than nearly any algorithm	• Very prone to overfitting training data
• Makes few assumptions about the data's underlying relationships	• Results in a complex black box model that is difficult, if not impossible, to interpret

Table 4.2 - Strengths and weaknesses of neural networks

Neural networks use an electronic replica of a neuron called a perceptron to make decisions based on differentially weighted inputs as part of a wider decision-making network. Once the algorithm is trained with training data it automatically adjusts its weights.

Many different networks can be built with a varying number of nodes, layers, topology and feedback but a simple feed-forward neural network will be used for this Proof-Of-Concept version of the Predistock AI.

5. Creating the dataset

As data was not yet available to train and develop the AI, mock data had to be made up.

This mock data was done by combining random number generation, extrapolation and relationship guessing. The following feature data was generated using the following methods:

Actual order quantity - extrapolated (int)
Mean rating - randomly generated (float)
Pre-orders - randomly generated (int)
Page visits - arithmetic mean of ratings and pre-orders (int)
Total visits - extrapolation with random noise (int)
Time spent on page - geometric scaled mean (float @1dp to allow for ping uncertainty)
Total time spent - noisy extrapolation with inferred relation (float @1dp to allow for ping uncertainty)
Weather - dummy-coded & randomly generated (int)
Current sale status - dummy-coded & biased random generation (int)
Competition evolution - normalised & randomly generated (float)
Number of tools used - normalised & randomly generated (float)
RDB - normalised & randomly generated (float)
RDA - normalised & randomly generated (float)
IPC - normalised & randomly generated (float)
PAEC - normalised & randomly generated (float)
Inflation - extrapolation with random noise based on typical CPI inflation rate of France in 2017. (float)
Political/social tendency - normalised & randomly generated (float)

Exact equations for each feature can be found in the attached file "Predistock AI - Dataset.xlsx".

The full data can be seen in CSV format in the "Predistock AI - Dataset.csv" file also attached to this report.

6. Building the AI

The first step to building the AI is to import the CSV data into RStudio in the form of a data frame.

This data frame must then be normalised across all features.

Then the data must be split into two sections: the training dataset and the testing dataset. This split should be roughly 3:1.

The algorithm is then trained using the training data which finishes the build of the full AI.

The full code used to build & test the AI can be found in the files "Predistock AI - Model.RData" and "Predistock AI - Model.Rhistory". This is also outlined in Appendix 1.

A first neural network is built using only a single hidden node in a three-layer feed-forward topology. This produces the following neural network:

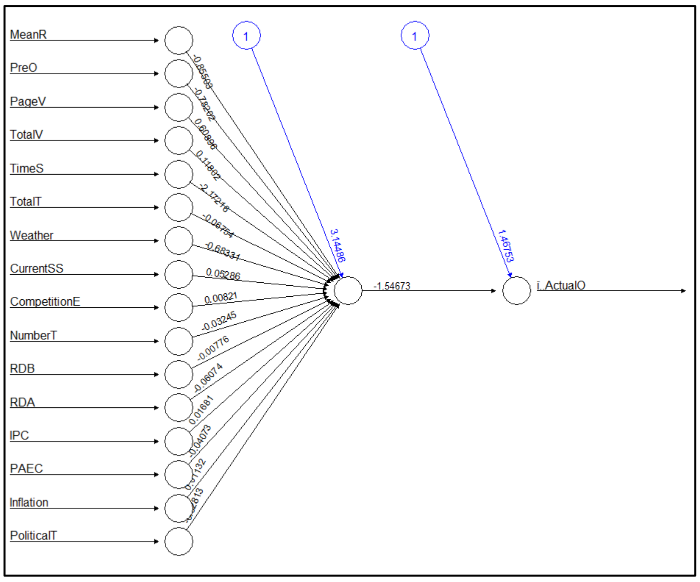


Figure 6.1 - Single hidden node three-layer feed-forward neural network Predistock AI

A second, more complex neural network is also built using a 5-hidden-node three-layer feed-forward topology. This produces the following neural network:

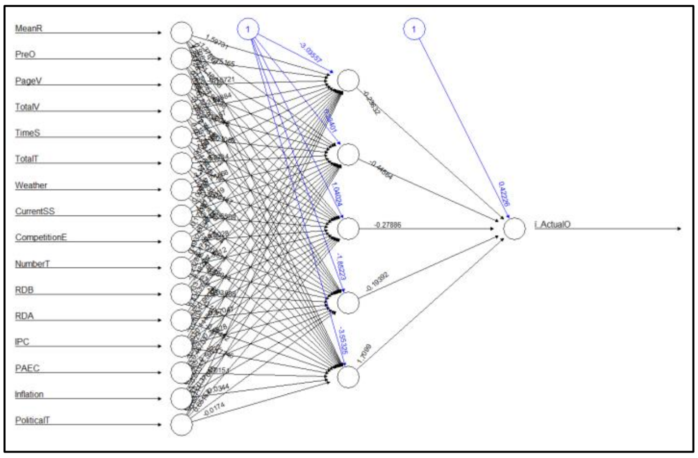


Figure 6.2 - 5-hidden-node three-layer feed-forward neural network Predistock AI

7. Evaluating AI performance

Now that the AI has been fully trained, we can evaluate and refine its performance on the testing data to see how well it makes stock predictions. The testing undertaken is also outlined in Appendix 1.

By asking both the simple and complex neural networks to make predictions on the test data we can compare the predictions of stock levels with the actual required stock levels.

The simple neural network shown in figure 6.1 has an excellent predictive performance with a correlation of approximately 0.93.

The slightly more complex neural network shown in figure 6.2 has a slightly less good but still excellent predictive performance with a correlation of approximately 0.87. This is most likely due to overfitting the data and is an issue to be investigated along with many other future development tasks.

8. Future development

This AI can be further refined by evaluating its performance using ROC and AUC to visualise its precision and accuracy.

This will allow us to determine whether the neural network has the most optimal topology, number of layers, number of hidden nodes and other tuning parameters.

These tuning parameters be improved by creating a meta-learner. Meta-learners learn how to learn. Essentially a secondary AI can be built to fine tune the parameters of the main AI to improve its overall performance. Bagging, boosting and random forests may also be used to improve model performance.

Once this is done, we can also use a multi-algorithm AI (by also including Support Vector Machines) which uses either voting or averaging to take into account the advantages of each algorithm without their respective shortcomings. This is an extremely powerful tool to improve performance and the AI's confidence in its own predictions.

These improvements should however only be undertaken on a real dataset, as well as a few other minor tweaks that must be done when dealing with real data (using 10-fold cross validation, stratified random sampling…).

9. Conclusion

The Predistock AI prototype successfully predicts optimal stock levels using the mock data. It does so quite accurately although this does have the opportunity to be improved with the recommendations outlined in section 8.

This is extremely encouraging since even without any of the future development steps suggested, a fairly rudimentary single-hidden-node three-layer feed-forward neural network has allowed us to build a powerful AI for the Predistock project that gives invaluable insight on optimal stock levels at high correlation levels.

Appendix 1 - Complete R algorithm implementation

```
> predistock_data <- read.csv("Predistock AI - Dataset.csv")

> predistock_data_norm <- as.data.frame(lapply(predistock_data, normalize))

> predistock_train <- predistock_data_norm[1:300]

> predistock_train <- predistock_data_norm[1:300, ]

> predistock_test <- predistock_data_norm[301:400, ]

> library(neuralnet)

> predistock_model <- neuralnet(ï..ActualO ~ MeanR + PreO + PageV + TotalV + TimeS +
TotalT + Weather + CurrentSS + CompetitionE + NumberT + RDB + RDA + IPC + PAEC + Inflation
+ PoliticalT, data = predistock_train)

> predistock_model_results <- compute(predistock_model, predistock_test[1:16])

> predicted_stock <- predistock_model_results$net.result

> cor(predicted_stock, predistock_test$ï..ActualO)

> predistock_model2 <- neuralnet(ï..ActualO ~ MeanR + PreO + PageV + TotalV + TimeS +
TotalT + Weather + CurrentSS + CompetitionE + NumberT + RDB + RDA + IPC + PAEC + Inflation
+ PoliticalT, data = predistock_train, hidden = 5)

> predistock_model_results2 <- compute(predistock_model2, predistock_test[1:16])

> predicted_stock2 <- predistock_model_results2$net.result

> cor(predicted_stock2, predistock_test$ï..ActualO)
```