HEC Montreal - Supervised Project

Comparative Studies of Conventional and Machine Learning Models in Online Retail Arbitrage

Haiyang Bao

Supervised by: Yossiri Adulyasak

2019-08-11

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Problem Statement

The process of forecasting has traditionally been the focus of econometrics where time-series data such as stock price and GDP are the major topics. Many established methods and models are available and more efforts are spent into the research. It is evident that with the arrival and spread in popularity of machine learning, neural network for a particular example, researchers and data analysts alike are harnessing these powerful tools for data science across the fields, armed at the same time with unprecedented amount of data from an ever more diverse sources, from human eye movement onto products to satellite images on shipment traffic. One of the ultimate goal is of course forecasting or prediction. The media coverage of machine learning applications is arguably largely on image recognition, speech and voice and perhaps natural language, all directed towards making machines artificially intelligent in human-like sensory capabilities. For more less-hyped data science topics, in the traditional econometrics or operation research fields, the use of machine learning and neutral network has received relatively less attention. The nature of machine learning as a whole still is about data analysis. In fact most of the methods and models have been available for decades. It is nothing more than another tool in the analytical box for an expert data scientist or statistician, as they had been more known for.

Accurate forecasting is a key competitive advantage in supply chain management, or to a larger extent in research, due to the practical demand and nature of problems that are applicable and widespread in the field. For example, most decisions in supply chain design hinge on the quality of inputs feeding into the grand optimisation model, such as periodic sales, cost figures in production and distribution, materials requirement in production and many more. For a retail store of general consumer products, for instance, a correct forecast of sales of items will lead to lower cost from saving in inventory and storage and simultaneously higher service level from satisfying demand. Such output-side of prediction is critical to perishable products such as milk, and food in general, where overestimation of sales leads to significant cost in wasteful production. In fact, the accuracy of final sales in a supply chain is a fundamental factor in the cost calculation and planning of the rest of the supply chain operation - from delivery planning to input purchase, from production planning to hiring decisions, and so on. Although big decisions such as supply chain design and manufacturing or warehouse setup depend on long term strategic business factors beyond short-term sales variations, the concept of forecast always play a vital role in any planning decision - macroeconomic forecast of sectoral growth or even national GDP is of the same nature of data analysis. As a result, demand forecasting is an essential capability for any operation. If an operation has quality data on where and how much the final output is sold for a period of time, other decision and planning can be derived subsequently. Therefore the extent to which an operation can forecast its demand can be a key competitive advantage in the market, with implications on profit and cost.

In the ideal system, the management decision making aims to fulfill or supply the “right” quantity of product or service to the “right” place at the “right” time, with the notion of “right” being determined by the demand of customers, ultimate or intermediate throughout the supply chain. And one of the main difficulties for the management is to decide on the “right” parameters, because fulfillment entails lead time: the amount of time in receiving the right inputs, in producing and delivering to the right place at the right time for the customers. Ideal fulfillment thus requires knowledge in advance about the demand information at the final consumers; for example, the high street individual consumers in a retail fashion supply chain, or the brand automobile firms such as BMW purchasing components in the upstream supply chain of auto industry, however the supply chain is defined. Once the information on the end consumer demand is obtained, what and when and how many are demanded, the rest of the planning in order to fulfill that demand is much less of a variation than calculation and optimisation, albeit other variabilities still exist in delivery and production stages. In particular, information on the price and stock variations are two of the key factors - in a competitive marketplace the price movement and the stock level and trend both are direct signals of demand impact in the near term.

In this paper, the focus is to compare different forecasting models in the context of online retail arbitrage. Because of market forces and asymmetric informations and conditions, products prices vary significantly across the different real or visual marketplaces. Based on the dataset of a set of game products sold on the online marketplace Amazon, the goal is to test the validity of accurate forecasting of online retail products and their price and stock movement by implementing various types of models that are commonly used for prediction. Although the dataset employed in this case is limited in scope and the context is strictly Amazon marketplace, the idea can be transferred to the study and application of procurement process in supply chain management. In procurement, the goal is evidently not to focus on the exploitation of differential prices from arbitrage. However, the ability to predict the price and stock movement of online products can be significantly useful to the cost reduction and better efficiency in procurement. For example, the need to purchase standardised supplies is omnipresent and often managed with little decision process. If an automatic price prediction system can be created that forecast the price movement of the require supplies through online marketplaces, the benefit of cost cutting will be advantageous.

1.2 Objective

CHANGE THIS

The goal of this paper is to compare both traditional and current machine learning models for prediction as applied on a sample dataset representing the online arbitrage scenarios. Specifically, the paper selects exemplary models typically employed in forecasting in both traditional and machine learning fields and test them on a set of consumer product time-series dataset. The main objective of the project involves surveying and implementing some of these machine learning models and theories applicable to time series data.

In brief, the dataset applied consists of a collection of data regarding video games listed on Amazon and Best Buy over time, up to 230 days. The aim is the build a model that achieves quality metrics in predicting the variation in price and stock of these items sold online. Albeit not an entirely realistic retail operation nor is there much control over the data and operation, this dataset reflects characteristics of supply and demand in a competitive market. Its result could be used as reference framework and model pipeline to many applications in forecasting real supply chain operations.

Given the depth of of machine learning and data science, the scope of this project is not to illustrate and study the implemented models in depth, such as constructing the theoretical ground, the mathematical derivation of the models and validity. Equally beyond the scope of the project is the searching of optimal architectures and hyper-parameters of the models, as these tasks are difficult in general and computation and time consuming for the many models employed in the project. The goal as mentioned is the apply the techniques with common setup, in model design and hyper-parameters setting, and compare the results with the same input dataset on the same common evaluation metrics. Of course, the particular dataset and the probabilistic nature of the models will mean that the results are not robust for production implementation. As the project is an exercise of the applying machine learning techniques to a traditionally optimisation framework oriented area of study, as Operation Research has been known for, any production model framework will require further optimality search in model design and hyper-parameter tuning, as well as the tuning to particular types of dataset and tasks.

1.3 Outline

The paper is organised into 4 chapters: it begins with a brief literature review on the methodology of forecasting including established and current machine learning frameworks, followed by a detail explanation of methodologies employed in this paper, then a pipeline of data cleaning and preprocessing is demonstrated and finally the series of models are illustrated with their results compared and analysed.

In the literature review chapter, a brief overview of some of the common and employed forecasting models in this paper are described, including ARIMA, Fixed Effect panel model, Elastic Net and Structural Time Series.

The chapter on methodology consists of explanation of each model applied in this paper, with the scope of understanding in application not theoretical derivation. Evaluation and metrics of comparison are also explained.

Data preparation includes the pipeline and description of transforming the dataset of interest from its raw state into the appropriate format and quality corresponding to the objective of the paper and the models implemented: cleaning missing data, creating the right target variables, etc.

Further, the cleaned dataset is then applied through each model; with common metrics to be evaluated from the results of each model, the final comparative analysis of performance concludes this chapter.

Chapter 2

Literature Review

2.1 Time Series Analysis

Time series analysis can be broadly described as the analysis of temporal data or random variables, and thus comprising of all aspects or models that could be associated with time series data. Time-series analysis has been a primary type of quantitative analysis in many disciplines, such as economics and finance in which the problems involve data over long period of time. Particularly in finance where time-series analysis has developped a wealth of models largely originated by both rich and diverse data sources generated from the financial markets worldwide. The primary objective of time series analysis is inference: the studies of constructing mathematical models emulating the generative process of sample data over time; for this task the approach known as time domain is a common method that presumes some underlying correlation between adjacent points in time or dependence between the past and present observations (Shumway and Stoffer 2017). The approach tries to study the functional mapping with current and past values as parameters to be estimated for generating the future values.

Forecasting on the other hand can arguably be a distinct subset of the general time series analysis, because the objective of forecasting often is only concerned with the quality of prediction of future values of some variables, as opposed to the in-depth study or understanding of the generative process, including the factors that may directly influence that process. Further, forecasting also can vary according to the time window or horizon to be predicted, types of data patterns, contextual nature of data, etc. There are wide ranging models can be used to forecast time series data, from the simple method such as the naive method, using the most recent observation, to highly complex models such as neural nets or deep learning. The choice is often not as clear because of factors such as the availability and quality of data, the predictability of context, time horizon to be forecast, and so on (Hyndman and Athanasopoulos 2018).

Broadly, forecasting has been growing as a distinct subset of statistics for decades. Gooijer and Hyndman (De Gooijer and Hyndman 2006) published a survey at the 25th anniversary of the International Journal of Forecasting that provides an invaluable overview of the major development of theories and models in forecasting up to its time. For example, models such as ARIMA and Exponential Smoothing have been two particularly well studied methods for time-series data and the stable in academic and application, demonstrating excellent practical results and attained popularity in many fields.

Time series forecasting has its particular difficulties. Compared to cross-sectional classification and regression, time series data bear the characteristics of temporal dependence among observations. As a result, special preprocessing of data and feature engineering - for more advanced machine learning models with multivariate time series - are necessary in modeling. Additional structures such as trend and seasonality also require treatment. Conventional methods for time-series forecasting has been dominated by linear, additive models like ARIMA for its well studied properties and effectiveness. However, such methods arguably bear certain deficiencies such as (Gamboa 2017):

- **completeness of data**: missing and unstructured data generally unsupported
- **linearity**: assumption of linear relationship fails to capture complex joint distributions
- **fixed temporal dependence**: the number of windows or lags in observations must be specified and analysed beforehand
- **univariate endogenous variable**: many conventional models only handle univariate endogenous variable and/or exogenous variable
- **feature engineering:** skillful feature engineering is often necessary and requires domain knowledge of the data of interest

2.2 Machine Learning

The simple group of time domain forecasting models only on information of the variable to be forecast, and not taking into account of the other factors affecting the variation. For example,the Smoothing family of models focus on extrapolating the underlying trend and seasonal patterns of the time series variable itself.

In the last decade or so, of course, software and hardware has brought machine learning techniques to a wider world in both academics and industries. With the sheer power of machines and more data, machine learning has been able to explore beyond the boundaries in data analysis. Methods such as regularised regression and tree models are reinvigorated with more data and computing power. Still, these methods are still limited by the features input, hence domain knowledge and feature selection still a decisive factor in the quality of modeling. Neural net models, or deep learning as known in wider fields, tackle this limitation in feature engineering by shifting the problem to computing power and model complexity. With enough computation and the right model design, the complex, non-linear architecture of deep neutral net models could potentially approximate a wide range of functional form between the input variables and the output variables. Currently, deep learning has very much dominated the world's numerical processing and modeling and has demonstrated its power in industrial applications in sensory capabilities. Of course, deep learning can equally be used to conventional data analysis in social sciences and operation research.

Building on top of the basic idea of matrix multiplication and iterative gradient descent to minimise an objective or loss function, deep learning has attracted colossal amount of investment in talents and funding. Therefore, some milestones in stable models and design such as recurrent neural network (RNN) and its variation Long-Short Term Memory (LSTM) have been understood to be well-established for sequential data modeling - and this suits time-series data naturally well. Recurrent neural networks are a type of neural network that add the explicit handling of order in input observations. This capability suggests that the promise of recurrent neural networks is to learn the temporal context of input sequences in order to make better predictions. That is, that the suite of lagged observations required to make a prediction no longer must be diagnosed and specified as in traditional time series forecasting, or even forecasting with classical neural networks. Instead, the temporal dependence can be learned, and perhaps changes to this dependence can also be learned. Robustness to noise, nonlinearity, and arbitrariness in input variables are properties of high value in time series and forecasting (Dorffner 1996). In addition, neural networks can be configured to support an arbitrary defined but fixed number of inputs and outputs in the mapping function; this enables multivariate input and output and multi-step forecasting, as long as the input and output dimensions are predetermined (in time series this means the lag variations must be preset in the input variables) (Sutskever, Vinyals, and Le 2014). However, this limitation is freed by the advent of LSTM, a special design of RNN able to model sequential data without fixed size windows (Gers, Eck, and Schmidhuber 2002).

In short, deep learning and its subset of RNN/LSTM models sequential variables that require no predetermination of input variables selection or engineering, such as number of lagged variates. Though generally since time-series properties such as trend and seasonality are mature enough to identify and model and thus it is good practice to preprocess the data before modeled by neutral network.

Chapter 3

Methodology

3.1 Model Process

The first task in any model pipeline may be to decide how the project objective ought to be studied through model pipeline in order to evaluate its performance and the quality of results, to have a degree of surety of the conclusion of project objective.

Although the primary objective outlined in the Introduction describes that this project aims to compare various models in predicting product price and/or stock for the goal of optimal supply chain control, the following will define the detail of such prediction task in this paper. In general, forecasting and ultimately its quality depend largely on the definition or scope of the task: other things being equal, forecast on aggregate data is generally more accurate than that is segmented; larger time steps or horizons are harder to predict or have higher variance/uncertainty than smaller ones.

The scope of this paper therefore is defined as the forecast of one time-step of the future value of price and stock, using any data available at current time-step. The reason is mainly for practicality and the objective of model comparison: larger time-steps into the future are intrinsically more difficult and thus less accurate, more time-steps require truncation and discarding of more data observations that would worsen the overall quality, given each entity or product having only 200 or so observations, and one time-step limits the comparative analysis of all models to their potentially best performing that provides clarity in results.

The comparison of model results, based on the following evaluation metrics, are carried out on the average and/or median of scores of all product datasets. Because with the exception of one model (Fixed Effect panel data model), all models is implemented using one set of time series dataset, not cross-sectional. Therefore, in order to establish a common ground for comparison across models while utilising all available data, the results of each model is evaluated on the sample metrics which are summarised as a mean and/or median to represent the forecast quality.

3.2 Evaluation and Metrics

With the model objective defined as the quality of forecasting one future time-step in price and/or stock of a product, the following outlines the decision on how to evaluate the quality of such objective of all models in order to draw logical and reasonable conclusion about the objective.

In the first place, it would be useful to reiterate the distinctiveness of concepts between correlation and causation in the context of forecasting. As all textbooks relating to data analysis alert at some point, the two concepts are sharply different: variable(s) X may well be useful in forecasting variable Y even though it is not true that X cause Y - though X might cause Y but the reverse is also possible, or even more likely that a complex relation exists between them. Causation is no doubt much harder to establish between random variables than correlation, which may exist in any form or shape. Therefore, it would be helpful to find features (exogenous variables or predictors as otherwise known) that have some correlative relationship with the target variables (or endogenous variables) because such information may be good enough to provide quality forecast of target variables. Again, the purpose of the paper is to compare models in their quality of forecast of the target variables instead of the study of underlying generative process.

Therefore, the evaluation metrics to be employed for the results of each model are primarily targeted at forecasting quality - the difference or error between true future values and the predicted values. As the paper solely focuses on forecasting continuous variables of price and stock, the metrics are all applicable to regression model.

Specifically, the standard pipeline of modeling is used:

- Exploratory Data Analysis (EDA) - Since time series models have extra requirement and factors to consider, a series of checks are included: patterns and trend, seasonality or cycle, autocorrelation of target variables and cross-correlation with features, and so on. These exercises would often guide the creation of feature variables that are conducive to the forecasting of target variable.
- Model Fitting and Predicting - Once the dataset are prepared, they are fitted with each model - concretely the dataset are split into train set and test set (some model may be useful to fine tune the hyper-parameters using a validation set that are often part of the train set) with the fitting done on the train set and the prediction on the test set. This framework evidently is critical in establishing confident evaluation of the quality of forecasting future, unseen values.
- Evaluation and Comparison - The final section in the Model Chapter includes the collation and presentation of results from all model according to the evaluation and metrics, which are outlined next. Multiple metrics are employed to provide a general assessment of the quality of results of each model and to give an inclusive comparison.

Regarding the method of prediction, since in the training or fitting the target variables are one time-steps into the future that the prediction is also on one time-step. Due to the small size of per product time series observation, the only a small size of test set will be allocated for evaluation. Further, each time-step of the test set is therefore being predicted by using the previous one time-step feature variables, which are the true values. But since the test set are unseen or not used to fit the model, the resulting test set prediction is still considered as “fair” forecast. Lastly, all of the evaluation metrics are consequently calculated based on the whole of the test set forecast values (or errors thereof).

As mentioned, because the dataset contains two different variables for both price and stock, it is in the interest of clarity that the each model will produce two sets of metrics, one based on the average of the two price variables and the other on the average of the two stock variables. More explanation is provided in the Implementation and Results chapter in the end.

In the following sections, all of the evaluation metrics employed are listed and explained.

3.2.1 Residual and Error

One of the key pieces of information from a model in time series forecasting stems from the residuals from training or the errors in prediction. Both are calculated as the difference between the true target values and the so-called fitted values produced by the model:

$$e_t = y_t - \hat{y}_t$$

where e_t represents either the residual or the error in train set and test set contexts respectively, and $\hat{y}_t$ the fitted or forecast target value accordingly.

First, in regression modeling context, the residual from the fitting process contains useful information regarding the adequacy of model in terms of the information captured in the data. According to the theory of Ordinary Least Square and its condition being Best Linear Unbiased Estimator (BLUE) based on the Gauss-Markov theorem, the quality of regression model capturing the data can be assessed by the residuals of fit, that they are uncorrelated over time and have expected value or mean as zero (Wooldridge 2015). Any so-called autocorrelation or correlation between the residuals over time implies some variations left out of the model. Further more, the condition of the residuals having constant variance and normal distribution are useful properties for uncertainty or variance calculation but are not essential in judging the quality of forecasting. As such, the analysis of residual, and error in test set prediction, is included as a preliminary metric to assess the performance of each model - not so much used as model searching as this would be too large a scope to apply to all the different types of model included in this paper.

3.2.2 Autocorrelation Function (ACF) and Partial ACF (PACF)

ACF is a tool useful in identifying linear relations that may exist within the time series variables, or between its lagged terms over time. The analysis of autocorrelation is an important component in time series analysis and also forecasting largely because of the application of linear regression models in the field. Because of the assumptions of linearity, stationarity and homogeneity of variance over time is critical from both theoretical and practical grounds for linear regression framework, therefore techniques such as ACF is always included in such model analysis pipelines. ACF provides a profile of all possible autocorrelations in the time series variables which are very useful factors in predicting its future values - also important is the structure and variance of such autocorrelation must be regular or having stationarity. In other words, it is necessary condition that a time series autocorrelations be measure with precision at some regular structure for the model to achieve any meaningful statistical analysis; furthermore, such restriction is indeed required only for the predictability of linear relationship and model that possible nonlinear relation may exist (Shumway and Stoffer 2017).

From the function below, it is clear that ACF measures the linear, correlation among a random variable throughout time, or among its lagged values.

$$r_k = \frac{\sum_{t=k+1}^{T} (y_t - \bar{y})(y_{t-k} - \bar{y})}{\sum_{t=1}^{T} (y_t - \bar{y})^2}$$

where r_k is the ACF score measuring the linear relation between y_t and y_{t-k} , T is the number of observations in the variable.

In application, many programming libraries (R, Python) has the so-called `correlogram` that graphs the ACF with confidence intervals assessing the strength of autocorrelation of the variable.

The presence of significant autocorrelation, possibly more many time-steps or lags, is a useful signal to include lagged terms of the variable itself as features for forecasting future values.

Another method for testing autocorrelation is used in analysing residuals and errors in this paper - **Ljung-Box test** (Hyndman and Athanasopoulos 2018):

$$Q^* = T(T+2)\sum_{k=1}^{h}(T-k)^{-1}r_k^2$$

where h is the maximum lag considered.

The test is based on the null hypothesis that the autocorrelations emulates that of a white noise series, in which case the distribution of Q^* would follow a χ^2 distribution with $(h-K)$ degrees of freedom, where K is the number of parameters in the model. Similar to ACF, the presence of large p-value means that there is insignificant evidence of autocorrelation (i.e. fail to reject the null hypothesis that the autocorrelation comes from a white noise series) (Hyndman and Athanasopoulos 2018).

The variation of ACF called Partial ACF is often more useful in essentially identifying the distinct influence of one lagged term to another, discarding the "lingering" effects of previous lags. The formal definition of PACF is as follows: assuming a stationary process of x_t, the PACF, denoted as ϕ_{hh} $(h = 1, 2, ...,$ (Shumway and Stoffer 2017):

$$\phi_{11} = corr(x_{t+1},\ x_t) = \rho(1)$$

$$\phi_{hh} = corr(x_{t+h} - \hat{x_{t+h}},\ x_t - \hat{x_t}),\ h \geq 2$$

where corr is the correlation function; the PACF ϕ_{hh} is thus the correlation between x_{t+h} and x_t removing the lingering effect of previous lags on both.

The main distinction of PACF is that it only reveals the pair effect between consecutive lagged values instead of the full effect carried from the all previous lags as in the case of ACF. As demonstrated in Data Exploration part later, PACF visualisation is much clearer in identifying which lags to use as predictors.

The following four metrics are centred around forecast errors as a measure of forecast accuracy.

3.2.3 MAE and RMSE

Both Mean Absolute Error (MAE) and Root Mean Squared Error (RMSE) are error metrics of the same scale as the data.

$$MAE = \frac{\sum_{i=1}^{T} |y_i - \hat{y}_i|}{T}$$

$$RMSE = \sqrt{\frac{\sum_{i=1}^{T} (y_i - \hat{y}_i)^2}{T}}$$

where $\hat{y}$ is the predicted values at time i and note that the total observation T is the length of the test set.

These two metrics of errors are arguably most common in regression analysis; because the scores are scaled as the data, they are inapplicable for cross data comparison. In this paper, however, the dataset are the same for all models.

3.2.4 MAPE

The Mean Absolute Percentage Error (MAPE) is percentage error and thus unit-free, useful for comparing models across different datasets:

$$MAPE = \frac{\sum_{i=1}^{T} 100 \times |\frac{y_i - \hat{y}_i}{y_t}|}{T}$$

One pitfall of MAPE is that when any y_t is zero the score is undefined. However in this project this is not the case and the metric is included.

3.2.5 R^2

This measures the goodness of fit of the regression model, also known as the coefficient of determination of the model:

$$R^2 = 1 - \frac{\sum_{i=1}^{T}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{T}(y_i - \bar{y})^2}$$

where $\hat{y}_i$ and $\bar{y}$ are respectively the predicted value at time i and mean of true values. The numerator is referred as Residual Sum of Squares (SSE) and the denominator as Explained Sum of Squares (SST), hence the R^2 effectively measures the share of variance captured by the model.

The validity of R^2 is questionable in most contexts where the objective focuses on finding the underlying model describing the data, as more feature variables will always increase the score. In this paper, however, the objective is to compare the forecast quality among models with identical features, therefore R^2 can still be included as a metric to compare as opposed to measure model quality.

3.2.6 AIC and BIC

The Akaike's Information Criterion (AIC) and the Bayesian Information Criterion (BIC) are two metrics often used in forecasting or prediction analysis for assessing the model fit of the data.

$$AIC = T\ log(\frac{SSE}{T}) + 2(k+2)$$

$$BIC = T\ log(\frac{SSE}{T}) + log(T)(k+2)$$

where k is the number of features or predictors in the model.

The general idea of AIC can be seen from its equation that the residual errors is penalised as k increases and thus regularises the complexity of model. When used in comparing across models, the minimum value of the AIC is commonly understood as the best fit model for the data. BIC is very similar to AIC except that the penalisation of features or predictors is stronger with the multiplicative term of T.

3.3 Models

This section lists all the models included for the purpose of comparing forecasting quality on the product dataset - price and stock variables in particular. The list is naturally not nearly as comprehensive, as that would be too large a scope to target in this paper, but the models are selected as that a broadly diverse methods are compared.

The following table shows the included models and their commonly referred classes

Table 3.1: Models Implemented

AR	Linear	Generative State Space	Neural Nets
ARIMA	OLS	Structural Time Series (STS)	LSTM
	Fixed Effect Panel		
	Elastic Net		

There are many model families not included in this paper, such as those of Exponential Smoothing with multitudinous forms. The models included in the table span an arguably wide range of model frameworks from linear regression to neural nets. The following explanation about each model broadly illustrate the basic formulation of the model and how it is used in practice for forecasting purposes. The exploration and analysis of data transformation, such as number of lagged terms of target and feature variables to include are included in the Data Preparation Chapter.

Applicable to all of the models included in this paper is the concept of various patterns that are well studied in time series analysis. Regarding the time series variable itself, there are three main types of such patterns: trend, seasonality and cycle. *Trend* is defined as the long-term rise or fall over time that it can be either linear or otherwise - swinging to the oppose direction over a long time horizon. *Seasonality* is commonly referred to against the calendar, in theory a fixed recurrence over the time units of hour, day, month, quarter or year. *Cycle* is a similar recurrence to seasonality except that the frequency is not fixed with longer duration and varying magnitude (Hyndman and Athanasopoulos 2018). It is very common in real world data to have a combination of the three patterns exist in the time series variables; consequently the identification of them will significantly help building the right model with accurate prediction.

3.3.1 Ordinary Least Squares (OLS)

Ordinary Least Squares (OLS) is definitely a time-tested model framework for any data analysis, perhaps the most utilised framework across the disciplines, especially in social science. As mentioned before, many analysis requires the valid conditions in order to justify that the fit of the model is meaningful. Although not all of them are required in this paper, as the objective is forecast accuracy instead of finding explanatory predictors or establishing meaningful relationships between the variables, some properties are useful to mention for the benefit of analysis. Also, there are conditions such as BLUE for the model to be unbiased with minimal variance. These are useful guidelines but are not necessary for applying the model in forecasting, concretely that the use of unfounded predictors or features are of practical needs. Not to mention that linearity in composition of the features to explain the target in real dataset such as retail product is hardly present, more complex relationship and, even more likely, no relationship could exist between the variables.

Linear regression model in general is defined as that the target variable y_t at time t is the result of the linear combination of m feature variables $x_{1t}, x_{2t}, ...x_{mt}$. Such relationship has the mathematical form:

$$y_t = \beta_1 x_{1t} + \beta_2 x_{2t} + ... + \beta_m x_{mt} + e_t$$

where $\beta_1, \beta_2, ...\beta_m$ are unknown coefficients or parameters and e_t the random error or noise that is independently and identically distributed (iid) following a normal distribution with mean zero and variance σ^2.

The above formulation is oriented towards time series analysis with the observation being timestamps, as opposed to normal cross-section case.

One of the main restriction of linear regression or OLS is the error term e_t being *iid* with standard normal distribution. As explained in later models such as ARIMA, time series variables commonly has autocorrelation with itself over time, therefore unless such lagged terms are captured as part of the features the model fails to explain that part of the variation(Shumway and Stoffer 2017). An analysis of the error term (or residuals in the train set) can reveal these informations. In addition to autocorrelation, the presence of changing variance (non-stationarity) and trend also renders linear regression less effective in forecast accuracy as the linear relationship fails to capture such dynamics.

OLS to in some respect is a simple linear model but having the need to fine tune not any hyper-parameters, such as that in more complex models like the neural nets, but the choice and transformation of features. This process of finding better model form is often carried out through Analysis of Variance (ANOVA) including metrics such as Mean Square Error (MSE), R^2, AIC, BIC, and so on. However, these are the methods aimed at building a meaningful explanatory regression model rather than the narrowly focused forecasting accuracy, which as mentioned before does not necessarily require statistical validity in modeling features and target variables relationship.

3.3.2 Elastic Net

Despite the peculiar name, Elastic Net is in fact part of the linear regression family of models that is also known as Shrinkage Methods: the idea is similar to the classic process of selecting best features or predictors for minimising the prediction error, but the Shrinkage methods are continuous instead of discrete and thus result in lower variability (Hastie et al. 2005).

Elastic Net is simply the weighted combination of two of the more popular Shrinkage Methods known as Ridge and Lasso regressions. The essential elements of Elastic Net, and therefore Ridge and Lasso, is the additional "shrinkage" or regularisation term added to a OLS. The idea is that in linear regression model, many feature variables are correlated and thus having poorly fitted parameters with high variance, and by imposing constraint on the parameters such problems are mitigated (Hastie et al. 2005). The following formula explain this idea from the perspective of fitting OLS parameters or coefficients via the finding the least squared errors:

$$\hat{\beta}_{ridge} = \underset{\beta}{argmin}\{\sum_{i=1}^{N}(y_i - \beta_o - \sum_{j=1}^{p} x_{ij}\beta_j)^2 + \lambda \sum_{j=1}^{p} \beta_j^2\}$$

*where i is the ith observation of a total of N (in the context of this paper the i represent the t observation in the time series dataset for each product) and j the jth feature/predictor parameter/coefficient (p features); the shrinkage term defining Ridge regression penalises the sums of errors with increasing parameters included in the model, with the strength of such penalisation controlled by the multiplicative term λ.

$$\hat{\beta}_{lasso} = \underset{\beta}{argmin}\{\frac{1}{2}\sum_{i=1}^{N}(y_i - \beta_o - \sum_{j=1}^{p} x_{ij}\beta_j)^2 + \lambda \sum_{j=1}^{p} |\beta_j|\}$$

instead of proportionally shrinking parameters, lasso penalises the parameters by λ and truncates them to zero

Focusing on the additional shrinkage term, Elastic Net is defined as:

$$\lambda \sum_{j=1}^{p}(\alpha\beta_j^2 + (1-\alpha)|\beta_j|)$$

where α is between 0 and 1 as weight to control the two different shrinkage terms

In simple terms, Elastic Net combines the benefits of both Ridge and Lasso, able to shrinkage wildly correlated parameters proportionally while truncating parameters to zero as a model selection feature.

In application, the only hyper-parameters, if feature engineering not counted, is α, which is often optimised in the testing on a validation dataset by searching various values.

In sum, Elastic Net is a powerful linear regression model in situations where the number of features are large and therefore there exists high chance of multicollinearity among the features that would otherwise impeding the model quality if not treated. However, such feature is arguably less useful for the dataset in this paper. Nevertheless, Elastic Net is included as an improved linear regression model and for the reason of its common applicability in production.

3.3.3 Fixed Effect Panel Data Model

Fixed Effect linear regression is a well-established method with popularity in certain fields regularly applying panel data, such as in econometrics where panel dataset have many cross-sectional entities - firms, consumers, or other subject or panel member - observed over time. The model assumes a constant, additive differences among the entities over time - α_i - which in effect is implemented as a dummy variable of the entities. As such, the model regresses the endogenous variable by the exogenous variable using observations over time to obtain β of each exogenous variables, accounting for entities. Therefore, Fixed Effect regression in this form only captures the linear relationship and can be simplistic and inadequate for arbitrary variables such as product features.

The Fixed Effect (FE) model has the form:

$$y_{it} = x_{it}\beta + \alpha_i + \varepsilon_{it}$$

where i indexes the entities - where each entity is the subject or panel member of the dataset, t time. All in vector form, β represents the parameters coefficients and α the time-invariant, entity-specific components unobserved in the data. Finally, ε is the idiosyncratic errors uncorrelated with α and the covariates exogenous the vector of variables.

The additional parameter α_i allows the OLS model to include an entity-specific effect, though invariant over time, arbitrarily correlated with x_{it} explicitly. As such the FE model requires the assumption that the entity effect term α_i to be uncorrelated with the error term. Another critical assumption in FE model, in order to keeping the robustness, is that the set of features of x_{it} cannot include any time-invariant or constant variables that are observable, as the assumption that α_t being the time-invariant unobservable variables would not be distinguishable (Wooldridge 2010). This restriction is not an issue in this paper because none of the features are time-invariant.

Because of the natural capability of FE model being applicable directly with panel data, unlike other models included in this paper, the fit and prediction is carried out directly using the dataset at hand. In other words, the evaluation on the forecast results are averaged into mean and/or median over the available product datasets, as they are fitted and tested one product per run, whereas the FE model is fitted and tested once using all of the product datasets as a single panel dataset.

3.3.4 ARIMA

The model named Autoregressive Integrated Moving Average (ARIMA) is a significant time series and forecasting model made popular by the famous Box-Jenkins method that identifies the structure of ARIMA model and the surrounding techniques of parameter estimation and forecasting (Shumway and Stoffer 2017). As the name suggests, the model is a larger framework combining components of autoregressive term (AR), moving average (MA) with treatment of non-stationarity (I).

The special characteristics of ARIMA framework is the multiplicative nature of factors involving differential and differencing equation operators modeling a white noise input (Shumway and Stoffer 2017).

The AR component of ARIMA stands for autoregressive model, which forecast based on a linear combination of lagged terms:

$$y_t = c + \phi_1 y_{t-1} + \phi_2 y_{t-2} + ... + \phi_p y_{t-p} + \varepsilon_t$$

where ε_t is white noise, c a constant

The format is a special type of multivariate linear regression with all features or predictors as the lagged terms of the target variable itself. The notation for this type of model is **AR(p)**, known as an autoregressive model of order p. AR model is very flexible that changing the parameter p and c can effectively model a variety of time series distributions, such as white noise, random walk, etc.

The second component MA is refer to as the **MA(q) model** with a moving average of order q. Differing to the use of lagged terms in AR, the MA model uses past forecast errors in the regression setup:

$$y_t = c + \varepsilon_t + \theta_1 \varepsilon_{t-1} + \theta_2 \varepsilon_{t-2} + ... + \theta_q \varepsilon_{t-q}$$

where ε_t is white noise that are unobserved values

The idea is that each time-step value of y_t can be seen as the weighted moving average of past few forecast errors (Shumway and Stoffer 2017).

The final piece in ARIMA relates to differencing, or in fact the reverse of it as "integration".

Differencing is a method used in order to remove non-stationarity of a time series variable. Differenced series is the change between consecutive observations such as:

$$y_t^{'} = y_t - y_{t-1}$$

The full model of ARIMA thus is in the form of:

$$y_t^{'} = c + \phi_1 y_{t-1}^{'} + ... + \phi_p y_{t-p}^{'} + \theta_1 \varepsilon_{t-1} + ... + \theta_q \varepsilon_{t-q} + \varepsilon_t$$

with the features the combination of both lagged values and lagged errors. This model is called **ARIMA(p, d, q) model** having p order of autoregressive features, d degree of first differencing and q order of moving average features.

One difficulty in applying ARIMA model clearly lies in the finding of the optimal parameters of p, d, q as there is no natural setting for these values. Fortunately, in many data analysis packages such as R an auto-search function is included.

3.3.5 Structural Time Series (STS)

Structural Time Series model is framework of modeling time series data that employs another framework known as State-Space model. The idea in of state-space model framework, in the case of time series modeling, is to create a generative model that describe the data by many latent processes, all together capture different signals, which is that integrated to compute the posterior predictive of the time series of interest - the name Structural Time Series refers to the nature of the particular type of state-space model that uses linear-Gaussian distribution and thus the main time series is modelled as additive sum of various signal processes (Murphy 2012). For example, there may be an underlying trend, a seasonality or cycles, and/or influence from other time series variables.

In its general form, the model is a decomposition of additive effects on the given time series variables:

$$f(t) = f_1(t) + f_2(t) + \cdots + f_n(t) + \varepsilon$$

$$\varepsilon \sim N(0, \sigma^2)$$

where $f(t)$ is the target time series distribution of interest that is modelled as a linear combination of other exogenous times series distributions, represented by $f_1, f_2, ...f_n$, and a white noise ε.

The exact content and functional form of each component are not delineated in the general form and are flexible enough to be modelled by the context of problems and nature of datasets. In the following, some of the most widely used and well-established effects and components are briefly described.

For example, the simplest latent process is known in the literature as **local level model**, which has the form:

$$y_t = a_t + \varepsilon_t^y,\ \varepsilon_t^y \sim N(0,\ R)$$

$$a_t = a_{t-1} + \epsilon_t^a,\ \varepsilon_t^a \sim N(0,\ Q)$$

where a_t is the the latent state; the model presumes that the observed data y_t is a linear sum of some hidden **level** term a_t and a Gaussian noise with 0 mean and variance R, and that a_t autocorrelate over time with another Gaussian noise with 0 mean and variance Q.

Below is a list of the main components in a typical STS each modeling a distinct effect influencing the endogenous time series (Dillon et al. 2017). This is not an exhaustive list of possible components employable for modeling time series to predict and forecast, but only some of the exemplary effects commonly used.

Table 3.2: STS Components

Component Model	Description
Autoregressive (AR) process	Model of latent level with steps of value modelled as linear combination of previous steps; in particular any first-order AR process with `alpha` coefficient < 1 are stationary or maintaining constant variance over time

Component Model	Description
Local Linear Trend	Similar to Linear Level model, this process has an additional `slope` process that drives the `level` process by another "`level`" term with the main process of f_t as: $$f_t = level_t + Gaussian(0, scale_{noise}),$$ level process: $$level_t = level_{t-1} + slope_{t-1} + Gaussian(0, scale_{level}),$$ and slope process: $$slope_t = slope_{t-1} + Gaussian(0, scale_{slope})$$
Seasonality	If any fixed set of recurring or regular pattern over fixed amount of timestamps, then it is said to have a seasonal effect: $effect_{season,\ occurrence_i} = effect_{season,\ occurrence_{i-1}} + Gaussian(0, scale)$
Covariates Regression	Linear combination of other covariate time series with weights as random variables inferred approximately by Hamiltonian Markov Chain or Variational Inference as the other model parameters

The general logical flow in the implement of Tensorflow Probability STS models the time series as a form of dynamic linear system whose parameters are learnt using Kalman filter algorithm and variational inference technique for the posterior distribution approximation.

3.3.6 Neural Nets

The final model to be included in this paper comes from the field of Neural Nets or commonly known as Deep Learning (Goodfellow, Bengio, and Courville 2016). In basic terms, Neural Nets model framework is based on a series of data transformation that maps the input data to the output in a flexible and non-linear fashion. As briefly described in Introduction, Neural Nets is at its core a sequential transformation of numerical data, hence its ability to approximate arbitrary mapping from input to output, in theory at least. One distinct pitfall of neutral network and the wider deep learning framework of machine learning stems from the so-called black box of relationship between model architectural design and overall performance, exacerbated by the computing demand in running model plus searching for optimality.

From the perspective of the architectural components, Neural Nets is deeply modular: each step in the transformation - the layers - can be replaced by multiple choices of functions and methods. Therefore, the complexity is extremely high from the perspective of the number of choices of component design and the number of parameters to be estimated in the entire model, with exponential increase as the nodes and layers increases. Therefore, the common wisdom in using Neural Nets is that a huge of data is required to exploit its power brought by complexity - which other models may fail to process and explore more complex relationship.

The particular Neural Nets architecture implemented in this paper is called Long-Short Term Memory (LSTM) which is classified as a form of Recurrent Neural Network (RNN) that is capable of processing sequential data such as time series where there is dependence over time (Goodfellow, Bengio, and Courville 2016). LSTM in basic terms allows the optimisation of long ranging sequential input processing by sharing and crucially adjusting the parameters or weights in the model (Goodfellow, Bengio, and Courville 2016). This in practical terms distinguishes LSTM from other Neural Nets in that it can be effective in learning long term dependence from sequential input without much of the pitfalls such as vanishing gradients and many other training issues.

A simple and intuitive way to illustrate the logic of neural nets in the basic construction is to view it in the angle from linear regression and matrix multiplication, two of the basis of data analysis.

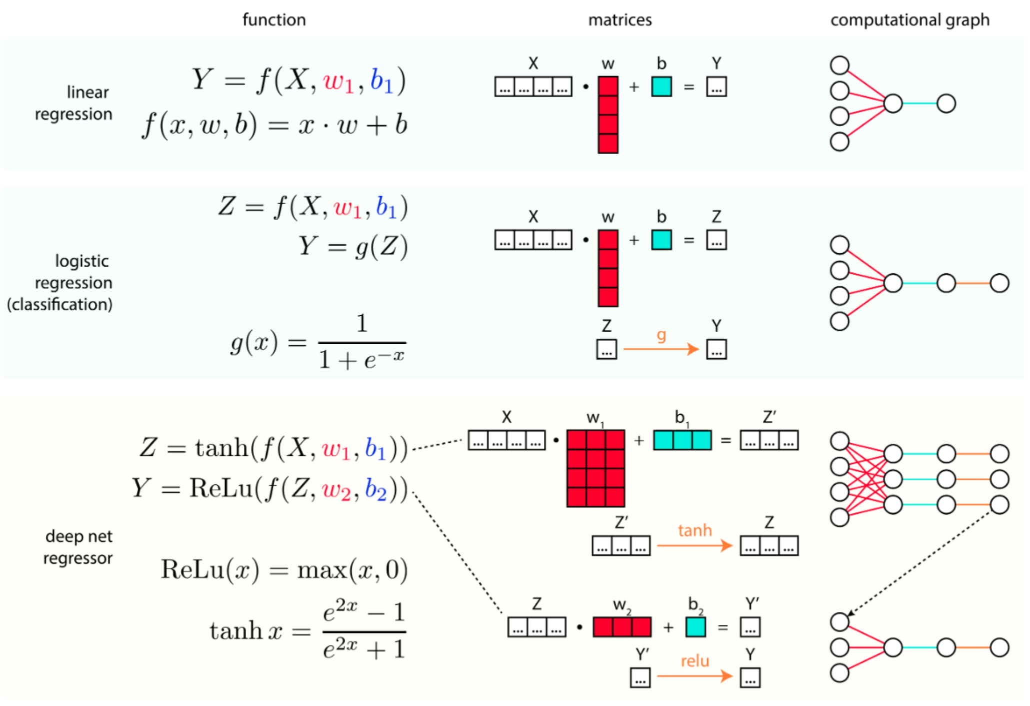

function
matrices
computational graph
linear regression
$Y = f(X, w_1, b_1)$
$f(x, w, b) = x \cdot w + b$
X
w
b
Y
logistic regression (classification)
$Z = f(X, w_1, b_1)$
$Y = g(Z)$
$g(x) = \frac{1}{1+e^{-x}}$
Z
g
deep net regressor
$Z = \tanh(f(X, w_1, b_1))$
$Y = \mathrm{ReLu}(f(Z, w_2, b_2))$
$\mathrm{ReLu}(x) = \max(x, 0)$
$\tanh x = \frac{e^{2x}-1}{e^{2x}+1}$
w_1
b_1
Z'
tanh
w_2
b_2
Y'
relu

Figure 3.1: Neural Nets Illustration

Note: Y represents the final output or target to be predicted, X the input tensor (vector or multidimensional array), W the weight/parameter matrix and b the the constant term.

Source: Presentation by Eric Ma at PyData NYC 2017 (https://ericmjl.github.io/bayesian-deep-learning-demystified/#/IntroductionSlide)

This illustration in Figure 3.1 by Eric Ma shows visually the relationship between neural nets and linear and logistic regression. The insight is that the classic regressions like OLS can be viewed differently from both matrix multiplication point of view (**matrices** in Figure 3.1) and as graph (**computational graph** in Figure 3.1). By the same logic, a basic neural nets or **Multilayer Perceptron** (as it formally is known) is essentially a *wider chain or series* of matrix multiplications. Although there are computational specialties like *activation function* that prevents the exponential growth of the values, the basic ideas of neutral nets are very similar to regression and matrix algebra - which interestingly provides the geometric transformation view of learning.

As mentioned, the neural nets architecture implemented in this paper belongs to the subset of deep learning called Recurrent Neural Network (RNN), specifically the Long-Short Term Memory (LSTM) model.

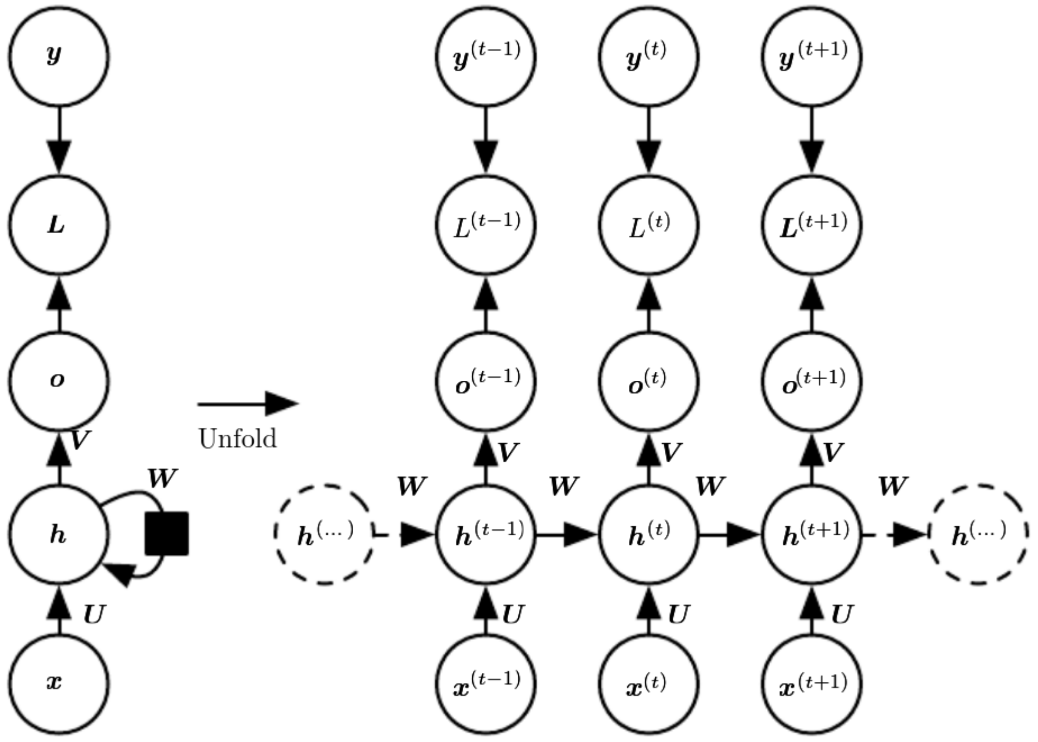


Figure 3.2: Basic RNN Illustration

Source: https://www.deeplearningbook.org/contents/rnn.html

Figure 3.2 illustrates the logic of one of the simplest RNN design. In this graph, x is the input node, h the “hidden” node(s), o the output node, L the loss function and finally y the true target value to be learned or predicted. The parameters to be learned is represented by U (weight matrix transforming x to h), and W is the weight matrix transforming the h to its own temporal nodes (illustrated in the right hand side as unfolded graph) and V the weight matrix transforming h to output o node. The left graph is an abstraction similar to the illustration of chained matrix multiplication in Figure 3.1. The difference that distinguishes RNN is the unfolded graph in the left of Figure 3.2. Since the input vector x is sequentially dependent, such as in the case of text, audio and time series data, the neural nets is designed to learn the dependence parameters between the input sequences. This is realised by W , the weight matrix that "recurs" over the sequences by sharing the weights in the next input sequence and so on. To be sure, there are a myriad of designs on the architecture of W and the whole neutral nets for both computational and theoretical goals.

Due to many pitfalls of the basic RNN design, one of the most well known is called *vanishing gradient* whereby the weight matrix W tends to approach zeros as the model is fed with longer sequences caused by the gradient decent method in optimising neural nets, a special RNN design is created to essentially learn through the model what information is kept and what is to be discarded. This type of model is called Long-Short-Term-Memory (LSTM) (Hochreiter and Schmidhuber 1997).

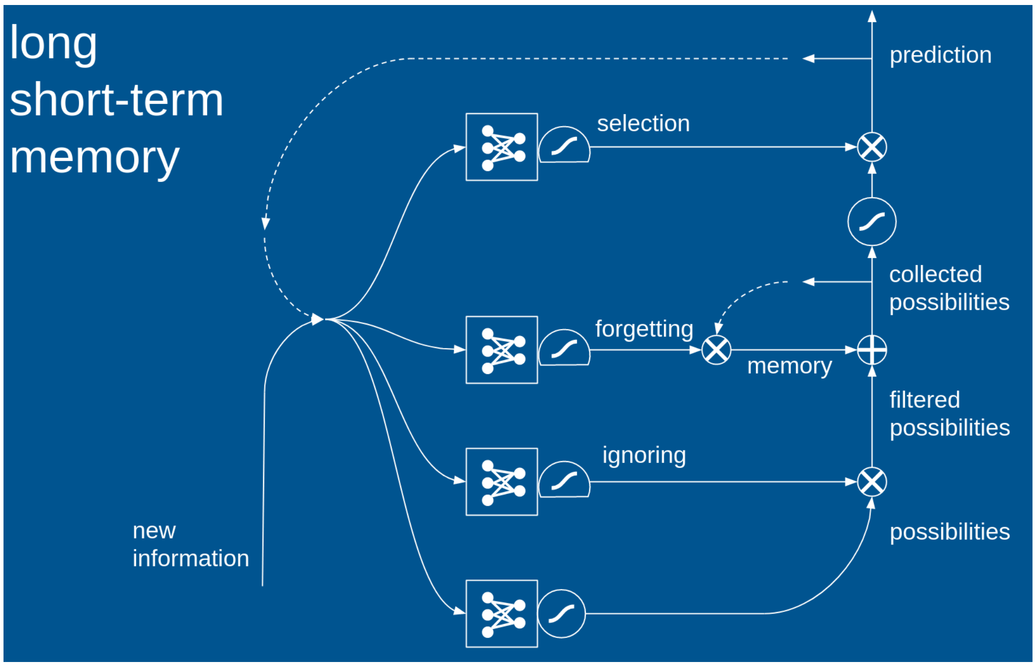


Figure 3.3: LSTM Illustration

Source: https://brohrer.github.io/blog.html

The mechanism of LSTM can be intuitively seen from this wonderful illustration by Brandon Rohrer in Figure 3.3. In the passing of each input vector, LSTM introduces several "gated" devices that control what is kept in "memory" and what is ignored, forgotten, and/or selected for the next sequence. The power of LSTM lies in the learning of these devices through fitting the input to the target values. All the needed parameters with various hidden nodes are automatically learned along with the other weight matrices similar to the basic neural nets.

Without going into the mathematical details of Neural Nets and LSTM, as that would require the many more complementary concepts, it is of the scope of this paper to provide the brief conceptual explanation as the above and also demonstrate how the model is applied for this project. It is implemented using an open source library called Keras (Chollet and others 2015), a popular Python library famous for its ease of use that enable practictionners to flexibly build the model with modular components that are the best practice of the latest researches in Deep Learning and hide the complexity of underlying mathematical treatment and the computational complication.

Chapter 4

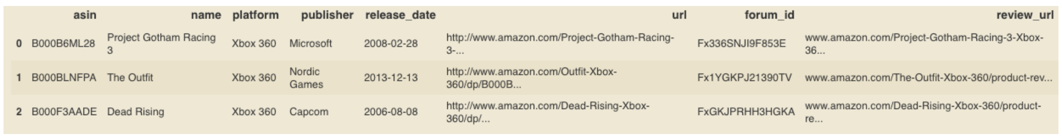
Data Preparation

This chapter of the paper consists of several parts that are oriented to prepare the dataset for implementation in the following chapter. Starting with an overview of the dataset available for this project, which is targeted at finding useful applications in forecasting product information such as price and stock movement for the goal of better supply chain optimisation. The second part of this chapter goes through several data cleaning and preprocessing steps that are reasonably needed to effectively transform the raw dataset into the format that is suitable for the models to fit and train. Once appropriately preprocessed, the final part involves a series of data exploratory techniques that are useful to have a better understanding of the data and conducive to the implementation of models, such as the ACF technique on the target time series variable for creating the right types of feature variables from lagged values.

4.1 Data Overview

Below is a head overview of available tables of the dataset of interest. The raw dataset spans several tables each containing some aspects of the product information extracted from the online marketplace. Like most real world applications, the dataset are unstructured and unorganised.

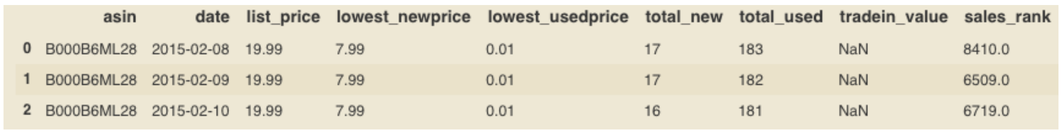
Dataset 1: Product (Amazon)



	asin	name	platform	publisher	release_date	url	forum_id	review_url
0	B000B6ML28	Project Gotham Racing 3	Xbox 360	Microsoft	2008-02-28	http://www.amazon.com/Project-Gotham-Racing-3-...	Fx336SNJI9F853E	www.amazon.com/Project-Gotham-Racing-3-Xbox-36...
1	B000BLNFPA	The Outfit	Xbox 360	Nordic Games	2013-12-13	http://www.amazon.com/Outfit-Xbox-360/dp/B000B...	Fx1YGKPJ21390TV	www.amazon.com/The-Outfit-Xbox-360/product-rev...
2	B000F3AADE	Dead Rising	Xbox 360	Capcom	2006-08-08	http://www.amazon.com/Dead-Rising-Xbox-360/dp/...	FxGKJPRHH3HGKA	www.amazon.com/Dead-Rising-Xbox-360/product-re...

This table contains the basic product information of 1945 video games on Amazon as shown.

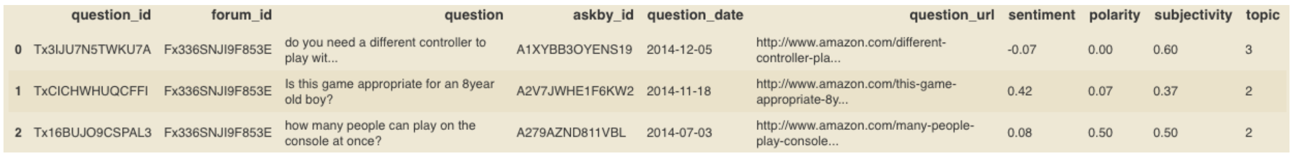
Dataset 2: Product Info (Amazon)



	asin	date	list_price	lowest_newprice	lowest_usedprice	total_new	total_used	tradein_value	sales_rank
0	B000B6ML28	2015-02-08	19.99	7.99	0.01	17	183	NaN	8410.0
1	B000B6ML28	2015-02-09	19.99	7.99	0.01	17	182	NaN	6509.0
2	B000B6ML28	2015-02-10	19.99	7.99	0.01	16	181	NaN	6719.0

Time series data on prices, inventory and sales rank per product on Amazon - 229 daily data per product.

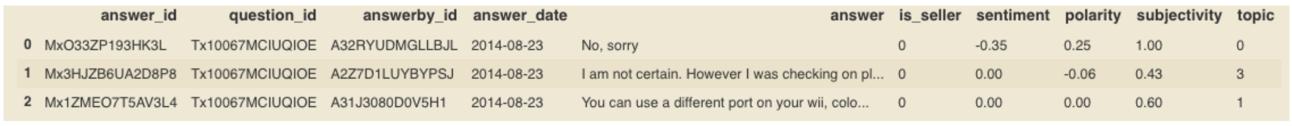
Dataset 3: Product Question (Amazon)



	question_id	forum_id	question	askby_id	question_date	question_url	sentiment	polarity	subjectivity	topic
0	Tx3IJU7N5TWKU7A	Fx336SNJI9F853E	do you need a different controller to play wit...	A1XYBB3OYENS19	2014-12-05	http://www.amazon.com/different-controller-pla...	-0.07	0.00	0.60	3
1	TxCICHWHUQCFFI	Fx336SNJI9F853E	Is this game appropriate for an 8year old boy?	A2V7JWHE1F6KW2	2014-11-18	http://www.amazon.com/this-game-appropriate-8y...	0.42	0.07	0.37	2
2	Tx16BUJO9CSPAL3	Fx336SNJI9F853E	how many people can play on the console at once?	A279AZND811VBL	2014-07-03	http://www.amazon.com/many-people-play-console...	0.08	0.50	0.50	2

Amazon product questions associated with each product

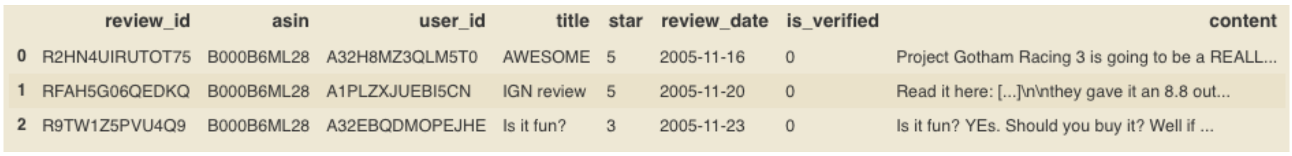
Table 4: Product Answer (Amazon)



	answer_id	question_id	answerby_id	answer_date	answer	is_seller	sentiment	polarity	subjectivity	topic
0	MxO33ZP193HK3L	Tx10067MCIUQIOE	A32RYUDMGLLBJL	2014-08-23	No, sorry	0	-0.35	0.25	1.00	0
1	Mx3HJZB6UA2D8P8	Tx10067MCIUQIOE	A2Z7D1LUYBYPSJ	2014-08-23	I am not certain. However I was checking on pl...	0	0.00	-0.06	0.43	3
2	Mx1ZMEO7T5AV3L4	Tx10067MCIUQIOE	A31J3080D0V5H1	2014-08-23	You can use a different port on your wii, colo...	0	0.00	0.00	0.60	1

Amazon product answers logged per question associated with each product

Dataset 5: Product Review (Amazon)



	review_id	asin	user_id	title	star	review_date	is_verified	content
0	R2HN4UIRUTOT75	B000B6ML28	A32H8MZ3QLM5T0	AWESOME	5	2005-11-16	0	Project Gotham Racing 3 is going to be a REALL...
1	RFAH5G06QEDKQ	B000B6ML28	A1PLZXJUEBI5CN	IGN review	5	2005-11-20	0	Read it here: [...]\n\nthey gave it an 8.8 out...
2	R9TW1Z5PVU4Q9	B000B6ML28	A32EBQDMOPEJHE	Is it fun?	3	2005-11-23	0	Is it fun? YEs. Should you buy it? Well if ...

List of reviews logged per product on Amazon

The dataset in essence consists of 1945 video games listed on Amazon marketplace online each having several time series data. The dataset is separated into different tables according to the types of information, such as price and stock, consumer question and answers, consumer reviews, and other meta data like brand, categories, etc. It is clear from these tables that some data is not suitable for the purpose of time series forecasting implemented in this paper. For example, the information on individual product types, such as `platform, publisher, published date` and so on are not applicable to be included as feature variables.

As with all real world dataset, the raw dataset such as these in table format need to be concatenated or joined together into a so-called dataframe or a single dataset to be processed by models. There are also several issues need to be treated such as missing values and mis-alignment of time-step values. The following part illustrates the steps and decisions in preprocessing these tables of data into a model ready format.

4.2 Preprocessing Data

From the previous list of tables, it seems reasonable to first join the various tables into a single dataset before further preprocessing is taken. The detail of concatenation is not shown but it suffices to give the final list of variables available after the combination - essentially the various tables are related by different keys such as *product_id, forum_id, question_id* and so on.

The table below lists out all of the variables in the dataset that are included in this paper to fit the models for comparison

Table 4.6: Variables Included

Variables	Explanation
lowest_newprice	The price of new product sold; vary over time and included as one the main target variables.
lowest_usedprice	The price of used product sold; vary over time and included as one the main target variables.
total_new	The stock level of new product available; vary over time and included as one the main target variables.
total_used	The stock level of used product available; vary over time and included as one the main target variables.
sales_rank	The tracked and categorical sales rank index associated with each product on the Amazon marketplace online as a rough indicator of the sales level for the product category rather than the product itself; a useful variation to include as a proxy for demand.
sentiment	The preprocessed sentiment value based on the answers provided by consumers in response to the questions asked by consumers associated with each product; this acts as a general indicator of reception of the product and therefore a potential feature to include.
polarity	Computed together with *sentiment* score on the consuer answers, this score is an indicator between [-1, 1] with 1 meaning positive statement and -1 negative statement.
subjectivity	Computed together with *sentiment* score on the consumer answers, this score is an indicator between [0, 1] with 1 meaning full subjective of the language.
star	This is the score submitted along with consumer comments to the product and therefore it contains helpful information about the reception; it is included as a feature to influence demand indirectly.

The variables listed in the above table are all the final features to be included in the models for their potential contribution to influence the target variables - specifically the first four variables *lowest_newprice, lowest_usedprice, total_new, total_use* as these are the price and stock variables that this project aims to forecast (henceforth the two price variables are referred to as *newprice, usedprice* and the stock variables as *newstock, usedstock* for clarity and simplicity). However, as the data exploration will later show, there is no distinct correlation between the four target variables, and therefore they are also included as features - this is a reasonable assumption as the information on either used and new products will influence the consumer demand of the two.

Each product spans a maximum 229 daily observations in price and stock variables. However, the other supplementary features such as `ranking, sentiment and review star` contain only sporadic observations and entries, which also only span over varied time periods.

The first thing to note is that this dataset only contains a very small size of features from the perspective of a typical machine learning model. This may well influence the resulting forecasting quality of the more complex models, but the full comparative analysis is at the end.

Once the data is joined as one, the next logical step is to examine the quality of the data, firstly the integrity of observations of the variables across the products or entities. Concretely this requires that the size of observations in the four target variables to be complete or largest - other variables is dealt with after sifting the dataset of poor quality product data.

Figure 4.1 shows a simple count of two variables `newprice, star` shows that most of the products contain daily observations in `newprice` above 200, or close to the maximum available in this dataset - the results are similar for the other three target variables; however, the distribution of observations in other features such as `star` is somewhat distorted: Most of the products contain 100 or less observations, followed by sudden drop and lingering till 1000. This is evident in the following data preprocessing where features such as `sentiment, polarity,`

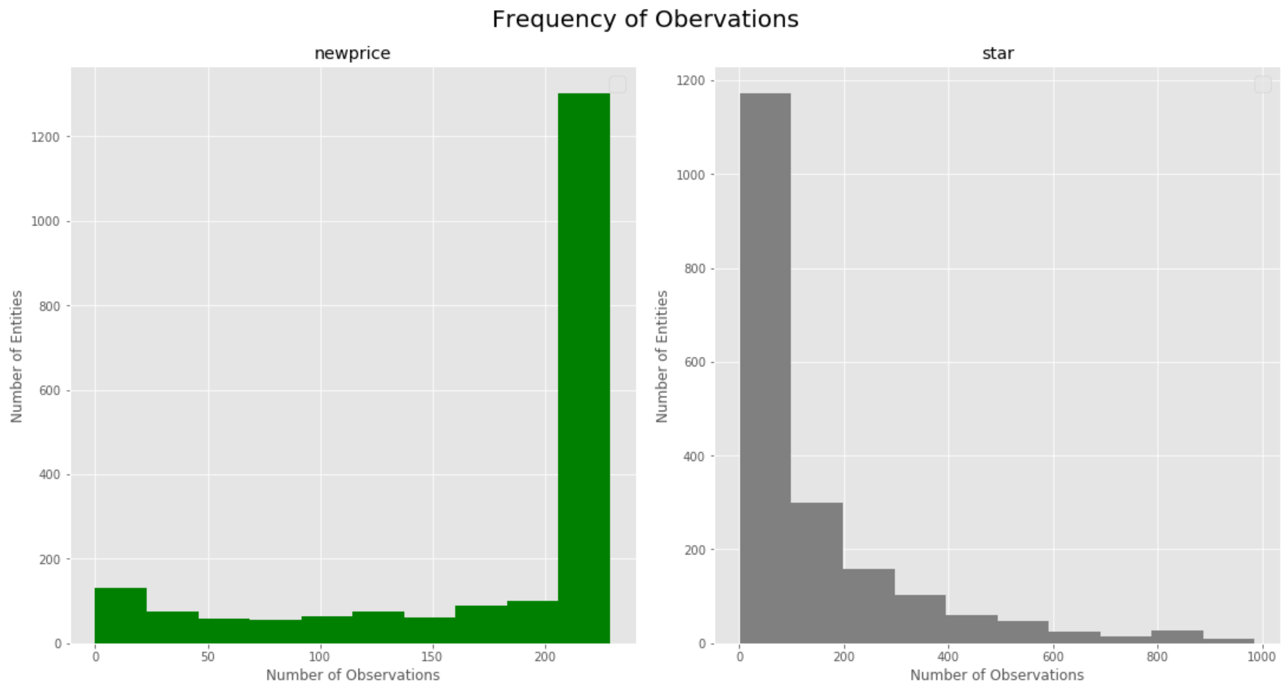


Figure 4.1: Feature Observation Count

`subjectivity, star` are not only wildly distributed in observation count but also mis-aligned with the target variables in time-step - i.e. most of the observations are concentrated within a few months or days in the beginning of the time series.*

The next step naturally is to filter the dataset on the number of observations in both the target and feature variables in order to have as much observations as possible for modeling. The focus is on keeping those product datasets having an adequate amount of observations in features `star, sentiment`: the resulting 50 product datasets have a minimum 74 observations in the two variables. Clearly from this step that some imputation and interpolation are needed.

The graph in Figure 4.2 is based on a single product dataset that represent the typical missing value situation for variables included. Specifically, there are mainly two issues which need to be addressed: the low-value ranges variables of `sentiment, polarity, subjectivity, star` have concentrated observations in the early time-steps relative to the rest of the price and stock variables, often tens during a single day; the target variables consequently begin only after certain time-step behind the other feature variables. The upper graph shows the

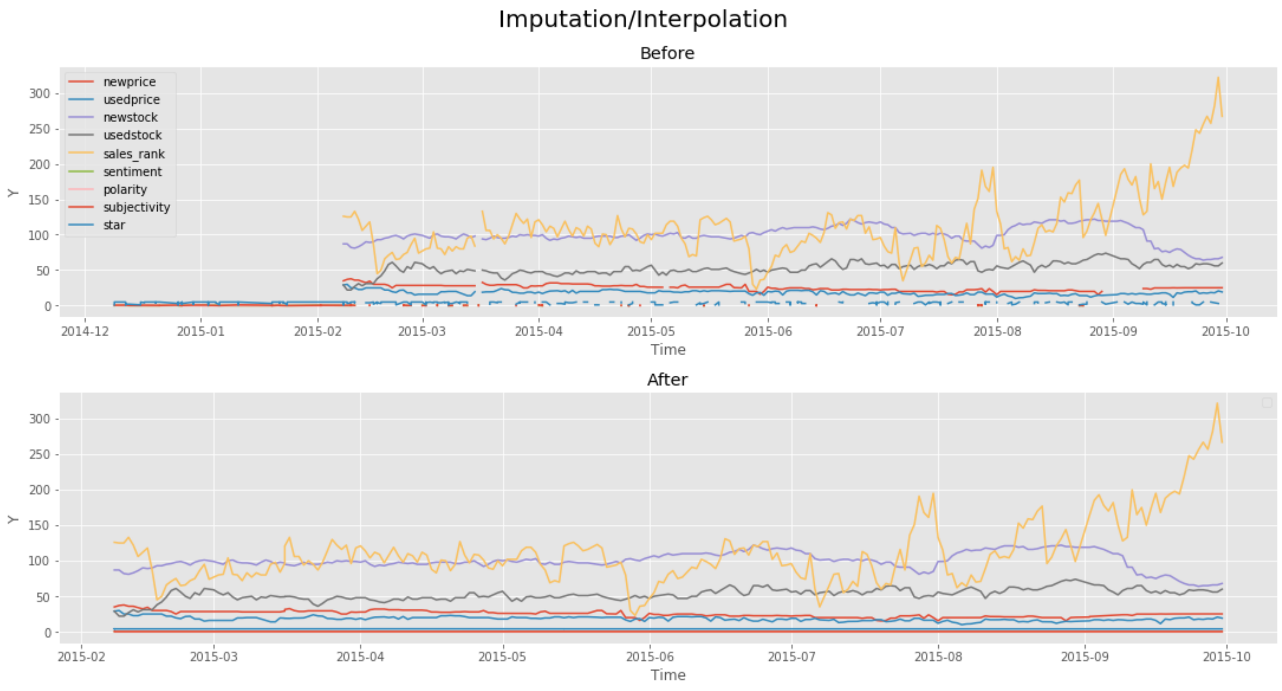


Figure 4.2: Imputation and Interpolation

distribution of data before the preprocessing and the bottom graph shows the after effect. This also illustrate the areas of preprocessing required for this particular dataset. Two main techniques are used to address the issues: imputation on the missing values of features variables by cumulatively expanding the time series with the average of two consecutive observations or the moving average smoothing of window size 2; then truncate the resulting dataset keeping only the time-steps of target variables before interpolating any missing gap with linear trend, as there are some missing time-step in the target variables. Imputation on features by moving average of pair time-step observations is based on the reasoning that online commentaries and review scores are often the average of all consumer submissions over time instead of discrete values, hence on any particular instance an consumer obtains the cumulative averages of these informations for the purchasing decision. Logically, once the cumulative averages are calculated, cutting off some early time-step is of little effect to the integrity of the future data, coupled with the difficulty in backward filling of that missing time gap for the target variables - as this would either be a linear trend or constant, neither is useful information for model fitting, the decision is to truncate the dataset based on the available observations of the target variables (shown as the bottom graph above).

At this point the data for each product is cleaned and ready for feeding into any model. The next part uses various tools to explore the dataset for the purpose of understanding the data and for creating particularly lagged terms as part of the features to forecast future values.

4.3 Data Exploration

The foremost step in any time series analysis, indeed any data analysis, should start with exploration of the data to be studied, specifically the careful scrutiny of common summary statistics, correlations, visual patterns like trend, seasonality, and variation (Shumway and Stoffer 2017).

This section lists a preliminary data exploration including visualisation of **sample product dataset,** autoregression analysis of targets and summary statistics. All the analysis is implemented on the resulting preprocessed dataset detailed in the previous section. An overview of the final dataset to be fed into the models reviewed at the end.

The first thing to note from the all variables plot in Figure 4.3 is that the four target variables associated with price and stock movement appear not to have any evident correlation with each other. Another clear information results from the necessary preprocessing the latter four feature variables: they all show a gradual convergence and diminishing variation after the point at which their original data points exhaust.

Even with simple visualisation, it seems arguably that no significant correlation exist among the variables. Of course variance analysis is carried out later but this is generally the phenomenon of real world dataset, especially as in this case when the data are not fully continuous and extracted from various aspects of an online product.

It would still be helpful to visualise the cross-correlation of the variables included to have an understanding in mind for the later pipeline of modeling and analysis:

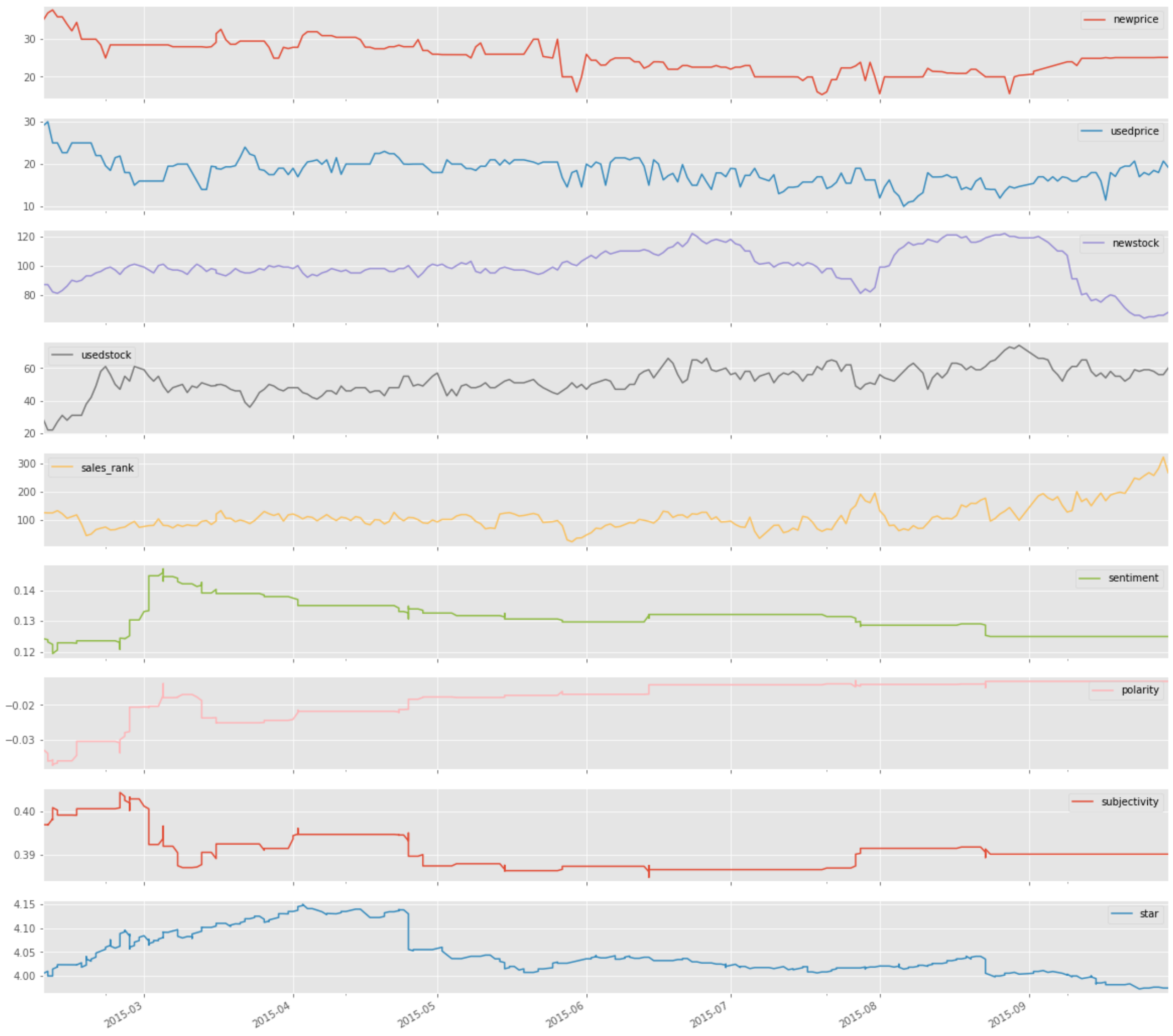


Figure 4.3: Plots of all variables

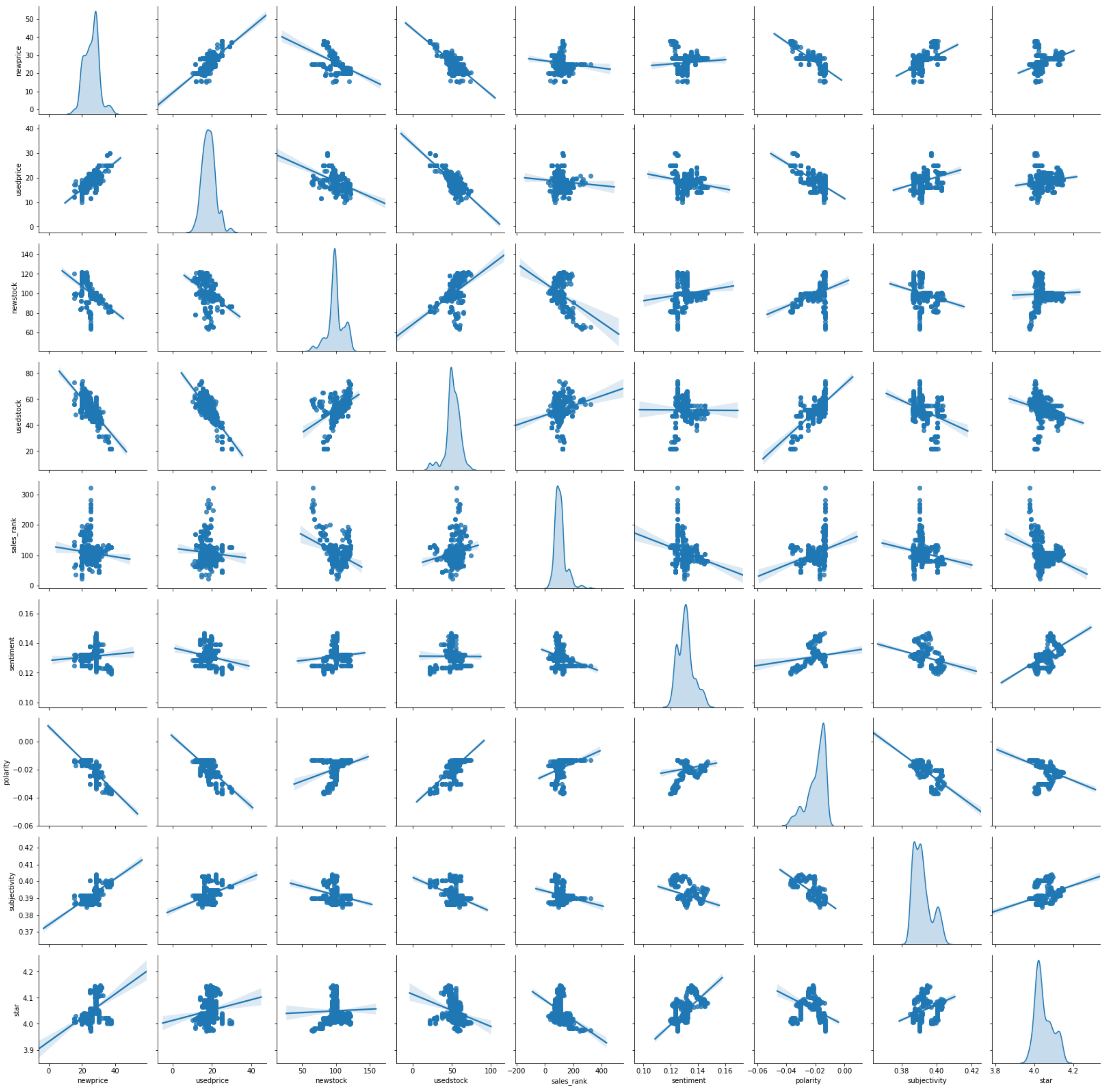


Figure 4.4: Pair Plot of Variables

The pair-wise plot of Figure 4.4. shows all variables of a sample product dataset, with diagonal cells as the kernel density estimate of univariate variables themselves, adding a fitted line for each pair plot; from the fitted line it seems quite interesting that there appear to be some correlation, at the same time-step indeed, among the price and stock variables, as well as that of the `polarity` variable based on the answers.

From the simple pairwise plot of Figure 4.4 the most interesting information is arguably the correlation of `polarity` with other variables, especially the four target variables. Further modeling should reveal the validity of the predictive power of such variable. On the contrary, the other feature variables such as `sentiment` and `star` show little correlation with price nor stock. Of course, the result is solely based on one single product at the same time-step which could be noisy, and, more importantly, it is very likely that more complex relationships exist among the lagged terms of price and stock, as that is the logical explanation in this context that a delay in consumer decision in purchasing hence the effects on demand and then price and stock movement.

Following the correlation visualisation, it would be helpful to mention the ideas of confounded predictors and the presence of multicollinearity and their effects on forecasting. As previously discussed, the presence of confounded predictors, where two variables have a joint effect on the target variable but hardly separable as they are correlated, and multicollinearity - essentially the degree of correlation is close to perfect - are of little impact to the quality of the forecast (Hyndman and Athanasopoulos 2018). Their presence is an important hurdle that needs to be treated when the objective of the model is to study and identify the effect of each predictors onto the target variables. However, it is a concern for forecasting if the future values of the predictors lie much outside the historical range in the presence of multicollinearity. Effectively, the prediction errors are much higher if no historical or train set data contains the range of values in the future predictors.

The next part examines the important factors of autocorrelation in the time series variables, and with it the analysis of the presence of trend and seasonality. As briefly explained in the ACF part under Methodology Chapter, ACF tends to reveal underlying trend if the autocorrelations for early lags to be large and positive, and seasonality revealed by the presence of regular "spikes" in ACF.

Figure 4.5 shows the ACF (left column) and the PACF (right column) plots for the four target variables with lags up to 60 time-steps (day). A couple of information are implied by the plot. First, there is strong evidence of autocorrelation in the four time series; this may be expected given the nature of price and stock movement in this context. The other aspect revealed by

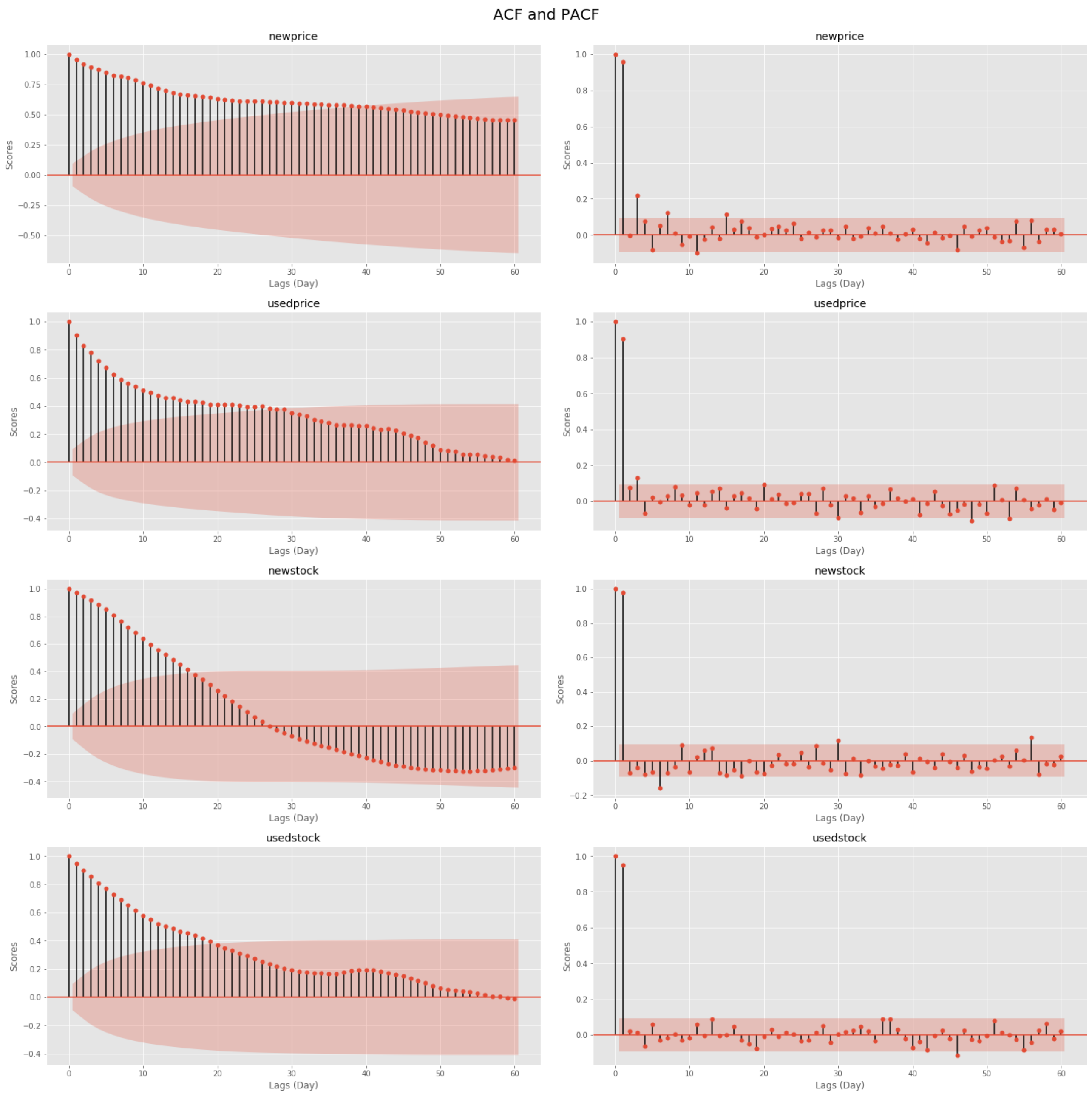


Figure 4.5: ACF

the ACF plot is that the lagged effect lingers on significantly up to 20 or so days for these variables, or the equivalence of three weeks. The final piece of information is most useful in model construction: the PACF plot clearly shows the distinct lagged effect as a result of 1 lag; in other words the effect from autocorrelation can be mainly captured by one day in the past - also worth noting is that there is little evidence suggesting the presence of seasonality as there is no regular “spike” in the PACF or ACF plot.

From the result of the previous autocorrelation analysis, it would be helpful to create a series of new features based on the one time-step of the target variables (i.e. for each x_{it} two new features are created: x_{it-1}, x_{it+1}). As the most impact from temporal values comes from the consecutive values, one time-step of the time series is sufficiently good as predictors.

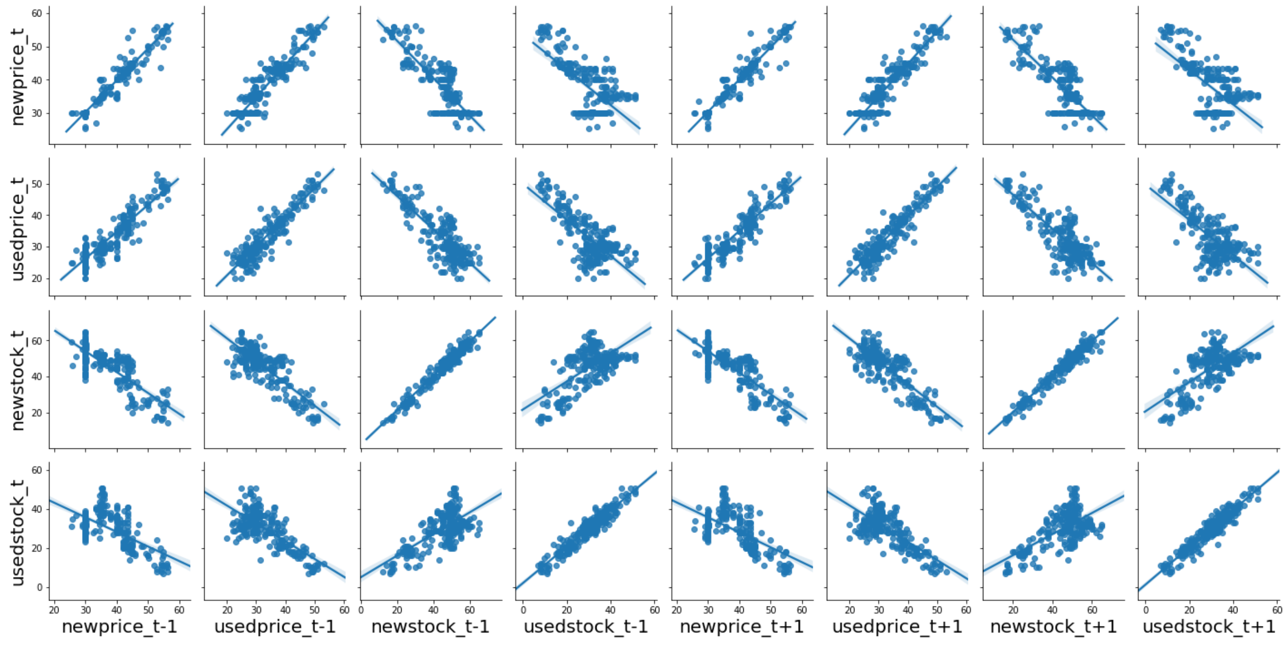


Figure 4.6: Pairwise Plot of New Variables

Another pairwise cross-correlation plot of the new one time-step variables among the target variables is shown in Figure 4.6. The evidence is surprisingly strong for the linear correlation between the consecutive temporal values among the target variables. Note that the results are based on a **sample product dataset** from the final list of 45 product datasets, therefore there may be situations where such strong temporal cross-correlation are not as strong. Nonetheless, it may still be beneficial for the prediction quality to include these new variables.

With the dataset cleaned and processed and preliminary exploration completed, the final dataset is ready for feeding into the series of models in the next section. Notwithstanding the reasonable assumptions on which the previous tasks are based, the decisions and the resulting dataset have significant influence on the output and performance of any model pipeline, a key insight alerted in any machine learning and data science project. The coarse mantra “garbage in, garbage out” can never be taken too seriously. As complex data mining is not the objective of this project, the previous pipeline of data preprocessing seems reasonable a basis for benchmarking purposes in the following model section.

Chapter 5

Implementation and Results

The final step in this paper includes the implementation of the included models using the cleaned dataset from the previous preprocessing and exploration parts of the data pipeline, concluded with analysis of comparison between the quality of forecast among the results. The structure of this chapter is a sequential description of how each model is implemented in terms of specificities configured in overall setup, parameters and model structure; also included are the visualisaiton and supplementary results, if of interest to the comparison, from the model. Finally, the collection of all evaluation metrics outlined in the Methodology is shown and used as the basis for comparison of model performance.

5.1 Model Setup

Each model is fitted on 70% train set and predicted on the rest 30% test set dataset, and the following section briefly describe the any configuration or treatment associated with each model. Of the train set, a further 20% is allocated as validation set in some models for the purpose of finding better hyper-parameters, such is the case in the Elastic Net model implementation. But most of the models are only applied directly with the default configuration and train-test data split in fit and prediction respectively.

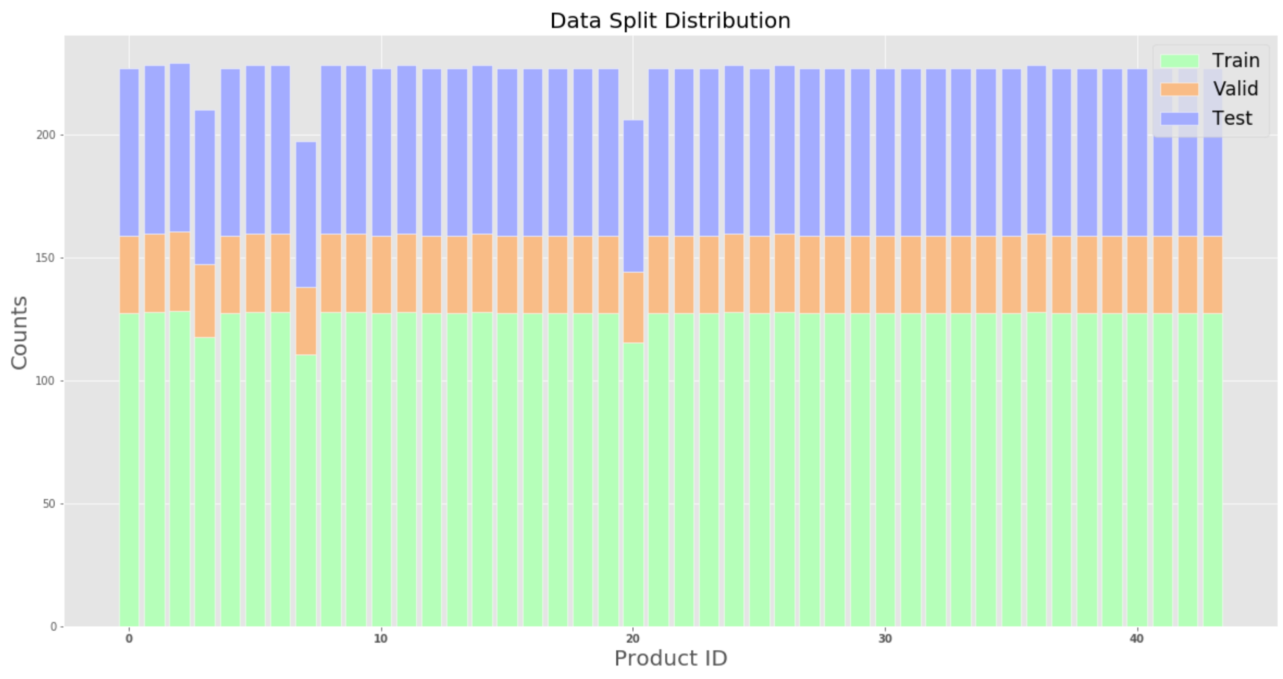


Figure 5.1: Data Allocation

Figure 5.1 shows the distribution of data split (in case of three-part split of train-valid-test). Also note that the truncated implementation dataset consists of almost equal size of dataset at roughly 230 observations or time-steps per product dataset, with only three of them having less.

As the models are diverse in nature, it is helpful to have an overview of the input or variables employed in each model. This will be particularly instructive at the final evaluation of the model performance, since the types and numbers of features fitted by the model may be an important factor in evaluating the quality and practicality of the model in production.

Table 5.1: Input Variables Table

	OLS	FE	EN	ARIMA	STS	LSTM
`newprice_t`	✓	✓	✓	✓	✓	✓
`usedprice_t`	✓	✓	✓	✓	✓	✓
`newstock_t`	✓	✓	✓	✓	✓	✓
`usedstock_t`	✓	✓	✓	✓	✓	✓
`sales_rank_t`	✓	✓	✓			✓

	OLS	FE	EN	ARIMA	STS	LSTM
sentiment_t	✓	✓	✓			✓
polarity_t	✓	✓	✓			✓
subjectivity_t	✓	✓	✓			✓
star_t	✓	✓	✓			✓
newprice_t-1	✓	✓	✓			
usedprice_t-1	✓	✓	✓			
newstock_t-1	✓	✓	✓			
usedstock_t-1	✓	✓	✓			

Table 5.1 shows the base set of variables prepared for the comparison implementation. These variables are mostly the preprocessed results from the raw dataset but added with the one time-step lagged variables of the four target variables of price and stock. In particular, some of the models may not use the exact set of variables listed in the table. For example, the first two models are very much using all of the variables listed as the models of OLS and Fixed Effect Panel Regression do not change the input variables in the fitting processing. However, in the case of Elastic Net (EN), despite indicated as using all of the variables as original input, EN model does control the weighted regularisation between $L1$ and $L2$, with the former (Lasso) may shrinking some predictors coefficients to zero as penalty. On the other hand, for ARIMA model the input variables are solely generated based on the target variables (hence the indication of the four price and stock variables as input) that may include multiple lagged and smoothed versions as a result of the auto-searching algorithm discussed in the implementation section below. Similarly in the case of STS, as explained later, its implementation ultimately gives the best performance of using only the target variable and some component distributions related to it, rather than using the other features as exogenous effects in modeling its distribution probabilistically.

5.1.1 OLS

The training or fitting of dataset in each model is reasonably simple and clear as no data mining and hyper-parameters tuning is included as part of the project. Each model is setup with the typical parameters, or best practices if there exist such set of parameters, detailed under each model section, and their results are listed and briefly evaluated. At the end of the chapter, a comparative analysis on the results is collected.

The first model to the cleaned dataset is OLS. This model is implemented using the Python library Scikit-Learn (Pedregosa et al. 2011), one of the most widely used machine learning libraries written in the Python programming language.

The model setup of OLS is the simplest as there is no special configuration on any hyper-parameter. The input of dataset is, as in the case in other model implementation below, split in the 70% train set and 30% test set. Once the model fitted and predicted on test set, the final averaged list of metrics (both mean and median) are reported in the final comparative table for further analysis.

5.1.2 Fixed Effect Panel

Unlike any other model in included in this paper, the Fixed Effect Panel OLS takes the input of a panel dataset, while others are fitted on the individual product (or entity) dataset. As explained in the Methodology, FE assumes a fixed, time-invariant effect per entity over time, similarly in idea as a dummy identifier for each entity but the effect is unobserved and therefore require special treatment in the fitting calculation.

The FE model is implemented using the Python library `linearmodels` which is an extension of the popular `statsmodels` to include some specialist models such as panel models that are commonly employed in econometrics.

Table 5.2: PanelOLS Parameter Estimates

Variables	Parameter	Std. Err.	T-stat	P-value	Lower CI	Upper CI
newprice_t	-0.0564	0.0179	-3.1477	0.0017	-0.0916	-0.0213
usedprice_t	0.1329	0.0166	8.0216	0.0000	0.1004	0.1653
newstock_t	0.0428	0.0112	3.8247	0.0001	0.0209	0.0648
usedstock_t	0.8811	0.0126	69.988	0.0000	0.8564	0.9058
sales_rank_t	0.0005	0.0001	3.8439	0.0001	0.0003	0.0008
sentiment_t	0.9040	2.2625	0.3995	0.6895	-3.5312	5.3391
polarity_t	1.7894	3.3167	0.5395	0.5895	04.7124	8.2911
subjectivity_t	-1.6531	2.1386	-0.7730	0.4396	-5.8454	2.5392
star_t	-0.6665	0.6975	-0.9556	0.3393	-2.0339	0.7008
newprice_t-1	-0.6263	0.0180	-1.4630	0.1435	-0.0616	0.0089
usedprice_t-1	-0.0156	0.0166	-0.9394	0.3475	-0.0482	0.0170
newstock_t-1	-0.0340	0.0112	-3.0324	0.0024	-0.0560	-0.0120
usedstock_t-1	0.0861	0.0127	6.8046	0.0000	0.0613	0.1110

Table 5.3: Model Statistics Summary

Item	Value	Item	Value
Dep. Variable	usedstock_t+1	R^2	0.9122
Estimator:	PanelOLS	**Log-likelihood**	-1.572e+04
No. Observation	7040	**F-statistic**	5580.7
Cov. Estimator	Unadjustted	**P-value**	0.0000
Entities	44	**Distribution**	F(13,6983)
Time periods	194		
F-test (Poolability)	3.2120		
P-value	0.0000		
Distribution	F(43,6983)		

While the standard results are summarised as other models in the evaluation at the end, the results shown in Table 5.2 and Table 5.3 provide the detail statistics of FE model fitted on the entire panel dataset regressed against the target variable `usedprice_t+1`. Some common statistics such as R^2 and $F - test$ on panel entity or fixed effect are relatively good, but this is likely due to the strong autocorrelation found between one time-step in the target variable. The quality of prediction objective is analysed on the test data prediction later.

5.1.3 Elastic Net

Same as OLS in implementation, Elastic Net model fit and predict is carried out by the Scikit-Learn library as well. The configuration is the basic setting suggested by the library - essentially setting the α weight control between $L1$ and $L2$ (or Lasso and Ridge) to 0.5.

The process is again executed on all the product dataset individually, and the final results are divided into `price` and `stock` categories for comparison - both mean and median values of all products are reported.

5.1.4 ARIMA

The implementation of ARIMA model has one distinct difference compared with other models analysed in this paper: it is a regression model that takes only the target variables as input, or rather using multiple transformation of the time series of interest.

As briefly explained in the Methodology, the famous and popular ARIMA model is one of the most studied and applied model framework in the field of time series analysis and forecasting. Perhaps a small part of the reason for its celebrity comes from its complexity of model hyper-parameters configuration. Not only has it the three fundamental hyperparameters in the order of MA and AR as well as the differencing, there are many settings such as associated with seasonality frequency that are quite sensitive in the influence of forecast results.

Therefore, although research studies may find it helpful to analyse the underlying properties of the appropriate hyper-parameters, from the exploration of differencing to stationarity, in practice and in the implementation of this paper it is more useful to employ robust programming libraries that iteratively searches for the improved configurations. The ARIMA model is implemented using the Python library `pmdarima`.

Table 5.4: ARIMA Parameter Estimates

Variables	Coef	Std. Err	Z	P-value	Lower CI	Upper CI
const	-0.1369	0.052	-2.632	0.009	-0.239	-0.035
ar.L1	-0.8188	0.126	-6.482	0.000	-1.066	-0.571
ar.L2	-0.8526	0.124	-6.859	0.000	-1.096	-0.609
ar.L3	0.1248	0.124	1.003	0.318	-0.119	0.368
ma.L1	0.1752	0.095	1.853	0.066	-0.010	0.361
ma.L2	0.3233	0.085	3.814	0.000	0.157	0.489
ma.L3	-0.7396	0.092	-8.033	0.000	-0.920	-0.559

Table 5.5: ARIMA Model Summary

Item	Value	Item	Value
Dep. Variable	usedstock_t+1	**S.D. of Innovations**	98
Estimator:	ARIMA(3,1,)	**Log-likelihood**	-350.328
No. Observation	159	**AIC**	716.656
Method	css-mle	**BIC**	741.208
		HQIC	726.626

The results shown in Table 5.5 and 5.6 demonstrate an example of the model summary of the `pmdarima` library after fitting on the train set from a sample product dataset on the target of `newprice`. The information to note from the summary is the hyper-parameters discovered by the algorithm, **ARIMA**$(3, 1, 3)$ in this case. Specifically this means that this particular time series dataset of `price` follows a AR at an order of 3 (time-step or lag), 1 degree of differencing (or one-time differencing conducted) and an order of 3 MA. The next graph shows the forecasting performance of this particular ARIMA model on the particular product dataset with the configurations mentioned.

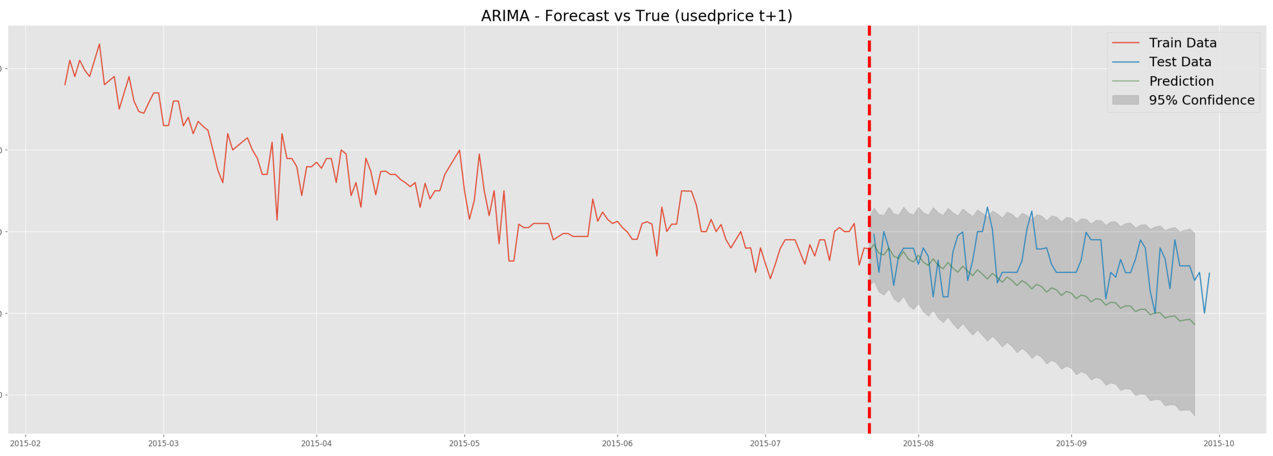


Figure 5.2: ARIMA Prediction

As shown in Figure 5.4, this sample model captures the decreasing trend correctly at the point of the train-test split, which evidently is when the future true price starts to stablise. It seems interesting that the model predicted relatively good for the early period of the test set; it may better capture the future movement if the time interval is shorter. Of course, this is only a single fit and predict on one dataset. The final comparison examines the overall performance of the ARIMA model against the others.

5.1.5 Structural Time Series

The STS model framework is implemented by TensorFlow Probability (Dillon et al. 2017), a probabilistic programming library created and open sourced by Google. It is based on Bayesian inference which in the case of STS model parameters using established methods of either variational inference (VI) or Hamiltonian Markov Chain (HMC) to compute point estimates and variances. The mathematical construction and validity are well beyond the scope of this paper. It suffice for the purpose of the project to note that the STS model architecture and computation design implemented by TensorFlow Probability are rather robust and easy to operate and interpret, as shown below.

In the Methodology, it is briefly explained that the STS framework models the distribution of the target time series by an additive composition of many component distributions. In constructing the model, the choice of whether to include the feature variables as exogenous effects emerged as an interesting question. Besides the base effects of *Local Trend, Weekday Seasonality* and *AR(1)*, as explained in the Methodology and based solely on the target variable itself, additional tests are performed to compare the post-model variance of various sets of exogenous effects. Figure 5.5 shows a sample forecast result of the STS model adding the exogenous effects as component, and it appears similar to the result in that modeled without in Figure 5.6. The difference is more revealing in their respective variance plots in Figure 7.1 and Figure 7.2 in the Appendix: the variance of each base component distribution of the target variable is heavily distorted by the introduction of the set of exogenous variables, whereas that of the base model shows a much smaller variance. Therefore, the final results are predicted using the STS model taking only the base components of the target variable based on the test explained. The test results also imply that in this case the features for the product help little to model the target distribution while adding lots of noise to the probabilistic model.

An important note of the STS model implementation is that the forecast results are generated from the sampling of the learned probabilistic model based on the components. Concretely, the forecast is not one time-step as with the other regression models where the forecast is predicted one time-step using the previous time-step true values of all of the features. This is similar to the ARIMA model where the forecast is effectively a continuous projection of the past ARIMA processes, not based on some past true values of features. Therefore, this vital difference in the nature of model fit and prediction will influence the evaluation of the comparison analysis and conclusion on the applications of the models studied in production.

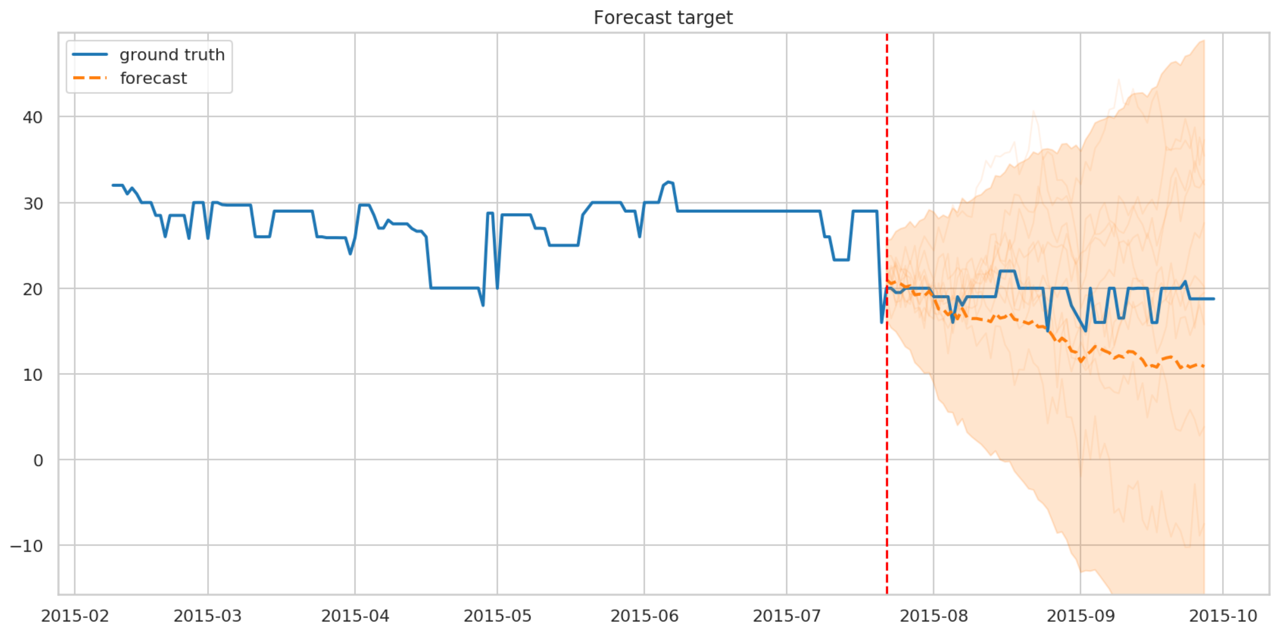


Figure 5.3: STS Sample Plot (with Exogenous Effects)

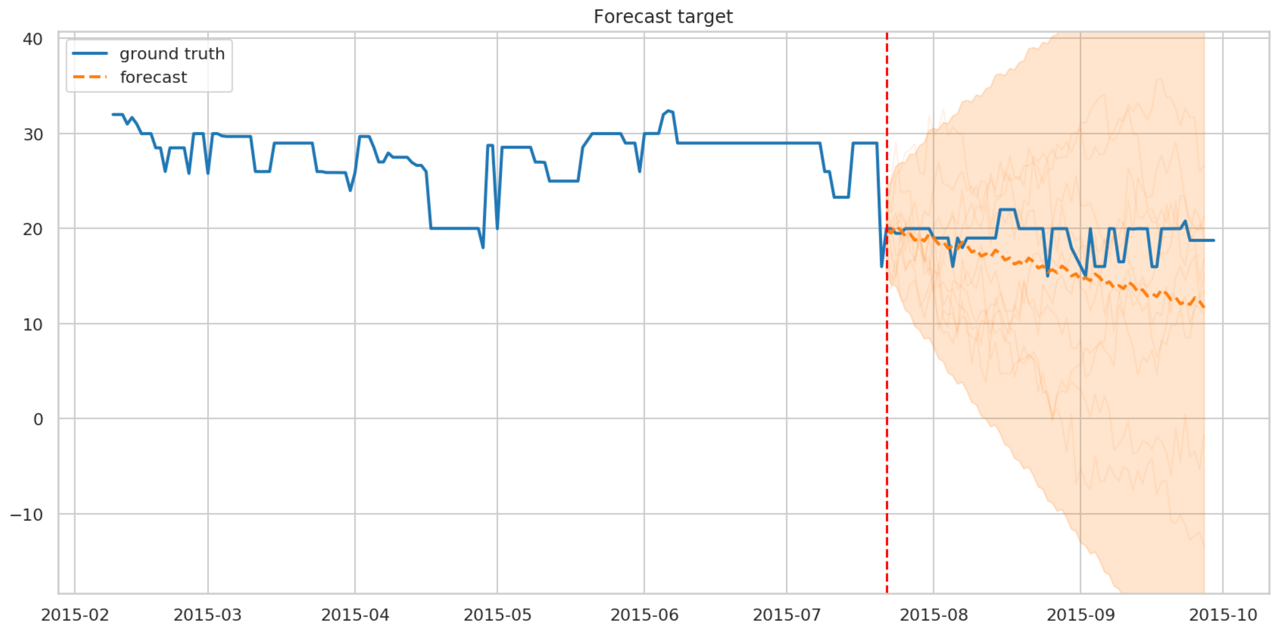


Figure 5.4: STS Sample Plot (Without Exogenous Effects)

5.1.6 Neural Nets

The final model to be compared in this paper falls under the category of neutral networks or deep learning. As introduced in the Methodology, neutral nets are very flexible in its capability to model complex relations between input and output. Such flexibility has its pitfalls. The modular nature of neural nets implies exponential possibility in the configuration of the architectural design. For the goal of standardised comparison with other models, the implementation of neutral nets in this case has certain preconditions. First, neutral nets is capable of a diverse types of input and output in the case of time series forecast: input can be univariate similar to that of ARIMA and also multivariate as in Elastic Net, while output can be any arbitrary time-steps like that in the ARIMA and STS. The decision is to choose a multivariate input with one time-step forecast output, similar to other models except STS.

Specifically, the neural nets implemented has the following configurations:

Table 5.6: LSTM Hyper-Parameters Explanation

Hyper-Parameters	Values	Comment
Input Time-Steps	7 (days)	*Each input sequence / data is a 3 dimensional array of* $Batch \times 7 \times 9$ *where Batch is the size of subset of dataset used per update of weights/parameters,* 7 *the window size (time-steps) used to predicted the future one time-step value and* 9 *the number of features used as input. (Note: the Batch size of the model is 1*
Output Time-Steps	1 (day)	*Each output/target is a scalar of the future one time-step value of the target variables, either price or stock*
Layer-1 (node)	LSTM (32)	First (hidden) layer of the neural nets is the LSTM layer with 32 (hidden) nodes - it transforms the input array into an 3 dimensional array of size $Batch \times 32$

Hyper-Parameters	Values	Comment
Layer-2 (node)	LSTM (16)	Second layer of LSTM type with 16 nodes with array of size $Batch \times 16$
Layer-Output	Linear (1)	Final output layer transforming the previous $Batch \times 16$ array into a scalar value as the target.
Epochs	30	The number of times the full training set is looped or repeated through the model in order to update the parameters / weights
Loss Metric	*MSE*	The loss function objective that the model is subject for the optimisation

With the standard and basic configuration shown in Table 5.2, the neural nets model is implemented by the same pipeline using the 45 product datasets repeatedly, with final results of price and stock summaries shown in the next section of the chapter.

5.2 Evaluation and Comparison

With all model implementations explained in the previous section, the total collection of their results ran and tested on the full dataset is summarised in the following tables, separated by target variables into `price` and `stock` tables.

Table 5.3 shows the results of evaluation metrics of all models based on the target variable `newprice_t+1` and `usedprice_t+1` taken as average and median values. Each cell contains the **mean / median** of the corresponding metrics of all product datasets (except Fixed Effect model) and then averaged over the two price target variables.

Table 5.7: Evaluation Metrics (Target = `price`) (mean / median)

Models	R^2	MAE	RMSE	MAPE (%)	AIC	BIC
OLS	8.887e+26 / -1.612	2.692 / 2.086	3.235 / 2.551	19.80 / 15.638	137.317 / 142.748	166.261 / 171.791
FE	**0.830**	3.120	3.833	35.190	7676.05	7777.54
EN	-3.086e+26 / -0.011	**1.499 / 1.297**	**1.874 / 1.705**	**12.713 / 9.836**	**91.404 / 101.491**	**122.574 / 132.77**
ARIMA	-5.513e+27 / -2.136	2.852 / 2.153	3.310 / 2.872	20.152 / 16.321	138.039 / 145.568	153.002 / 157.67
STS	-1.976e+26 / -2.591	3.260 / 2.718	3.788 / 3.260	22.776 / 19.227	NA	NA
LSTM	-1.719e+28 / -1.880	3.011 / 2.224	3.456 / 2.818	22.441 / 17.831	564.985 / 571.631	1029.99 / 1043.34

Table 5.8: Evaluation Metrics (Target = `stock`) (mean / median)

Models	R^2	MAE	RMSE	MAPE (%)	AIC	BIC
OLS	-1.132 / -0.125	5.444 / 3.934	6.478 / 4.924	10.898 / 8.998	241.935 / 236.76	171.791 / 265.839
FE	**0.972**	4.264	5.088	9.978	9442.26	9543.745
EN	0.675 / 0.758	**2.141 / 1.849**	**2.697 / 2.342**	**4.633 / 4.355**	**154.918 / 144.548**	**186.089 / 175.826**
ARIMA	-6.635 / -2.576	10.343 / 8.330	11.931 / 9.750	21.87 / 14.468	309.145 / 311.61	324.724 / 325.318
STS	-29.366 / -3.413	15.349 / 8.823	17.695 / 9.958	19.227 / 17.172	NA	NA

Models	R^2	MAE	RMSE	MAPE (%)	AIC	BIC
LSTM	-29.313 / -4.176	17.335 / 9.169	18.36 / 10.63	23.958 / 18.621	735.845 / 724.31	1200.847 / 1203.31

Similar to the results in Table 5.4 for the `stock` target variables, the model with the lowest measurement errors across the various types is evidently Elastic Net. The result may first seem surprising given that the less complex models such as OLS and Elastic Net achieve lower forecast errors than the much more advanced ones such as STS and LSTM. In particular, the performance of Elastic Net is consistently better than the rest across the metrics in the two tables. Elastic Net generally performs better than OLS in cross-sectional datasets with large size of feature variables as it regularises the poor variables. In terms of `stock` target, the overall results have the same story. The performance, especially based on **median** values, of the latter three models are closely following each other.

However, it is vital to account for the context and settings in which the experiment and implementation are conducted in this paper. First, although there are 45 product datasets, all models but Fixed Effect only takes individual dataset at a time, which is only 220 observations. This is very small size in the field of neural nets modeling. The second consideration is that the prediction window in this case is limited as the future one time-step. For models like ARIMA, STS and neural nets, they are more than capable of long time-step prediction compared with traditional regression models which cannot handle long sequential input and output. In fact, the STS model is implemented with the prediction window as the full time-step of the test set, not one time-step at a time.

Another aspect of the results is that for models such as neural nets, there amount of architectural setting is large that at one hand it is powerful to have complex layers of different transformations but it requires large amount of data to learn all the parameters. In this respect the performances of the neural nets and STS may be well underestimated in this case. Equally, the implementations are not exhaustive in the sense of hyper-parameters tuning, which is a process of repeatedly searching for the optimal setting of various hyper-parameters of the model that is not learned during the training. Though this is the basis for all models, models with more hyper-parameters such as STS and LSTM may be far less accurate than OLS and EN that has a few ones to tune.

Chapter 6

Conclusion

The results of the final evaluation suggests that the forecast performance implemented in this paper for traditional methods are still producing good forecasts given the premise of single time-step window. It requires further experiments on different model settings and datasets to study the validity of such conclusion in the general case. However, based on the nature of the models it is true that models such as STS and neural nets are naturally capable of handling large datasets with high dimensional feature input and, critically in application, the ability to generate long range window forecast.

In short, based on the set of models, the overall result seems to suggest that the advanced machine learning methods are not outperforming the baseline models of OLS and Fixed Effect regression for this particular project dataset. This could partly be explained by the size of time series dataset - small 200 or so observations for each entity modelled that even with a simple linear mapping, the OLS regression ends up with a relatively lower MSE and MAE metrics compared with advanced, high capacity model frameworks of STS and LSTM neural network.

For the objective of online arbitrage of supplies, from the perspective of procurement management, the application of forecast system may implement traditional regression models such as OLS and Elastic Net, if the purpose is short window forecasting to arbitrage at online marketplaces. For long term trend prediction, the probabilistic model of STS and neural nets are more applicable in processing large datasets and features set. For example, these two models are more capable of future price trend, such as going up or down, that is useful in managing lower cost procurement by sourcing supplies whose price is forecast to be falling (assuming contract price is dynamic to market prices).

In this paper, the limitation of machine learning models is revealed by applying on this dataset with the characteristics of having small size of observations. As such, the complex models produce worse results in the given setting than simpler models of OLS and Elastic Net. In more serious application, however, the amount of datasets will be large with tens or hundreds of features available. More advanced models that can handle diverse input and output characteristics but requiring large dataset would likely to be more useful, if not the only applicable models. Further experiments on applying the models included in this paper on larger datasets would be interesting to test the idea.

Chapter 7

Appendix

Structural Time Series - Component Variance (Sample 1)

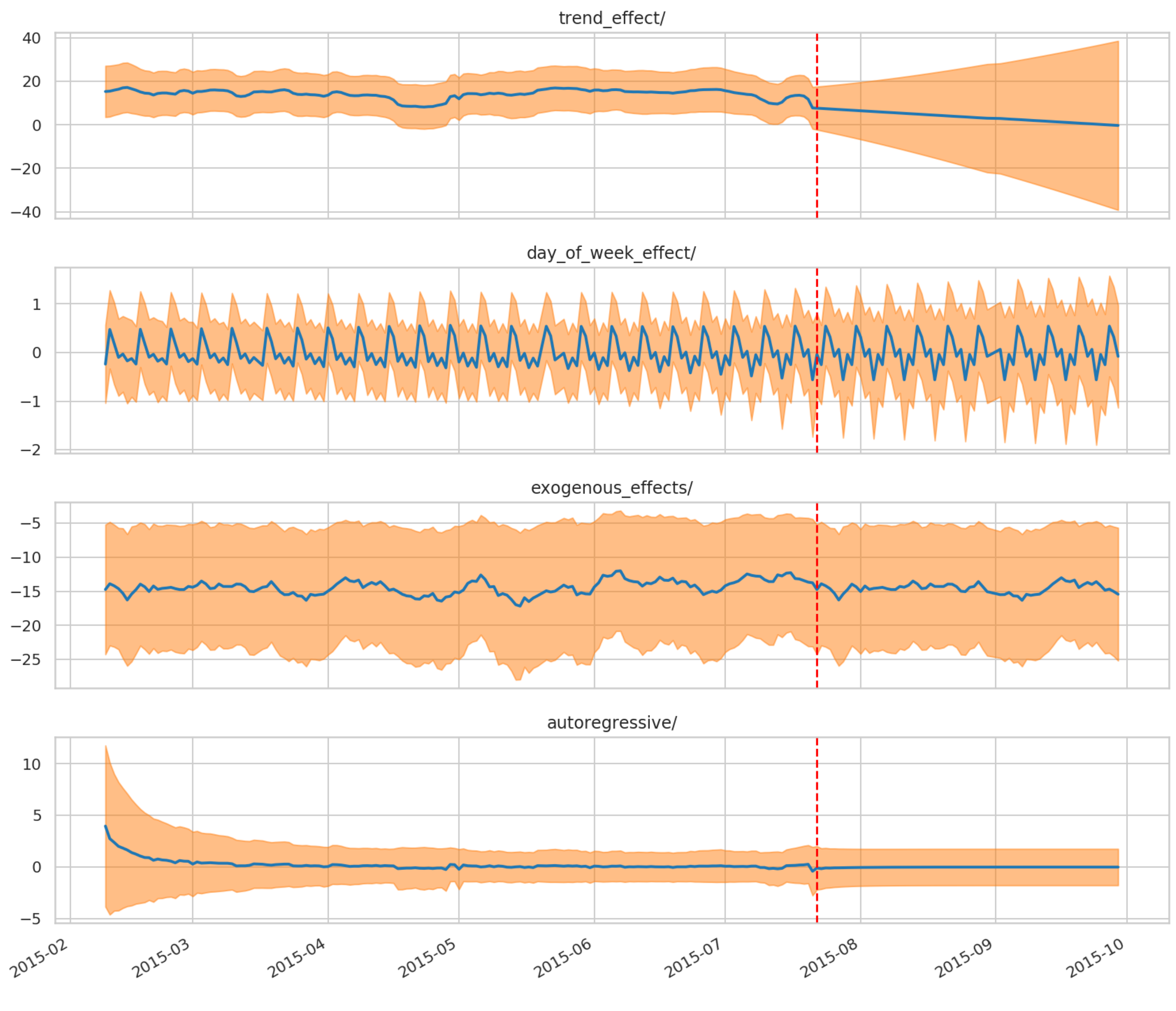


Figure 7.1: Decomposed component plot 1

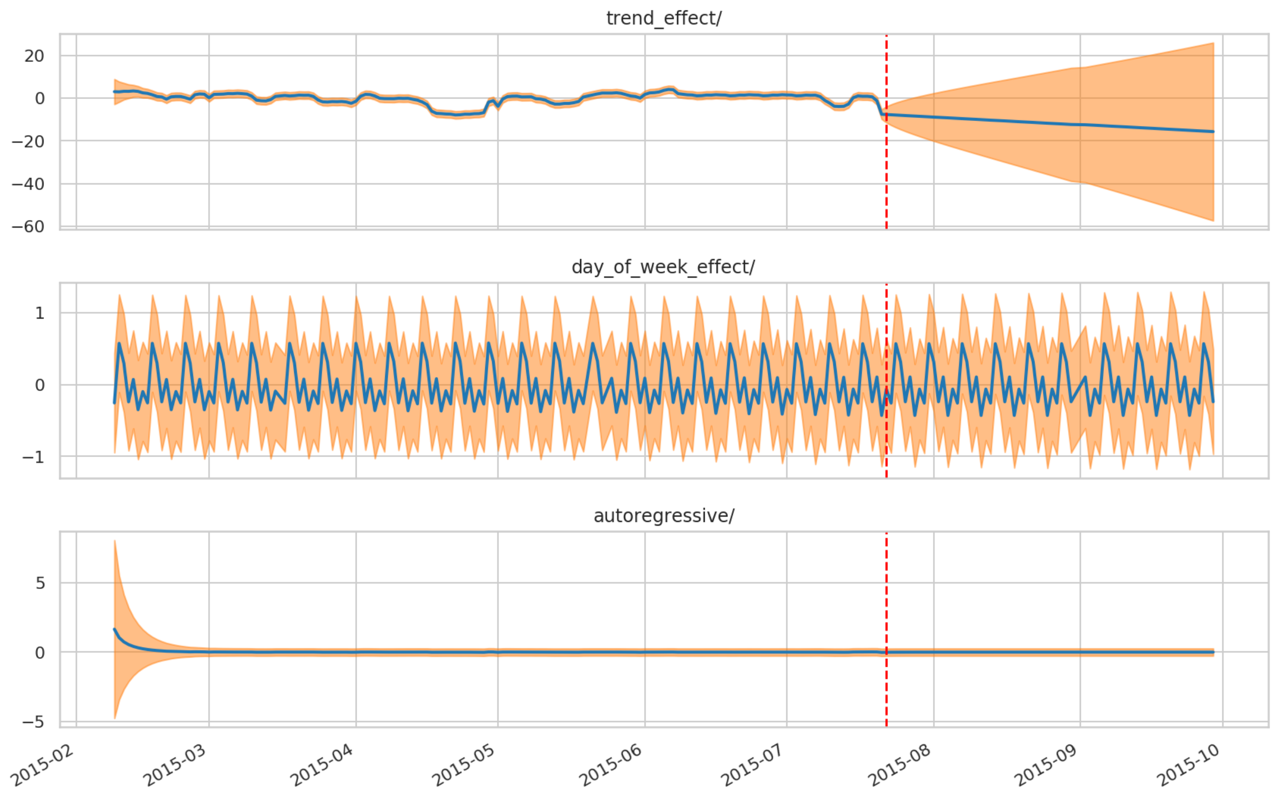


Figure 7.2: Decomposed Component Plot 2

References

Chollet, François, and others. 2015. "Keras." https://keras.io.

De Gooijer, Jan G, and Rob J Hyndman. 2006. "25 Years of Time Series Forecasting." *International Journal of Forecasting* 22 (3). Elsevier: 443–73.

Dillon, Joshua V, Ian Langmore, Dustin Tran, Eugene Brevdo, Srinivas Vasudevan, Dave Moore, Brian Patton, Alex Alemi, Matt Hoffman, and Rif A Saurous. 2017. "Tensorflow Distributions." *arXiv Preprint arXiv:1711.10604*.

Dorffner, Georg. 1996. "Neural Networks for Time Series Processing." In *Neural Network World*. Citeseer.

Gamboa, John Cristian Borges. 2017. "Deep Learning for Time-Series Analysis." *arXiv Preprint arXiv:1701.01887*.

Gers, Felix A, Douglas Eck, and Jürgen Schmidhuber. 2002. "Applying Lstm to Time Series Predictable Through Time-Window Approaches." In *Neural Nets Wirn Vietri-01*, 193–200. Springer.

Goodfellow, Ian, Yoshua Bengio, and Aaron Courville. 2016. *Deep Learning*. MIT press.

Hastie, Trevor, Robert Tibshirani, Jerome Friedman, and James Franklin. 2005. "The Elements of Statistical Learning: Data Mining, Inference and Prediction." *The Mathematical Intelligencer* 27 (2). Springer: 83–85.

Hochreiter, Sepp, and Jürgen Schmidhuber. 1997. "Long Short-Term Memory." *Neural Computation* 9 (8). MIT Press: 1735–80.

Hyndman, Rob J, and George Athanasopoulos. 2018. *Forecasting: Principles and Practice*.

OTexts.

Murphy, Kevin P. 2012. *Machine Learning: A Probabilistic Perspective*. MIT press.

Pedregosa, F., G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, et al. 2011. "Scikit-learn: Machine Learning in Python." *Journal of Machine Learning Research* 12: 2825–30.

Shumway, Robert H, and David S Stoffer. 2017. *Time Series Analysis and Its Applications: With R Examples*. Springer.

Sutskever, Ilya, Oriol Vinyals, and Quoc V Le. 2014. "Sequence to Sequence Learning with Neural Networks." In *Advances in Neural Information Processing Systems*, 3104–12.

Wooldridge, Jeffrey M. 2010. *Econometric Analysis of Cross Section and Panel Data*. MIT press.

———. 2015. *Introductory Econometrics: A Modern Approach*. Nelson Education.