Internal Handover Documentation

Forecasting Competitor Media Spend

Authors
James Dborin
Ishraq Irteza
Ryan Sephton

Project Start
June 25, 2018

Version 0.1
August 31, 2018

List of acronyms

ARIMA	Autoregressive Integrated Moving Averages
GBRF	Gradient Boosting Regression Forest
MLPR	Multi-layered Perceptron Regression
NN	Neural Network
VAR	Vector Autoregression

MAKETEA MLPR ARIMA KNN ERRF TSF EN AWS

Contents

1 | Overview of Structure

This section will discuss the structure of the presented project. The model consists of two primary classes; `data` and `Predictor`, and three models – `ARIMA`, `VAR`, and `GBRF` – all child classes of `Predictor`. A basic workflow diagram of the project can be found in figure 1.1, though the reader is reminded that the processes have been greatly abridged for clarity. Details of the procedures within each of the model classes and how the data is partitioned is discussed within their respective sections.

The project has been completed with the aim of limiting user input where possible. The only obligatory user inputs are an `input path`, the path to the `.csv` file containing historic media spends, an `output path` to where predictions are stored and a model selection variable which chooses between ARIMA and GBRF for selected datasets. For future development, it would be beneficial to have an automatic model selector that is able to choose between ARIMA and GBRF dependent on their individual strengths.

Using the `input path`, an instance of the `data` class is created. Here, functions act to clean the data extracted from the `input path` for entry into the models and set a forecast length, the default being the number of months between the last data entry and the end of the next year. The data is then partitioned dependent on the completeness of the dataset. Within each partition, the data is grouped based on the similarity of historic spend. The `empty set` contains groups where over a third of all data entries are empty, with the rest of the data put into the `complete set`.

Partitioning data is important to allocate the appropriate models to each group. Furthermore, some models are multivariate, and thus require datasets that exhibit similar behaviour to compensate for the relatively small number of data points.

For groups in the `empty set`, instances of the `VAR` class are created. A dataset with so many consecutive zero entries generally indicates either that there is unlikely to be further activity from the company or very limited and completely unpredictable behaviour - thus allocating significant optimization and prediction time would be unnecessary. VAR was found to be the model that both requires the least optimization time and outputted the most sensible results for empty set groups.

For groups in the `complete set`, the model selection variable choose between the `GBRF` and `ARIMA` models. All models are child classes of the `Predictor` class, which contains a switch-case mechanism (triggered by the labelling of the two models) to use the correct functions for each model. Within `Predictor`, the function `forecast` triggers three functions within the selected model class; `process`, `optimize`, and `predict`.

Each of these functions creates a variable of the model class that is handed down to

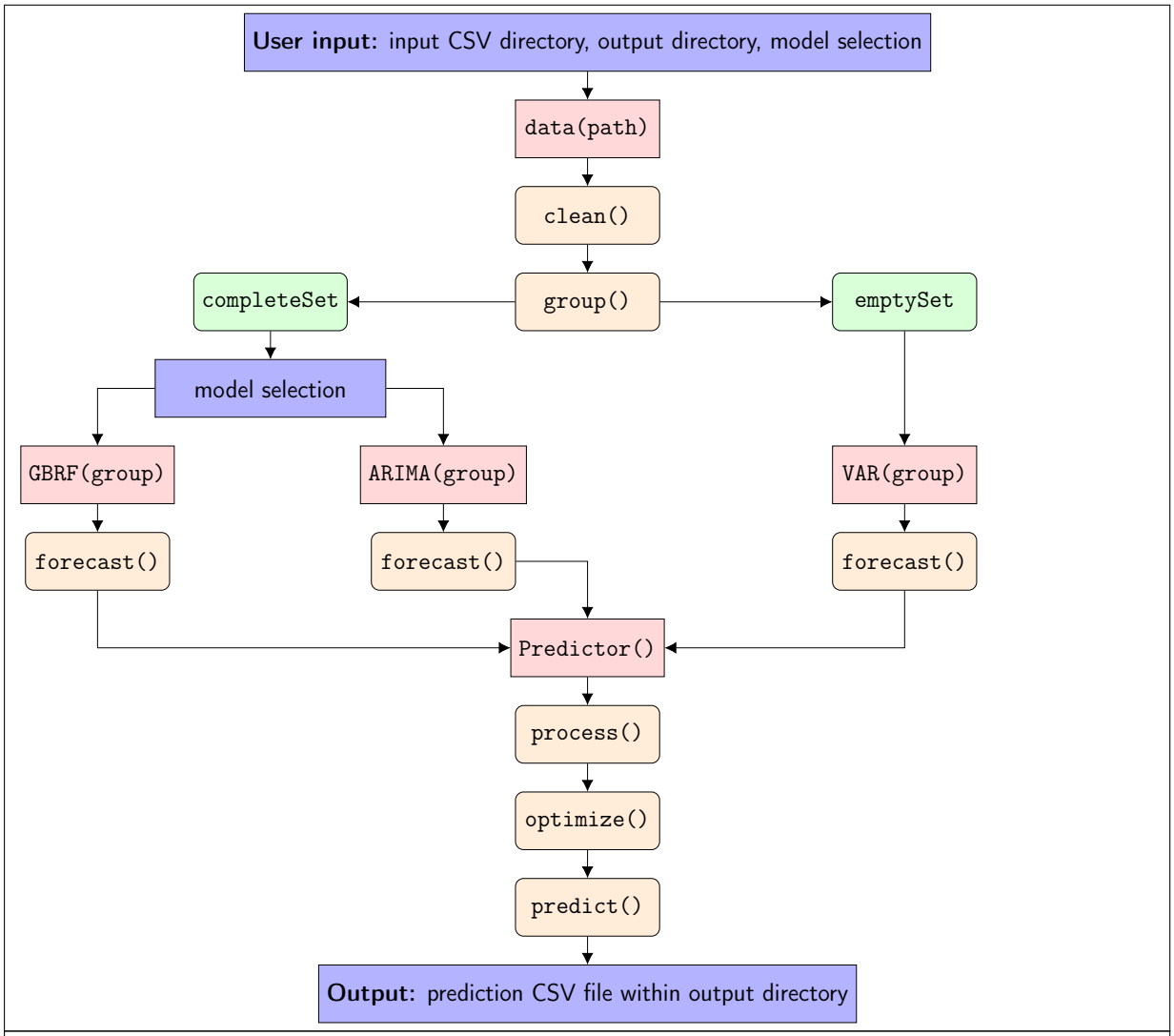

Figure 1.1: Workflow diagram of submitted model.

eventually create the forecast. `process` acts first to take in the cleaned data, and process it to the specifications of each model, the output vairable being `<model>.processed` , which is handed down to the `optimize` function. `optimize` reads in the processed data and runs tests on a validation set of the data to find the optimum parameters of the model, outputted as `<model>.parameters`. This variable is handed down to the `predict` function, which creates a model using the specified parameters and acts on the processed data to return `self.predictions`.

The subroutines within each of these functions are covered in their respected chapters.

2 | Data

2.1 Requirements on the data

Data must be useful and acceptable to the model in order to obtain the best predictions. For data to be useful, it must be filtered correctly from the Pivot tables. To be accepted into the model, this data then has to adhere to the formatting specified in section 2.1.2.

2.1.1 Filtering the Pivot table

During testing of the models, historic competitor spending data was taken from the Redmill Tuk-Tuk system. The goal with data filtration is to limit the number of zero entries whilst still accounting for seasonal trends. A balance was found through filtering for need states within countries, however this is just a suggested parameter, and one is free to experiment with which filtration variables give most useful data.

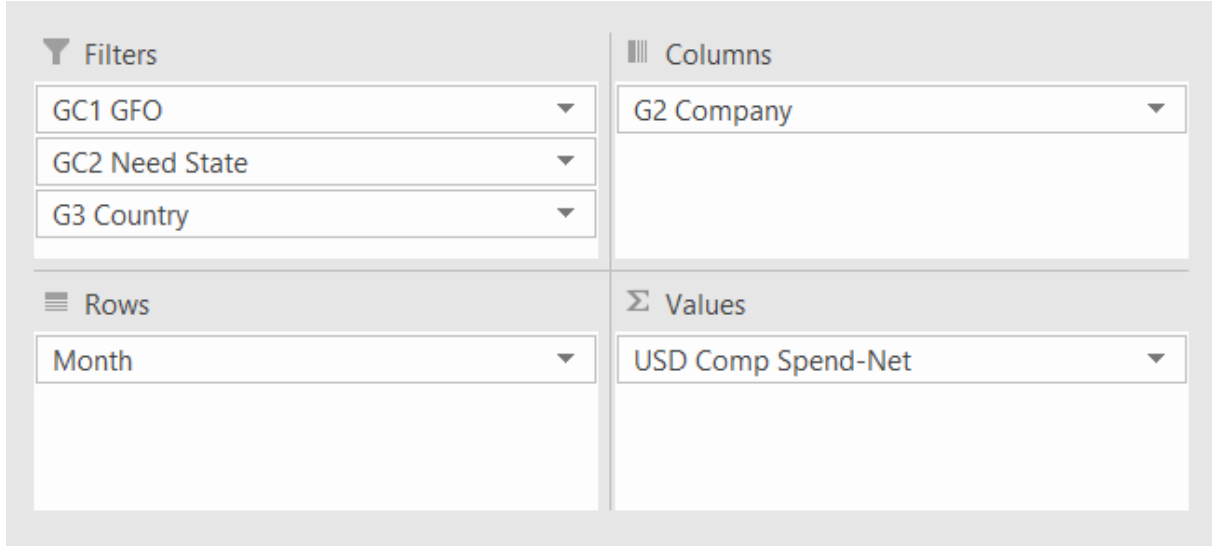

Figure 2.1: Example Pivot filters

The only mandatory filter requirments are that the `Rows` and `Values` areas within the Pivot are the same as in figure 2.1.

2.1.2 Microsoft Excel

	A	B	C	D	E	F
1	Row Labels	Company A	Company B	Company C	Company D	Company E
2	Jan-13	7726	5090	1536	293	9336
3	Feb-13	6126	8179	3293	6497	5901
4	Mar-13	7041	9532	9717	6036	2171
5	Apr-13	3219	5242	5255	1239	4173
6	May-13	7589	6653	4864	9615	2516
7	Jun-13	346	4492	3082	5068	8472
8	Jul-13	7984	3491	5793	8978	479
9	Aug-13	7682	1955	3101	1383	9516
10	Sep-13	97	8224	7346	5688	5597
11	Oct-13	2802	9361	4849	632	8844
12	Nov-13	2260	394	7022	6492	1258

Figure 2.2: Example `.csv` file

Once filtered, only the values are to be extracted from the Pivot table. One should make sure that the data type in MS Excel for all cells is `General`.

The layout of the `.csv` file should mimic figure 2.2, with flexibility on the format of the dates. Current accepted date formats are: `MMM-YYYY` (e.g. Jan-2013), `MMM-YY` (e.g. Jan-13), or `MMMYYYY` (e.g. Jan2013). More date formats can easily be added via the `clean` function within the `Data` class.

The final file type should be a `.csv`, which is then ready to be imported.

2.2 Data class

2.2.1 Grouping

Let X and Y be two sets of historical media spend values for companies A and B. To establish correlation, one can plot X against Y and calculate an estimation of the linearity of the resulting line using the Pearson correlation coefficient. Note that we are, at this version of the model, uninterested in higher order correlations between sets of data, as the multivariate models included require similarly correlated data to exaggerate the trends

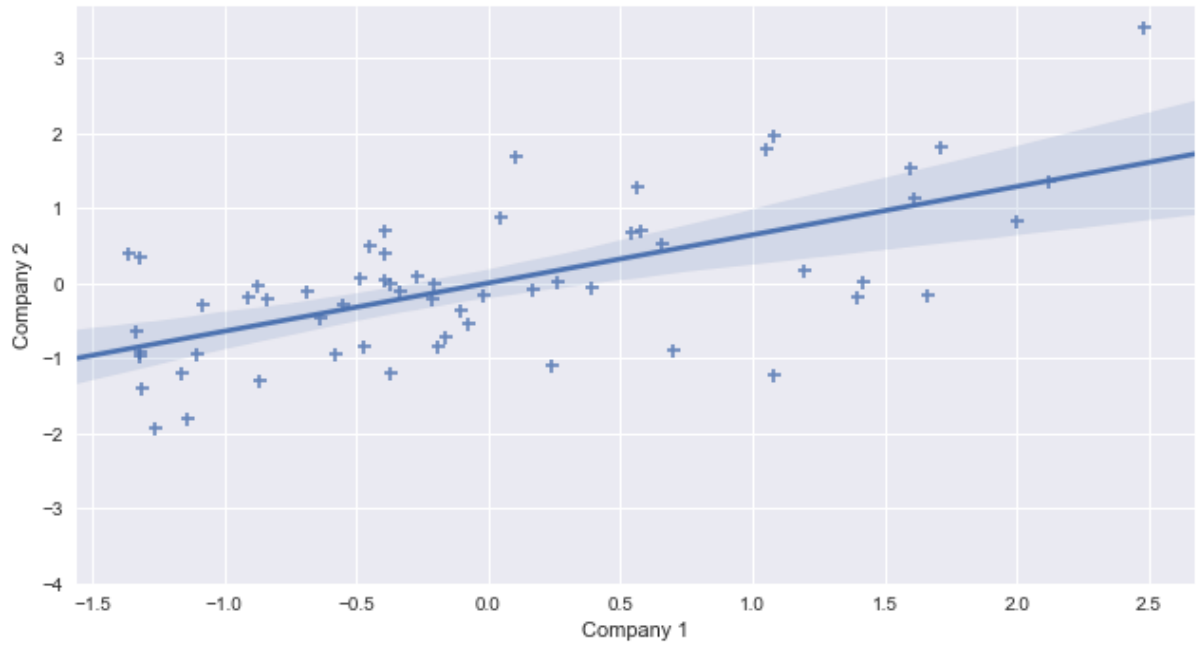

Figure 2.3: Calculating the Pearson correlation coefficient. Data transformed to a mean of 0.

within each datasets to compensate for the low number of data points in each individual set. Figure 2.3 shows an approximate linear line drawn through two sets of data.

The Pearson correlation coefficient between two datasets X and Y, $r_{X,Y}$, is defined as

$$r_{X,Y} = \frac{\text{covar}(X,Y)}{\sigma_X \sigma_Y} = \frac{\langle (x - \langle x \rangle)(y - \langle y \rangle) \rangle}{\sqrt{\langle (x - \langle x \rangle)^2 \rangle} \sqrt{\langle (y - \langle y \rangle)^2 \rangle}} \tag{2.1}$$

$r_{X,Y} = 1$ implies perfect correlation, $r_{X,Y} = 0$ implies no correlation, and $r_{X,Y} = -1$ implies perfect negative correlation.

The Pearson correlation coefficient is an indication of the strength of the correlation, but gives no information regarding the significance. To distinguish between strength and significance, consider two datasets A and B with with only one element inside. Now if the element within A and B is the same, the strength of the correlation, $r_{A,B}$, is 1. But now if we consider two datasets C and D both containing the same 10 elements, $r_{C,D}$ is again 1. However, if we assume the data is distributed identically for A, B, C and D, the significance of 10 identical elements between sets C and D is much greater than 1 between A and B. This point is later shown in figure 2.4.

Let us assume that the line between two datasets with N data points has form

$$y_i = \alpha + \beta x_i \tag{2.2}$$

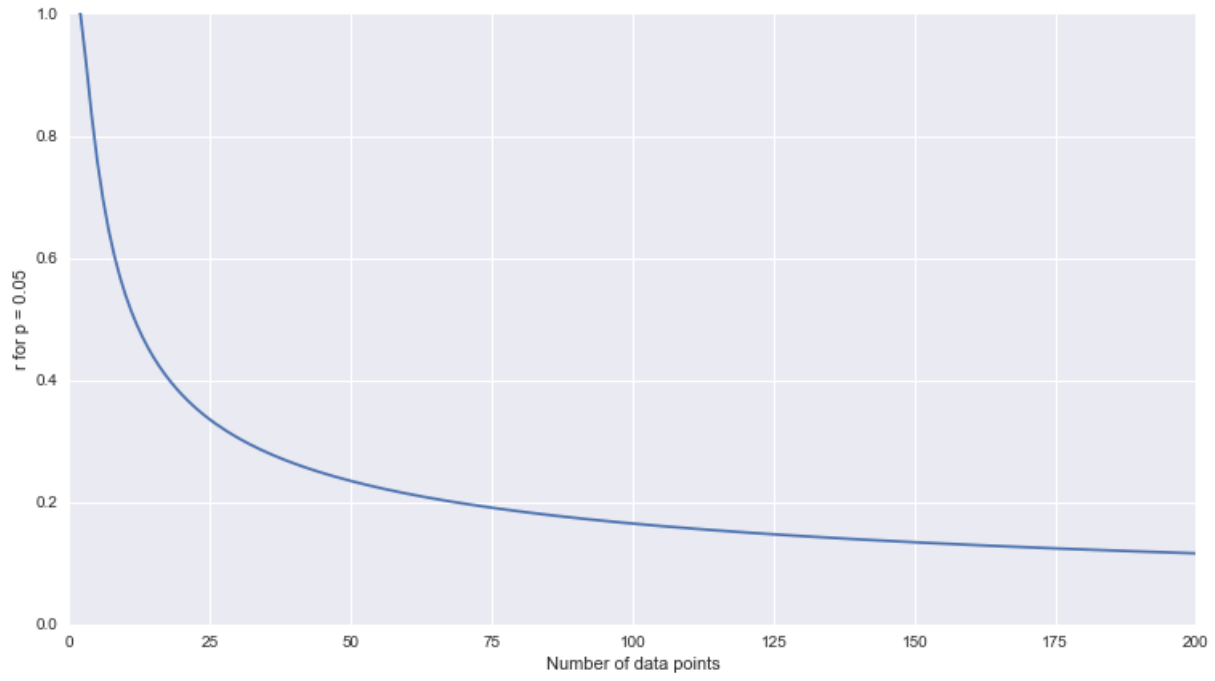

Figure 2.4: r value needed for p = 0.05, plotted against number of data points.

We want to test the null hypothesis that $\beta = 0$ – implying that X and Y are not correlated. Provided that sets X and Y are distributed by a bivariate normal distribution (which approximates well after transformations) it can be shown that the test statistic

$$t = \frac{\beta - 0}{\sqrt{\frac{MSE}{\langle x - \langle x \rangle \rangle}}} \tag{2.3}$$

where MSE is the mean-squared error from choosing a certain β, is equivalent to

$$t = \pm \frac{r_{X,Y}\sqrt{N-2}}{1 - r_{X,Y}^2} \tag{2.4}$$

where N is the number of data points.[1] The cumilitive probability density function of this variable is that of the student's t-test distrbution with $N-2$ degrees of freedom, given by

$$f(t) = \frac{\Gamma(\frac{N-1}{2})}{\sqrt{\pi(N-2)}\Gamma(\frac{N-2}{2})}(1 + \frac{t^2}{(N-2)})^{-(\frac{N-1}{2})} \rightarrow p = 1 - \int_{-\infty}^{t(r)} f(t)dt \tag{2.5}$$

Calculating p from 2.5 (found via 2.4, using r between two datasets) gives the a numeric value to the 'significance' of r between two sets of data.

[1] https://onlinecourses.science.psu.edu/stat414/node/254/

Figure 2.5: Dataset decomposed into groups

Note that as N tends to ∞, 2.5 approaches the normal distribution.

$$\lim_{N\to\infty} f(t) \propto \lim_{N\to\infty} (1 + \frac{t^2}{(N-2)})^{-(\frac{N-1}{2})} = \frac{1}{\sqrt{2\pi}} e^{\frac{-t^2}{2}} \tag{2.6}$$

A default p-value to group two datasets was set at 0.05 throughout the project, however this can be adjusted to suit the strength of correlation required within a group. The set of results shown in figure 2.5 are using default values.

2.3 Model Selection

Currently, the choice between ARIMA and GBRF for `complete set` is specified by the user. There is scope in the future to automate this procedure by looking at the properties of the data within each group and applying the optimal model per group. Details about what our testing indicated as the stengths of each group is presented within the handover files, and thus, with the appropriate conditionals, one could concievably create an automated model selector.

3 | ARIMA – *Earl Grey*

The regime in which the following model works well is primarily with heavily-seasonal and cyclical data. The ARIMA predictions are simply a 'best-guess' conjured up by considering previously observed behaviour (in fact, each future data point is considered as a linear combination of previous data points and their respective errors, but this will be discussed in detail later), hence the most accurate predictions will be observed when the data isn't fraught with aggressive level shifts, wild trends or noise. This type of behaviour, by in large, is considered to be 'out-of-scope' for a linear model such as ARIMA (see figure 3.1).

To first understand how an ARIMA model predicts, one must first understand each of the three constituent parts, or, to address the acronym, the 'AR', 'I' and 'MA' parts of the model and understand what contribution each part offers.

3.1 AR – Autoregression

The autoregression part of the model is commonly referred to as the 'monthly lag' and whilst this is somewhat of an overly-simplistic outlook it, nonetheless, encompasses the crux of the solution. To introduce some formality to the phrases – an AR process estimates the value of a data point, based on a certain (linear) combination of any number of the data points that have preceded it. Mathematically this can be written as:

$$Y_t = a_0 + a_1 Y_{t-1} + a_2 Y_{t-2} + ... + a_N Y_{t-N} \tag{3.1}$$

where Y_t represents the value of some quantity at time t (in this case it is normally spend in USD) and all of the a_i represent some unknown numbers that are determined by trying to fit the above equation on some number of data points and tweaking them (the exact

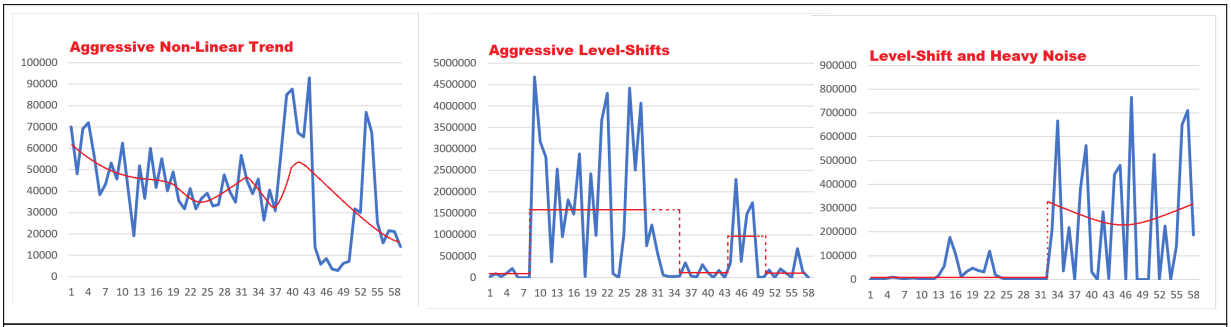

Figure 3.1: Behaviour that is difficult for linear models such as ARIMA to predict

mathematical underpinning is not explored here[1]).

To make intuitive sense of the equation one could imagine a shop that is selling some summer clothes. Clearly the price of an item this month must depend on the previous months, namely whether or not these were winter or summer months, and the previous summer. If an item was priced at, say, £12 last summer then it must be the case that a similar product cannot cost £4000 this year, and so the 12 month seasonal lag of the price would weight heavily. More specifically in the above equation we would see a larger value for a_{12} compared to all other a_i and in principle, there may be further non-zero values for a_1 and a_2 as these act to 'steer' the price throughout the sale campaign. After considering all of these factors 3.1 may reduce to

$$Y_t = a_{t-1}Y_{t-1} + a_{t-2}Y_{t-2} + a_{t-12}Y_{t-12} \tag{3.2}$$

where in this example Y_t represents the price of an item in this campaign.

3.2 MA – Moving Average

The moving average part of the model is commonly referred to as the 'monthly shock'. Unlike the autoregression discussed above, the name encapsulates what is at play here almost intuitively. Whereas the previous process reasoned that any data point was crafted from a combination of the data points that preceded it, the moving average tries to find how the 'noise' in the prior months propagates into future months.

One must be careful to understand what exactly is meant by 'noise' in this context as it is far more nuanced than simply referring to how loud a room is. Noise in this case is anything left unaccounted for in a previous data point – that is anything that could be considered either random or 'unpredictable'. It is much easier to explain what is meant by these terms with a simple example, namely the seasonal shopping sale from before.

At this point, one can hopefully accept the proposition that the price of an item is based upon the history of previous campaigns. Another layer of complexity to such a setup can now be added, namely within any given month, it is more reasonable to assume people have greater levels of disposable income closer to pay-day, and this amount of disposable wealth tails off as the month continues. Hence one would expect a larger amount of sales closer to these 'pay-day points' but more than this, people are indecisive and respond to other external influences aswell. If the weather is nice, people are more inclined to shop compared to when the weather is miserable. It is also conceivable that on a given day the business selling the items could have an online advertisement that drums up more footfall than on other days.

[1]see https: //en.wikipedia.org/wiki/Autoregressive_model

Now one could conjure up examples in order to crowbar further levels of complexity into the hypothetical shop scenario ad. infinitum but the essence of the problem described is that some contribution, to the sale price, depends on the levels of demand for the item and it is this component that is comprised of many different random, unforeseeable contributions. The combination of this random 'noise' then creates some level of influence in the price of the item, and so one may express this quantitatively as

$$Y_t = b_0 + b_{t-1}\epsilon_{t-1} + b_{t-2}\epsilon_{t-2} + ... + b_{t-N}\epsilon_{t-N} \tag{3.3}$$

where this time the ϵ_i represent the 'noise term' from each data point and the b_i again represent some other set of unknown numbers that reflect the importance of each term and have to be found by other, more complicated, means.[2]

3.3 Integration

The name 'integration' is somewhat of a misnomer for this process, however, one must adhere to convention and that is the name by which it is known. For the mathematically inclined reader, it is at this point that one should discard the usual mathematical interpretation of the term and instead directly substitute 'finding the level-line' in its place. The integration component of the model ensures that the two previously discussed techniques are 'oriented' in the correct direction. Namely if some quantity is known to be steadily increasing in price over time and has done so for the past 5 or 10 years then there is no point forecasting a future value to lie on a flat line. Instead the future values should be inclined above horizontal to continue the pattern(see figure).

To get the correct orientation for the series (more formally known as 'stationarising' a time series) a simple technique known as 'differencing' is employed. This process is rather an example of 'does what it says on the tin' and directly corresponds to subtracting consecutive elements from one another a certain number of times. Usually the number of differences needed to make the data set 'globally flat' corresponds to the order of the relationship between the variables, i.e. for the relationship $y = t^3$ one would require 3 differences to be computed before a dataset is stationarised. The 'usually' component of the preceeding statement has been substituted instead of 'always' to account for the fact sometimes, certain transforms are performed on the data set before the model is fit (see 'preprocessing') and so, in these regimes, differencing has somewhat of a diminished or more obfuscated mathematical intuition.

To illustrate differencing more clearly and see explicitly how 'stationarising' works, consider the following example. Suppose some response variable 'y' follows some mathematical relationship with time 't', that can be expressed as $y = t^2$.

[2]see https: //en.wikipedia.org/wiki/Moving−average_model

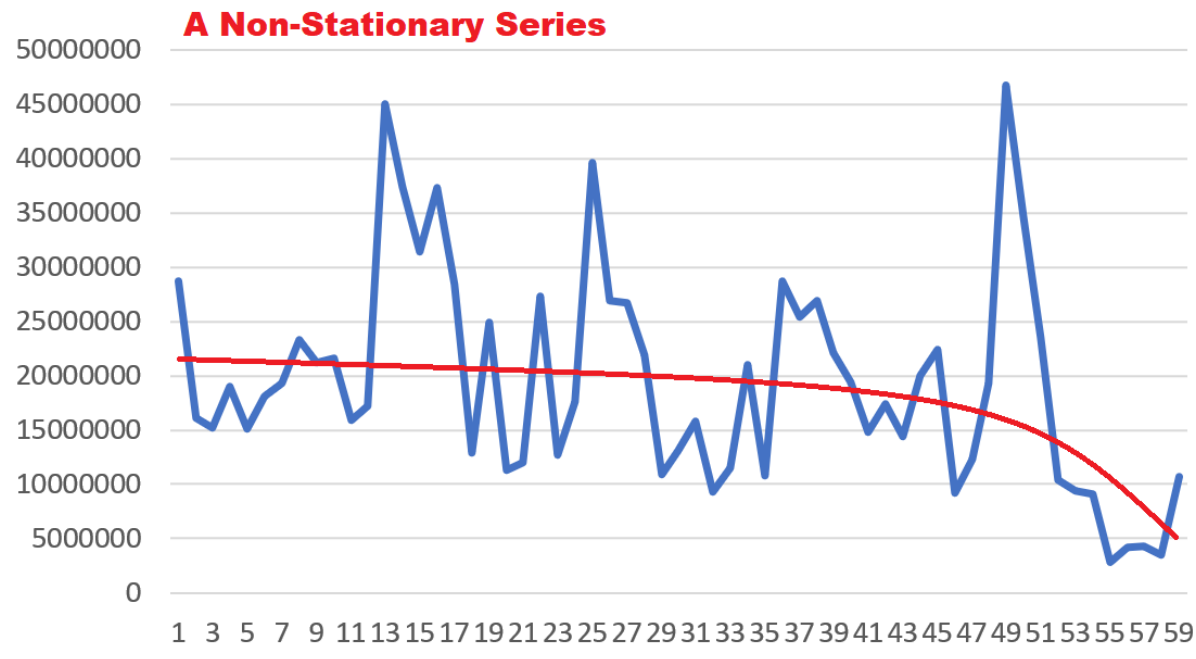

Figure 3.2: Example of a non-stationary time series.

For the first few values of t, the following data is generated,

$$y_t = 0, 1, 4, 9, 16, 25, 36, ... \tag{3.4}$$

and after one set of differencing, the series Δy_t is obtained, given by

$$\Delta y_t = 1, 3, 5, 7, 9, 11, ... \tag{3.5}$$

and after subsequent differencing, the final series $\Delta^2 y_t$ is obtained

$$\Delta^2 y_t = 2, 2, 2, 2, 2, ... \tag{3.6}$$

which one can now, hopefully, see is a stationary data set. Hence the relationship $y = t^2$ was stationarised after 2 subsequent differences, which goes some of the way to validating the claims from before.

In practise, it would be very computationally expensive to calculate difference after difference and somewhat arbitrary as one would have to provide some minimum threshold at which data would be considered stationary, due to the fact data sets are rarely devoid of noise or fluctuations in the real world. It therefore follows that the model instead uses a different mechanism through which it determines a data set to be sufficiently stationary but the underlying mechanics are identical.

3.4 Pre-processing

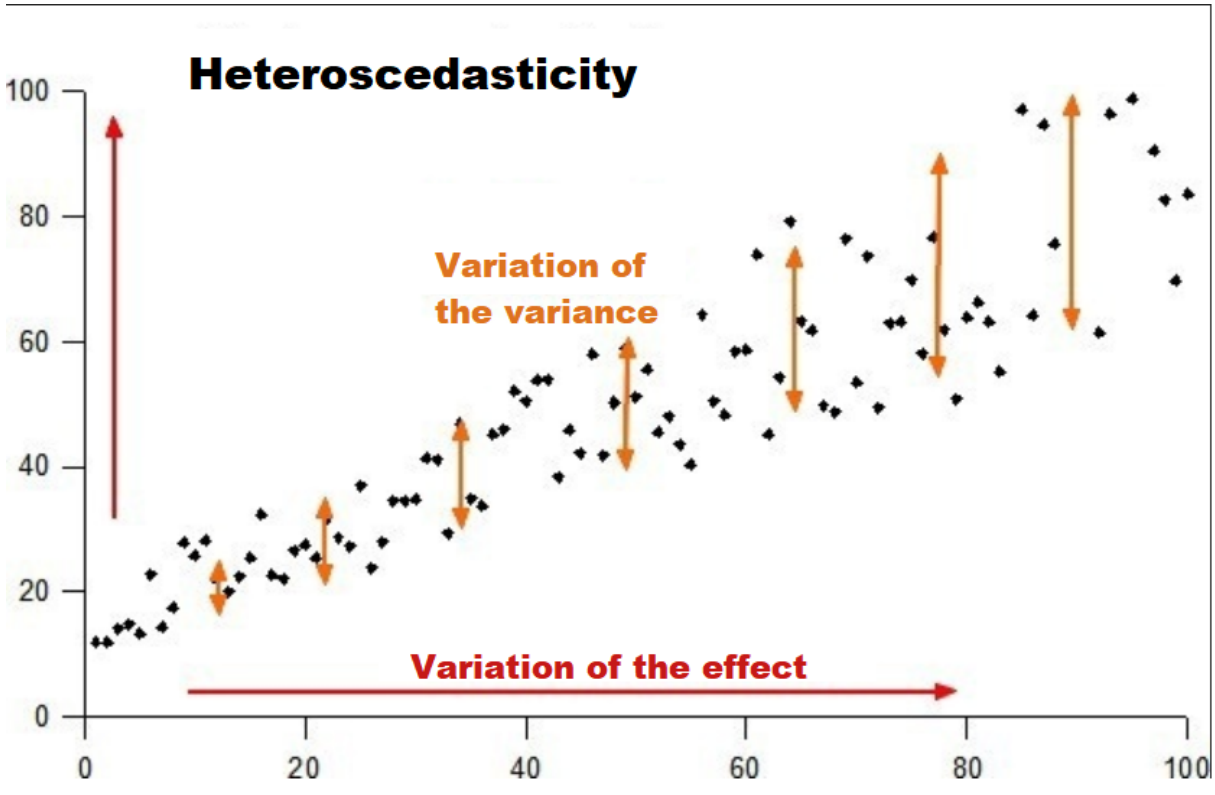

Figure 3.3: Example of heteroscedasticity.

Sometimes, data sets are encountered where, no matter how sophisticated the ARIMA model may be, there is simply no 'best guess' polynomial that captures the underlying behaviour and this may occur for a whole host of reasons. One such reason may be that the data set requires infinitely many terms to fully express its behaviour in a form similar to 3.1. Clearly this problem needs to be worked around before any meaningful progress can be made, but in practise there is a systematic way of detecting when this behaviour occurs and various methods for preventing it from occurring.

The generalization of the aforementioned problem is related to the variance of the data set, more informatively, it is the 'seasonal variation' being non-constant that gives rise to a lot of the problems. Having a non-constant variance throughout the data set is referred to as heteroscedasticity. If this effect is present in a data set an appropriate transform must be performed to remove them. Transforming the data set can, in principle, be somewhat of a dark art, however, a somewhat contrived example is provided to ease with understanding.

Suppose there is some data where a variable x has mean μ and variance σ_x^2. Now further

suppose that there is some new variable y, that depends on x through some unspecified function $f(x)$. One can now Taylor expand the variable y around the mean to give

$$y = f(\mu) + (x - \mu)\frac{df(\mu)}{dx} \tag{3.7}$$

and now the variance can be taken of both sides to yield,

$$\sigma_y^2 = \text{var}[f(\mu) + (x - \mu)\frac{df(\mu)}{dx}] \tag{3.8}$$

$$\sigma_y^2 = \text{var}[f(\mu)] + [\frac{df(\mu)}{dx}]^2\text{var}(x - \mu) \tag{3.9}$$

and now the final equation can be easily simplified using some basic statistical properties, namely that the variance of a scalar is 0, which eliminates the first term. The latter term is also further simplified by recalling that the variance is unaffected by linear coding, and so one now has

$$\sigma_y^2 = [\frac{df(\mu)}{dx}]^2\sigma_x^2 \tag{3.10}$$

At first this may not seem like much of a result, but in fact it is very surprising that the seasonal variance of a data set only varies as the square of a scalar related to the underlying map from one set of variables to another. To conclude this section an explicit but very generalised example is given. Namely when the seasonal variance is proportional to some power of the seasonal mean.

Suppose that the seasonal variances are given by some constant 'c' multiplied by the seasonal means 'μ' raised to an arbitrary power 'a'. Explicitly,

$$\sigma_x^2 = c\mu^a \tag{3.11}$$

and using 3.10 we can now see, to keep the variance still, we wish to have

$$\sigma_x^2[\frac{df(\mu)}{dx}]^2 = 1 \tag{3.12}$$

$$(c\mu^a)[\frac{df(\mu)}{dx}]^2 = 1 \tag{3.13}$$

$$\therefore [\frac{df(\mu)}{dx}]^2 = \frac{1}{c}\mu^{-a} \tag{3.14}$$

$$\therefore [\frac{df(\mu)}{dx}] = \frac{1}{\sqrt{c}}\mu^{-\frac{a}{2}} \tag{3.15}$$

$$\therefore f(x) \propto x^{1-\frac{a}{2}} \tag{3.16}$$

and in the case when $a = 2$ in 3.15, one is left with

$$f(x) \propto \ln(x) \tag{3.17}$$

which is the familiar logarithmic transform that is so commonly misused in data science literature. Such a detour has been taken to explain this small area of mathematics because it is a pre-requisite for ARIMA to only deal with homoscedastic data hence the model tests for, and performs, various transformations when necessary to ensure this constraint is met.

3.5 Parameter Optimisation

Now that a brief overview of the mathematical intricacies of some parts of the model have been given, it is time to address the crux of the problem, parameter optimisation. ARIMA models have three constituent ingredients and the output varies tremendously depending on the quantity of each ingredient. The first parameter 'p' corresponds to how many terms one wishes to keep in the autoregressive polynomial (i.e. how many lags are needed, see 3.1), the second parameter 'd' denotes the number of differences that have occurred to keep a time-series stationary (see 3.4 to 3.6) and the parameter 'q' illustrates how many terms one is keeping in the moving-average polynomial (i.e. how many shock/error terms are needed, see 3.3). There are two distinct ways of performing this parameter selection and both will be briefly outlined below, though in the closed-loop model, only one of the methods is implored.

3.5.1 Autocorrelators (open loop)

The vastly more popular way (in the world of data science) of selecting appropriate parameters for ARIMA models involves interpreting different measures of correlation between the lags and trying to remove these correlations from the residuals of the ARIMA model.

The name autocorrelation is somewhat unintuitive for the uninformed reader and so it is instead better to think of this as the pattern that would remain after smudging two objects over each other[3]. In fact, for the example of a time series, it is the pattern that remains after 'smearing' a time-series over a time-lagged version of itself. The way in which one chooses to 'lag' or 'smear' the two data sets gives rise to various different measures of autocorrelation.

When considering the autocorrelators, one must be careful to remember there are in fact two types; one for the autoregressive lags, the partial autocorrelator(PACF) and one for the moving average lags, the autocorrelator(ACF). There are also other auto-correlator measures that various industry models consider, such as the inverse autocorrelator(IACF) and the sample autocorrelator (SACF). Each of these have their respected uses, though the regimes in which any new information is obtained from these measures compared to the aforementioned two, are very few and far between to say the least.

It is somewhat straight-forward to check that one has selected the correct model to fit the data, as when the various autocorrelators have been computed and plotted, a random 'white noise' distribution to the errors and time lags should be observed, which shows that any underlying trends have been captured (see figure 3.4). With that being said,

[3]In fact one can prove that an autocorrelator function is in fact a manifestation of the Fourier convolution, this is known formally as the Weiner-Khinchin theorem.

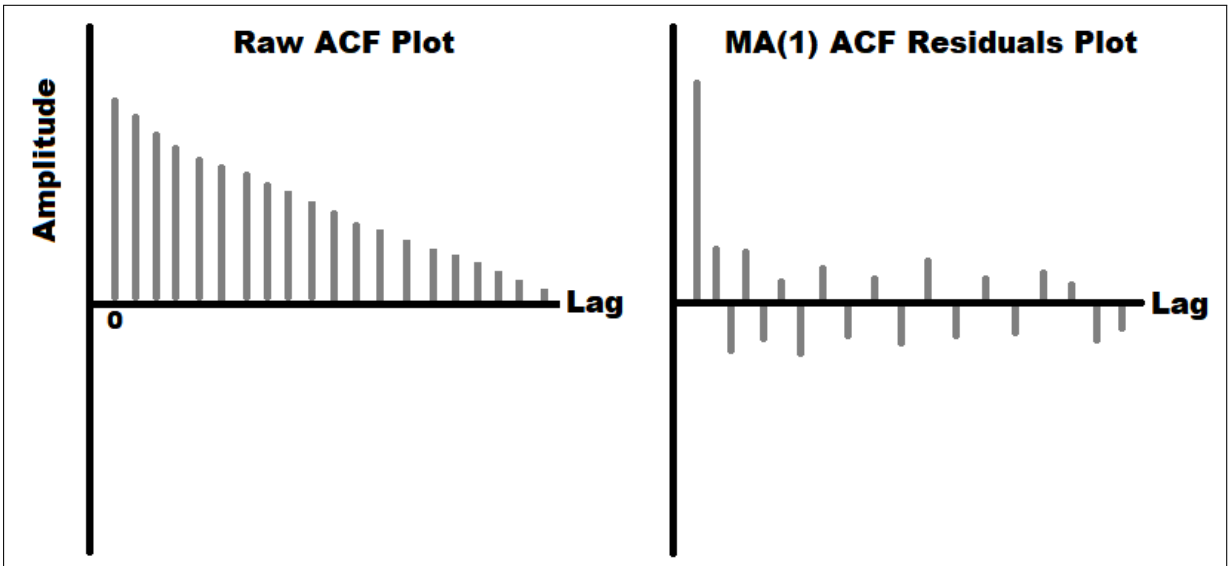

Figure 3.4: Example ACF and ACF residuals plots for MA(1) models.

it is a highly non-trivial task to correctly select the model needed to'untangle' the correlations that may be present. The main reason such exercises can be so convoluted is because 'echoes' may occur, i.e. certain lags may be seen in multiple places throughout the autocorrelator plots.[4]

In conclusion, the PACF plot will tell one how many lags are needed to remove any correlation from the AR residuals, which then determines the 'p' parameter and the ACF plot will tell one how many lags are needed to remove any correlation from the MA residuals, which determines the 'q' parameter.

3.5.2 Grid search (closed loop)

The way parameter optimization occurs in the Redmill package is using a novel grid searching algorithm. Grid searching works by building a plethora of models with different combinations of parameters and then measures the 'goodness of fit' of the models. The exact method by which the Redmill model tests for this fitting accuracy is unique to Redmill and not documented online, hence the specifics are not documented here but rather in the appendix of the document, instead what will be discussed here, is the advantages and disadvantages that occur with traditional, naive grid searching algorithms.

The obvious advantage to grid searching the parameters is that the conditions are almost perfect in the world of ARIMA. The parameter space is particularly 'shallow' com-

[4]This effect is essentially identical to 'harmonics' of a musical note.

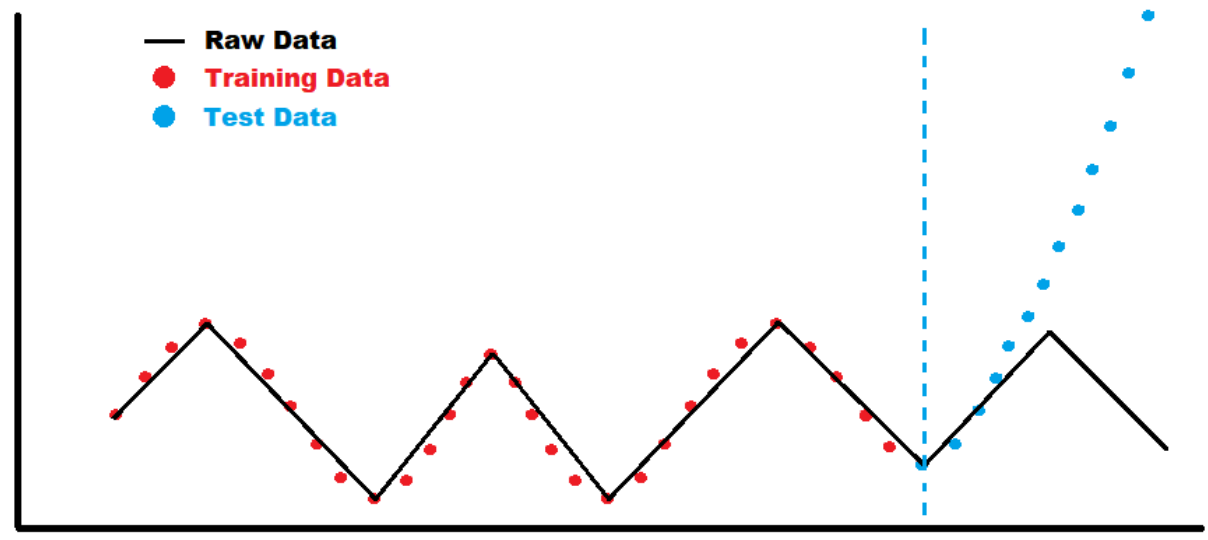

Figure 3.5: Example of overfitting – training set accuracy does not imply powerful long-term forecasting power for linear models like ARIMA.

pared to other regression methods.[5] This feature, in turn, means that very comprehensive searches can occur without the usual heavy penalty, in the form of computational expense, that such algorithms usually face.

A further advantage, to a grid search method, comes from the fact that a more quantitative, systematic approach can be employed as oppose to the rather 'dark art' method of interpreting various autocorrelation plots and aimlessly running models with painstaking residual checking. Having some quantitative measure of how well a model performs allows one to forecast with an amount of certainty that the discordant method of selective guessing simply does not offer.

The main disadvantage of this method, which usually stops this process dead in its tracks, is an issue of 'over-fitting'. The problem arises because the introduction of additional terms in the AR and MA polynomials provides more degrees of freedom, the residuals are then nearly always further minimised, which means the most complicated and computationally expensive models are always selected. When 'overfitting' occurs, all of the degrees of freedom are exhausted trying to fit the training data 'point-to-point' and then no flexibility remains when it comes to the test set(i.e. no forecasting power). This type of overfitting leads to flat, mean lines being drawn through the data set which is somewhat of a trivial prediction, or the predictions are mathematically unstable and tend to infinity or 0 after a few forecast points (see figure 3.5).

[5]see 'Bayesian Optimisation' for how to navigate vast, volatile parameter spaces

3.6 Support Vector Regression Hybrid (SVR)

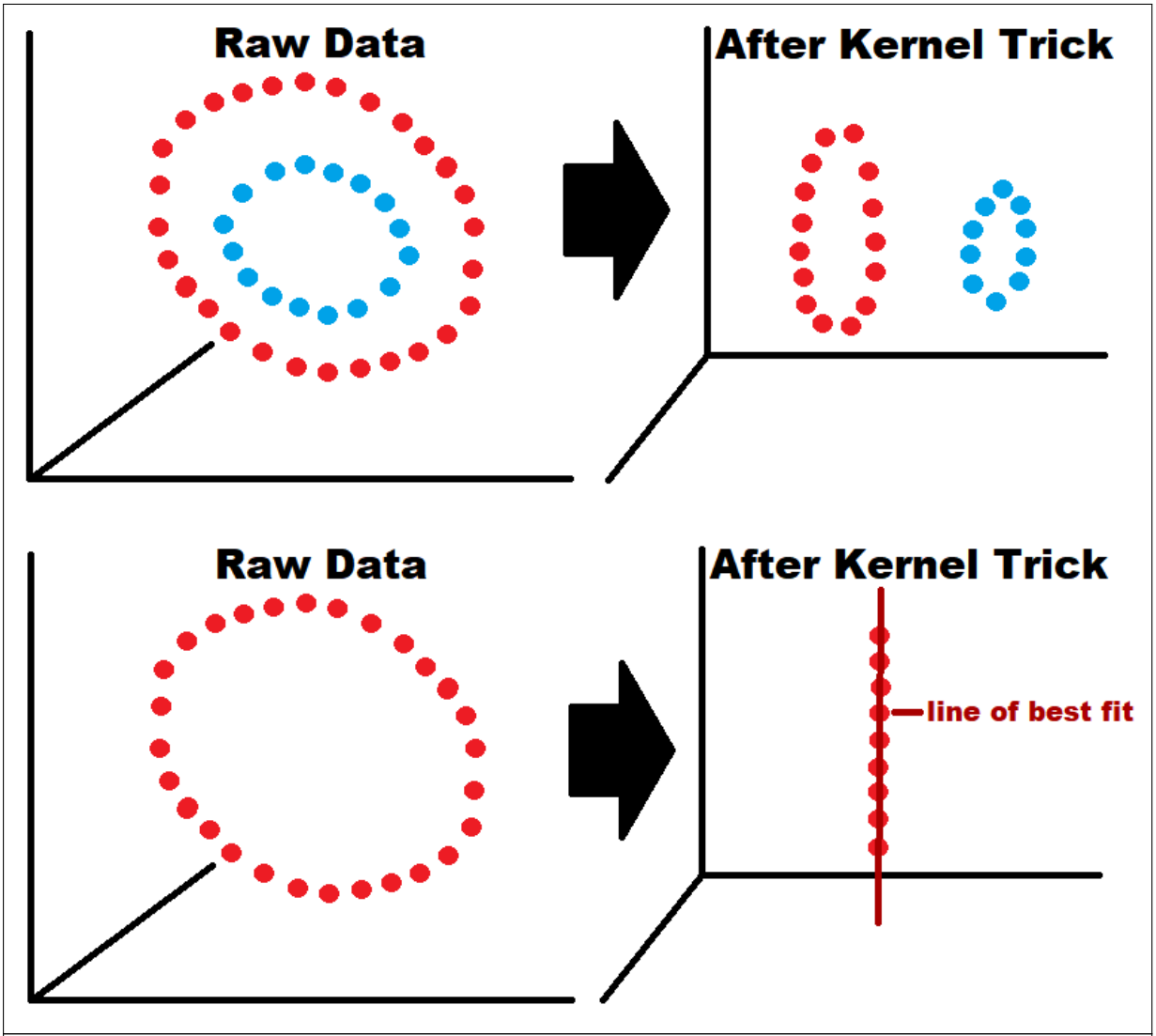

Figure 3.6: Kernel tricks.

The following section addresses the potential for the ARIMA hybrid model, though this element requires further coding and testing before it is properly introduced into the 'Earl Grey' model, hence this component is still 'under construction'. Initial testing that was conducted using a VAR-SVR hybrid on holding company spending data showed some very promising results indeed.

The opportunity for an ARIMA-SVR hybrid is obvious from the get-go when conducting time series forecasts and could, in theory, provide unique results and levels of accuracy that

standalone ARIMA models cannot achieve. Before further delving into the potential uses, one must detour slightly to understand the core principles behind an SVR.

SVR uses a concept known as a 'kernel trick' to draw best-fit lines through data sets. In essence, a kernel trick refers to the procedure of choosing an appropriate transformation to align a set of scattered data points, so that a straight forward linear regression can be performed on the data set and a straight line can be drawn through it, which can, in turn, be used to forecast future values (see figure 3.6). This method of using a kernel trick was primarily used for classification problems originally, mainly to draw a line of distinction between two or more, up until now, indistinguishable groups (see figure 3.6) though in recent years the regression analogue to this process has seen increasing popularity.

In the proposed hybrid model, the SVR would be used to draw a non-linear trend line through the data set and cut through all of the 'noise response'. This data set could then be de-trended and ARIMA would just have to focus on the linear behaviour, not captured by the SVR. The advantage of this is that ARIMA is a solely linear regressor, so any 'exotic' behaviour [6] will remain uncaptured, but this problem is surmounted using SVR. The SVR will always orient the linear response from the ARIMA model in the correct direction and free up some of the degrees of freedom.

After the SVR line has successfully de-trended the data set, the non-linear mean line can be forecasted forward in time and then recombined with the ARIMA response.

[6] For instance, think about trend lines that can only be expressed as a fractional number of differences!

3.7 How The Model Works

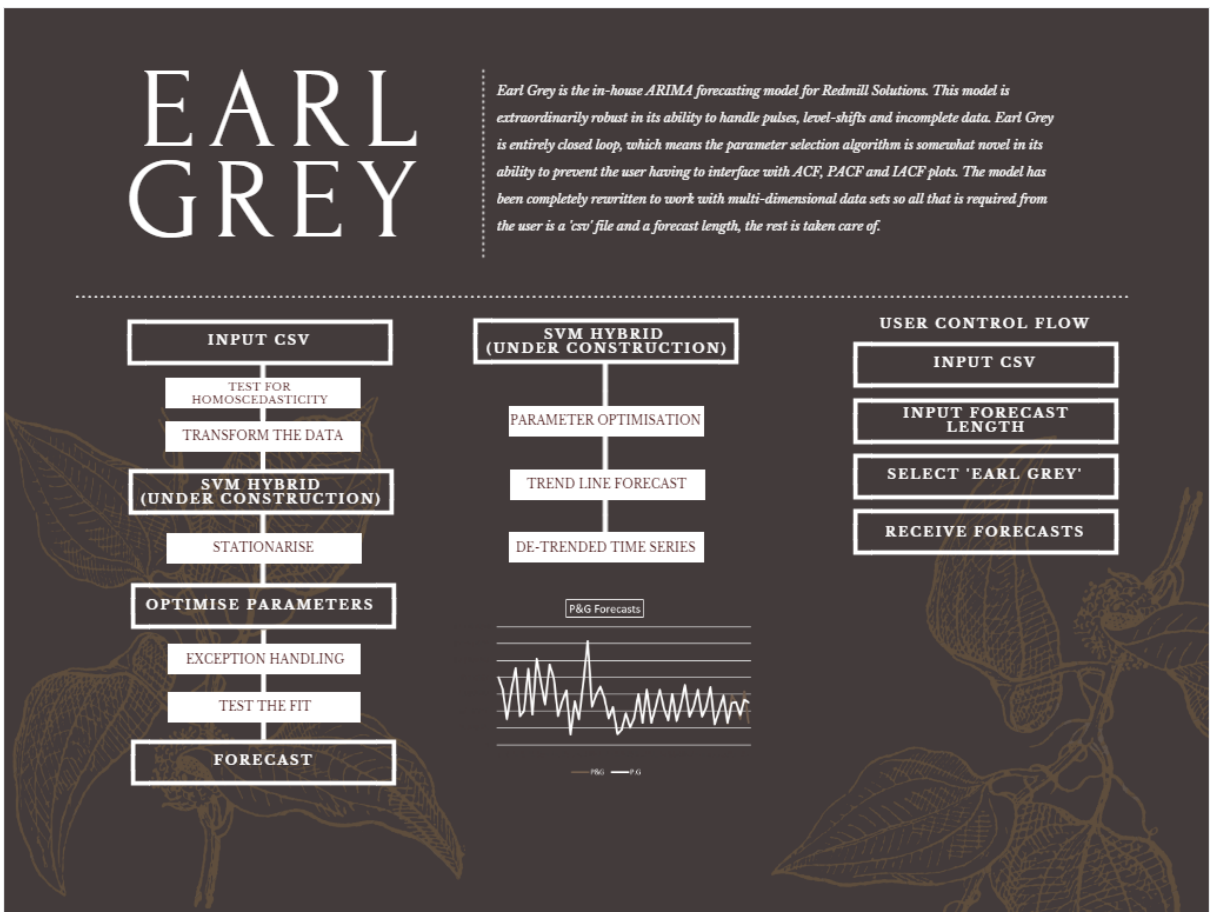

Figure 3.7: Work-flow diagram of the *Earl Grey* model.

Now that all the individual components have been discussed, all that is left is to briefly discuss how the components are combined in the final model.

The exact flow of the model is illustrated in the accompanying diagram (see figure 3.7) though the complexity hidden within each step varies throughout the different stages. The first process that occurs in the model is a test to see if the data is sufficiently homoscedastic[7] (see 'Pre-processing'), this process is carried out in a statistical hypothesis fashion, with the null hypothesis being that a certain transform is not needed and the alternative is the converse. In principle, if one wanted to make the criteria more stringent or lenient for performing transforms, the tolerance variable within the `checkHSkedastic` function would need adjusting to suit.

[7]Aston, J.A., Findley, D.F., McElroy, T.S., Wills, K.C. and Martin, D.E., 2007. New ARIMA models for seasonal time series and their application to seasonal adjustment and forecasting.

Next, the hybrid component of the model would run, de-trending the time series and leaving only the noise to be forecasted, the parameter optimisation algorithm within the SVR will be either a multi-layered grid search or a Bayesian optimisation scheme. The grid search algorithm in this case will be different from the ARIMA scheme as the 'goodness of fit' will be determined, based upon a different set of criteria.

After the data has been transformed (and de-trended) the data set must be stationarised. This process proceeds using a procedure first documented in an economics research paper.[8] This procedure uses a statistical test to find the probability that a unit root has been detected in the trial polynomial for the proposed ARIMA model, the model then recursively differences the data set and performs the test again until all unit roots are deemed statistically insignificant (i.e. not present).

The penultimate step to the model is parameter optimisation, this uses the closed loop variant mentioned in the self-titled section. Within this parameter optimisation subroutine, error handling is conducted to ensure all of the models are viable and selects simpler models in favour of overly-complex models that risk 'over-fitting'. The novel parameter selection method means that the long outdated method of using model selection criterion can be completely discarded, as this methodology is fundamentally flawed and very dangerous in the regime that 'Earl Grey' needs to work in.

Finally, after the correct configuration of parameters has been chosen, the model then performs 'rolling forecasts,' a process whereby one data point is forecasted using the model and then agglomerated into the training set and used to forecast a further data point. This process repeats as many times as needed (dictated by the desired forecast length) before the final predictions are returned to the user. This method of single point prediction and then updating minimises the exponential error propagation that occurs in ARIMA models, limiting the amount of exotic behaviour that can be carried through the forcasted data points.

[8]Schwert, G.W., 1989. Why does stock market volatility change over time?. The journal of finance, 44(5), pp.1115-1153.

4 | VAR – *Chai*

The *Chai* model, Redmill's vector autoregression package is the most robust of all the forecasting packages Redmill has to offer and is used as somewhat of a 'work horse'. This model is predominantly used to handle: sparse, incomplete and single pulse data sets. [1] Consequently, this model contains the most intricate error handling sub-routines and optimises it's intrinsic parameters using a safer, averaged grid search procedure, both of these features will be discussed in detail below.

4.1 Vector Auto Regression

In principle, all of the machinery under the bonnet is captured in the 'ARIMA – Autoregression' section (section 3.1). The only difference with the vector analogue is the mathematical gymnastics that occur to estimate some of the coefficients. With this being said, what will be given here instead is a more rigorous derivation as to how the generalised process works. This section can be omitted, in principle, it simply serves as a curiosity for the mathematically inclined that wish to know more.

To perform vector auto regression one may start, exactly as before, by assuming a target variable $y_{1,t}$ can be written as

$$y_{1,t} = a_0 + a_1 y_{t-1} + a_2 y_{t-2} + ... + a_N y_{t-N} \tag{4.1}$$

but now, further terms must be introduced to account for the fact the target variable is no longer unique, i.e. the target variable is now a vector, $\mathbf{Y}$, comprised of all the other target variables,

$$\mathbf{Y_t} = [y_{1,t}, y_{2,t}, \ldots, y_{N,t}]^T \tag{4.2}$$

where 'N' specifies the number of target variables. Each element of this vector now depends on a linear combination of all the different lagged values across all the different target variables, so now one may write

$$\begin{aligned}
y_{1,t} &= c_1 + a^1_{1,1} y_{1,t-1} + \ldots + a^1_{1,N} y_{N,t-1} + \ldots + a^p_{1,1} y_{1,t-p} + \ldots + a^p_{1,N} y_{N,t-p} && (4.3)\\
y_{2,t} &= c_2 + a^1_{2,1} y_{1,t-1} + \ldots + a^1_{2,N} y_{N,t-1} + \ldots + a^p_{2,1} y_{1,t-p} + \ldots + a^p_{2,N} y_{N,t-p} && (4.4)\\
&\vdots \\
y_{N,t} &= c_N + a^1_{N,1} y_{1,t-1} + \ldots + a^1_{N,N} y_{N,t-1} + \ldots + a^p_{N,1} y_{1,t-p} + \ldots + a^p_{N,N} y_{N,t-p} && (4.5)
\end{aligned}$$

[1] For the mathematically minded, a lot of the data is cluttered with Dirac-delta's and consecutive zeros.

where 'p', denotes the number of lags one wishes to use in the var(p) process. It is then considerably tidier to denote this system of equations in matrix and vector form and so some new notation is introduced,

$$\begin{aligned}
\vec{A_1} &= [a_{1,1}, a_{2,1}, a_{3,1}, \cdots, a_{N,1}] && (4.6)\\
\vec{A_2} &= [a_{1,2}, a_{2,2}, a_{3,2}, \cdots, a_{N,2}] && (4.7)\\
&\vdots \\
\vec{A_j} &= \sum_{i=0}^{i=p} a_{i,j}\hat{e_i} && (4.8)
\end{aligned}$$

where the $\hat{e_i}$ are unit vectors in the specified direction and we also have,

$$\mathbf{B} = [\vec{c}, \vec{A_1}, \vec{A_2}, \cdots, \vec{A_p}] \tag{4.9}$$

lastly a matrix comprised of the target variables needs specifying at all the different lags, for some time t

$$\mathbf{Z} = \begin{pmatrix} \vec{y_0} & \vec{y_1} & \cdots & \vec{y_{t-p}} \\ \vec{y_1} & \vec{y_2} & \cdots & \vec{y_{t-p+1}} \\ \vdots & \vdots & \ddots & \vdots \\ \vec{y_{p-1}} & \vec{y_{p-2}} & \cdots & \vec{y_{t-1}} \\ 1 & 1 & \ldots & 1 \end{pmatrix}$$

Now that all of the cumbersome notation has been taken care of, the original regression equation can now be succinctly expressed as

$$\mathbf{Y_t} = \mathbf{BZ} \tag{4.10}$$

The problem now boils down to solving this equation for the matrix of coefficients, $\mathbf{B}$. The inherent flaw in solving such a an equation, lies in the fact that this equation is easily solved by multiplying through by $\mathbf{Z^{-1}}$, but there is no guarantee that this matrix exists as $\mathbf{Z}$ could be singular! Instead, the common practise is to multiply both sides through by the transpose of $\mathbf{Z}$, which is a sensible choice as $\mathbf{Z^T Z}$ is always invertible as it is square and symmetric in nature. After carrying out the multiplication, one is simply left with

$$\mathbf{Y_t Z^T} = \mathbf{BZZ^T} \tag{4.11}$$

whereby now an inverse of one side can be taken to yield

$$\mathbf{B} = \mathbf{Y_t Z^T}(\mathbf{ZZ^T})^{-1} \tag{4.12}$$

which can now be solved to yield all of the coefficients needed to best-specify the original auto regressive system.

4.2 Parameter Optimisation

The parameter selection algorithm for *Chai* proceeds almost identically as the one for 'Earl Grey' with the exception that the goodness of fit this time is an averaged goodness of fit across all of the target variables. This averaged measure can be changed somewhat if a particular target variable is wanting to be forecast at a higher accuracy than the others, in this case, a weighted average would instead be used.

The grid search also proceeds in a slightly different manner to before, this is because in a strictly auto regressive model the only parameter to be optimised is the 'p' parameter. As there is no longer an integration component to the model, a drift term (constant or time-dependent) can be included in the auto regressive polynomial, to capture the overall trend of the data set.

Along with the slight modification to the grid search, there is also some exception checking which allows singular pulses and incomplete data set to be handled. This set of criteria has been determined empirically after rigorous testing on many data sets.

4.3 Forecasting

The forecasting algorithm for *Chai* is a straight forecast, rather than the rolling forecast from the Earl Grey model as the average saving in computation time far outweighs the decrement in forecast accuracy. The main premise of the *Chai* model is to be unparalleled in speed and dynamic in its required inputs, hence sophistication and nuance has to be traded away in places.

5 | GBRF – *Chamomile*

5.1 Gradient Boosting Regression Forest

This section outlines the methodology behind one of the major predictive tools used in the *Chamomile* model, the GBRF. A GBRF is composed of three major algorithmic developments that combine to create a very robust and versatile machine learning algorithm. The basic underlying principle is that of a decision tree, covered in the following subsection. This is generalized into a regression forest. Finally the gradient boosting algorithm is introduced for faster training times and more robust performance.

5.1.1 Decision Trees

The simplest regression problems involve the estimation of some function, $f(x)$, which maps one number onto another, denoted $f : \mathbb{R} \to \mathbb{R}$. We estimate $f(x)$ by fitting a curve to a set of points with co-ordinates, $(x, \tilde{f}(x))$, that we know to at least approximately follow this mapping. This might not be an exact mapping due to the presence of noise in the collection of the data, for example.

The CART decision tree algorithm was developed by Breiman et al. The algrotihm works as follows: a data set is divided into separate groups in such a way that the response variables ($f(x)$) of the samples in each group are similar to one another. This split is made by querying the data with a condition on the explanatory variable (x); the points that satisfy this condition are put into one group, and those that do not are put into a separate group. When the tree has finished splitting, either because no splits can be made that improve the fit or because a limit has been imposed by the user, the response variable of all the points in a group are averaged, giving a value to each 'leaf'. To get a prediction for a single explanatory variable, $x\prime$, the point is placed at the top of the decision tree and moves down the branches, following only the branches for which it satisfies the query that splits that branch. Once $x\prime$ reaches a leaf, the value of that leaf is assigned as the response variable for $x\prime$. A split is determined by choosing a query such that the impurity of a node is minimised. The impurit of a group with N points is defined as;

$$\sum_{i}^{N} (x_i - \bar{x})^2 \tag{5.1}$$

where

$$\bar{x} = \frac{1}{N} \sum_{i}^{N} x_i \tag{5.2}$$

5.1.2 Gradient Boosting Regression Forest

The term 'Regression Forest' refers to any algorithm where many individual 'weak learners' are combined to produce a better prediction. A weak learner is an individual model that, by construction, fits the overall data relatively poorly, but when combined these weak predictions give a much stronger one than any individual model ever could. There are many different algorithms that build these forests of weak learners in different ways. GBRF is one of the most popular variants of this general purpose algorithm.

GBRF builds a predictive model one weak learner at a time. The forest is trying to estimate some function $y = f(x)$. An initial guess at the function is made, $F_1(x)$, which does not perfectly model the output of the function. The model then assumes that there is some additional function, $g(x)$, that can be added to the initial guess, such that

$$y = F_1(x) + \alpha_1 g(x) \tag{5.3}$$

This can be trivially rewritten as:

$$g(x) = \frac{y - F_1(x)}{\alpha_1} \tag{5.4}$$

In other words, the second forest ($g(x)$) is fit on the *residuals* of the first guess. A model is then iteratively built up by continually fitting the residuals of the previous best guess at the function.

For those interested, the term 'gradient boosting' originates from the fact that, depending on the loss function used by the algorithm, fitting the residuals is equivalent to finding the gradient of the function $F_1(x)$ and stepping downwards towards the minimum of the loss function. This is not one of those cases but the name is so common that it would be silly to change it for our purposes.

A key point that should be evident from the discussion of how decision trees work is that this algorithm is not well suited to predicted future values that have not already been seen before - it will struggle to continue to forecast a growing signal because the leaf values are the averages of previous values seen in the training sets. This is not a problem for the high and medium frequency components where the values oscillate back and forth. However the low frequency components represent the trend of the data, and hence are expected to potentially climb above any previously seen values. Therefore the GBRF algorithm is not suitable for this component. Instead a neural network is used.

All work with the GBRF in this project was conducted using the SkLearn package `GradientBoostingRegressionForest`, from their ensemble learning packages.

6 | Results

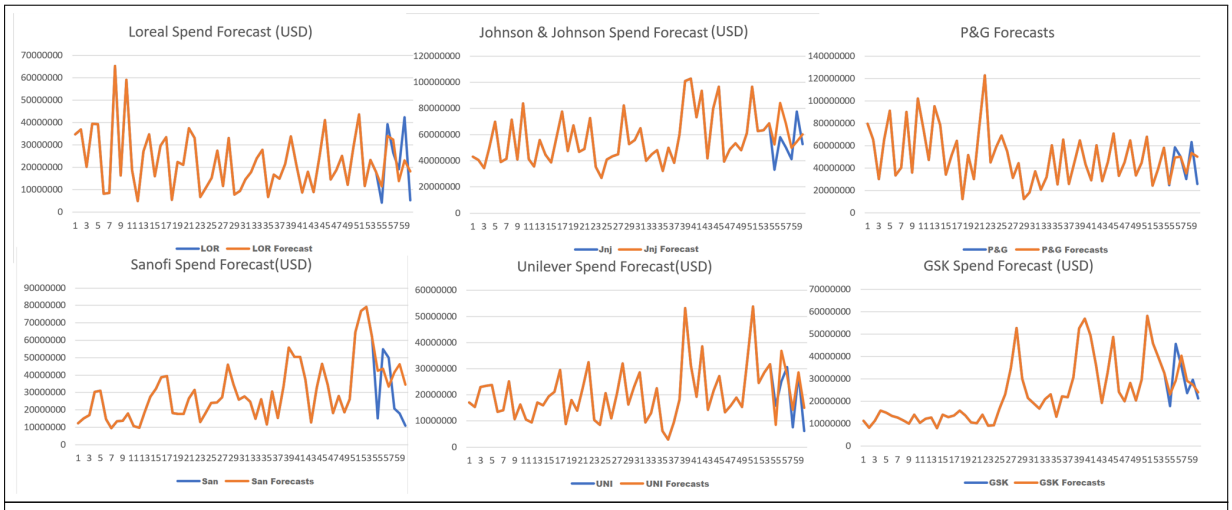

Figure 6.1: ARIMA results

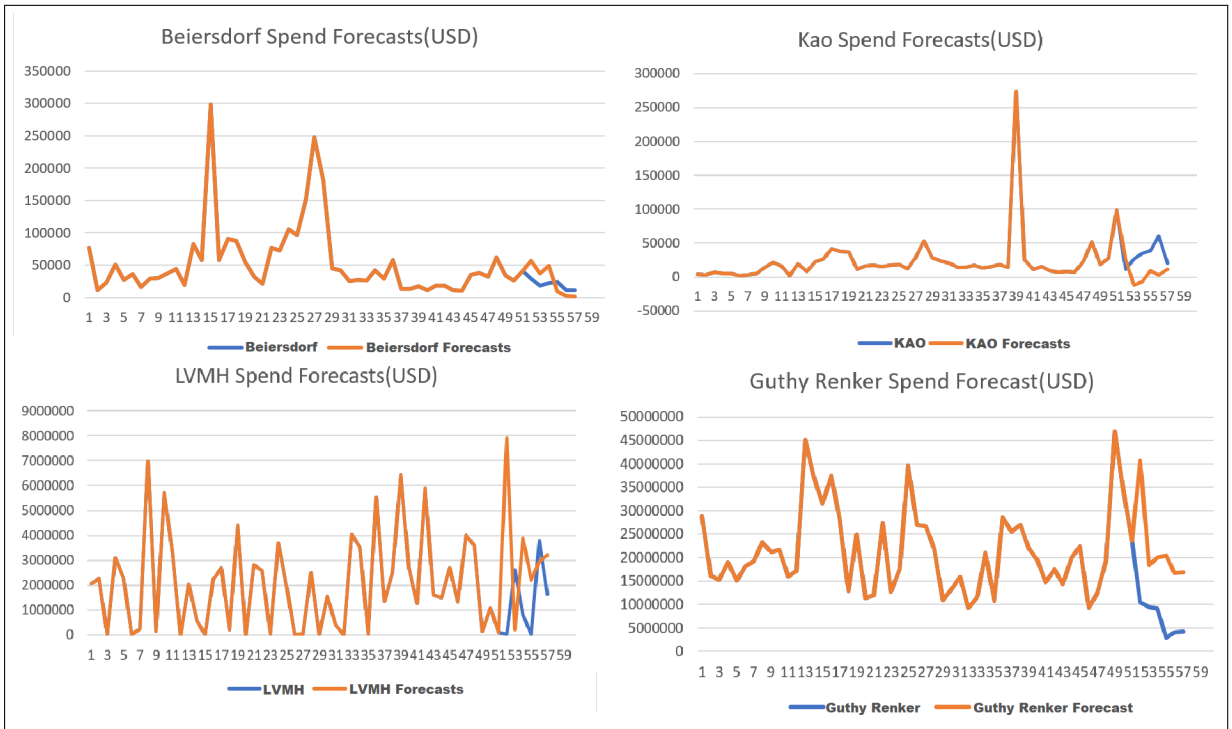

Figure 6.2: VAR results