Using Gaussian Processes to Address the Issues of Predicting Financial Time Series

Student ID: 21182741

Abstract

In this report, the statistical modelling technique *Gaussian process regression* is exploited to generate predictions over a financial time series (logarithmic returns of the cryptocurrency pair *ETH/USD*). The method is shown to overcome challenges of complex systems, such as non-linear dependencies between variables and non-stationarity, producing a model that captures the highly volatile trend of this financial time series and generalises well to future observation times.

1 Introduction

Developing a statistical model to describe and interpret the distribution of values in a financial market as it evolves is a challenging task, as this data is part of a complex system comprised of many connections and dependencies between features and variables, and requires us to formulate a multivariate probability distribution that accurately captures the properties of the dataset. Because of this complexity, using such a model to predict how outcomes will evolve—forecasting the value that these variables will take in future—is an even more challenging task.

Data from financial markets, such as prices of a given cryptocurrency, can be easily represented as a *time series*: a sequential ordering of data points where each value y is observed at a time t, with the collection of observation values Y (the dependent variable) being ordered by a chronological index of observation times X (e.g. seconds, days, or years). For example, this report focuses on the evolution of *closing prices* of the cryptocurrency pair *ETH/USD* (i.e. how the final value of Ethereum at the end of each five-minute trading window changes over time based upon exchanges between it and the US Dollar); in this case, the dependent variable $Y\colon y_0, y_1, \ldots, y_n$ is a sequence of these closing prices, with $X\colon x_0, x_1, \ldots, x_n$ denoting the times at which these prices were observed.

Hence, to make predictions of what value the variables in a time series will take in future we want to find a mapping $Y = f(X) + \varepsilon$, such that the relation between times X and values Y is represented with the smallest possible residual error ε, and extrapolate observations Y^* at unseen time points X^*. This process is known as *regression*, and can be learned *parametrically* (where we assume the parameters of the probability distribution the data is drawn from a priori) or *non-parametrically* (where no such assumption is relied upon). Due to the complex non-linear interdependencies between variables, performing regression on systems such as financial market time series can generate limited success when using traditional methods such as *linear regression* (Lin et al. 2007) or *logistic regression* (Baidoo & Priestley 2016), resulting in regressions that still have a large residual and thus cannot be relied upon to make accurate predictions.

In this report, *Gaussian process regression* (GPR) will be explored as an alternative to typical regression approaches, and its utility evaluated in the context of probabilistic prediction of financial market time series. GPR is a kernel-based non-parametric regression technique based upon *Gaussian processes* (GPs) that use Bayesian statistics (where variables are assumed to be drawn from some unknown probability distribution) to learn a regression and model a predictive distribution. This approach has been successfully used to model time series such as the Mackey-Glass differential equations (Girard et al. 2002), respiratory rate patterns (Brahim-Belhouari & Bermak 2004), and weather sensor data (Roberts et al. 2013). Limited research has been conducted on using GPR over financial time series, such as the volatility forecasting of Liu et al. (2020) and Han et al. (2016), however, this field has not yet been extensively explored and the potential of GPR for financial predictions showcased.

Hence, this report will explore the utilisation of GPR for the prediction of financial returns of the cryptocurrency pair ETH/USD, evaluating the benefit of exploiting GPs in this domain. This exploration will be structured as follows: initially, the challenges of financial datasets will be highlighted, and the failing points of traditional methods on these challenges identified; secondly, GPs and GPR will be introduced, exploring the methods underlying this report; finally, the results generated by GPR on our dataset will be discussed, with the benefits and issues associated with this model evaluated and future improvements considered.

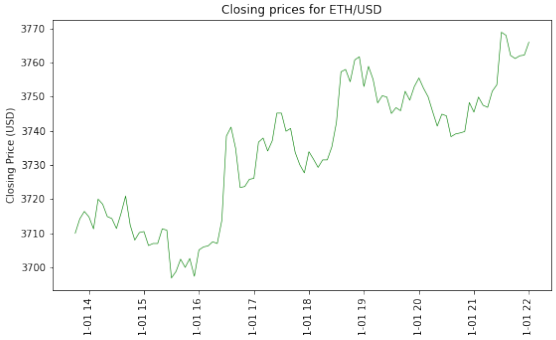
2 Methodology and Data

2.1 Challenges of the Dataset

2.1.1 The Dataset

As discussed, this report focuses on the evolution of closing prices of the cryptocurrency pair ETH/USD. Specifically, over a global time horizon of 500 minutes, the price of Ethereum (ETH) in US Dollars (USD) was observed at 5-minute intervals, with the price at the end of each interval recorded as the closing price at that time. This is represented through the time series $T_{price} = (X, Y)$ where $X : x_0, x_1, \ldots, x_{99}$ are indices representing the 100-time steps over which observations were taken, and $Y : y_0, y_1, \ldots, y_{99}$ the closing prices at each of these times. Figure 1 shows the ETH/USD exchange over the 100 data points in question, starting at *13:45* on *01-01-2022* and ending at *22:00* on *01-01-2022*. The figure illustrates how an upwards global trend is seen in price over this time horizon, but fluctuations in price around this trend are common, often exhibiting large jumps in value in the positive and negative directions. These fluctuations are known as the *volatility* of the prices, which is a statistical measure of dispersion.

Figure 1: Closing prices of ETH/USD over a 500 minute time horizon.



2.1.2 Stationarity

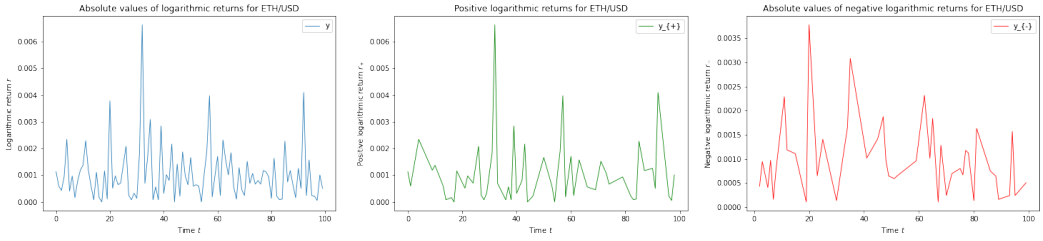
When developing a model for a time series, one must confirm that the underlying statistical properties of that series are consistent for the entirety of the time horizon over which we are modelling to ensure that the model remains valid. Thus we must validate that the series is statistically identical over the length of the time series, with parameters of the distribution, for example, the standard deviation, remaining constant. This is known as the *stationarity* of the data. Upon visual inspection of our ETH/USD dataset, it appears to exhibit *non-stationarity*—meaning the statistical properties do change over time—as an approximately linear trend of increasing value can be observed. This non-stational hypothesis can be checked quantitively through the *Augmented Dickey-Fuller test* (Mushtaq 2011). The test defines the null hypothesis H_0 that the time series being tested exhibits some non-zero trend, and is thus non-stationary. We then attempt to falsify this hypothesis, in turn leading us to believe the alternative hypothesis H_1 that the time series is stationary. To check the stationarity of the ETH/USD closing prices used in this report, the Augmented Dickey-Fuller test is conducted using the *adfuller* method provided by the *statsmodels* library (Seabold & Perktold 2010) which outputs a

p-value (the probability under H_0 that any other series of observations is more extreme, and hence fits H_0 less than the given series). When evaluating our ETH/USD prices, a p-value of 0.5939 was output. Using the p-value threshold of 5%, we cannot reject the null hypothesis H_0 that the prices are non-stationary (as $0.5939 > 0.05$) and hence we must assume non-stationarity in our data.

Non-stationarity poses an issue for building a model of the data, hence, to accurately model this domain it must be removed. Typically, to detrend a financial dataset and remove non-stationarity, the *returns* over the closing prices are computed and used as an alternative time series that demonstrates the same information whilst being stationary (Liu et al. 2020). The return is computed as the forward difference between consecutive prices in the time series, and the logarithm of this is taken to simplify the computation process, generating the *logarithmic return* (or *log-return*); this is shown mathematically as the computation of the return R_t in Equation 1. This method is exploited over our dataset before a model is constructed, reformulating the time series as $T_{return} = (X, Y)$ where $Y : r_0, r_1, \ldots, r_{98}$ represents the log-returns indexed by X (note as returns represent a difference, the time series will contain one less element). The Augmented Dickey-Fuller test can again be implemented to establish the stationarity of this dataset; evaluating ETH/USD log-returns generates a p-value of 8.306×10^{-13}, thus (as $8.306 \times 10^{-13} < 0.05$) the null hypothesis that the time series T_{return} is non-stationary can be rejected and the alternative hypothesis that T_{return} is stationary accepted. Hence, the subsequent work exploited the time series T_{return} over the log-returns of ETH/USD prices to ensure stationarity and build a valid model. Specifically, the absolute log-return $|R_t|$ was used, which combines the properties of positive and negative returns into a single distribution (Figure 2).

$$R_t = \ln\left[\frac{price\,(t)}{price\,(t-1)}\right] = \ln\left[price\,(t)\right] - \ln\left[price\,(t-1)\right] \tag{1}$$

Figure 2: Log-returns of ETH/USD over a 500 minute time horizon.



2.1.3 Dependencies and Correlations

Beyond the stationarity of the dataset, the dependencies between variables along the time series pose a challenge for developing a model, as to infer accurate predictions the interrelations between elements have to be accurately captured. Statistically modelling dependencies is inherent to the regression process, as the mapping $y = f\,(X) + \varepsilon$ exists if and only if a relation between the observation times X and the dependent variable Y exists. However, when conducting regression over a time series, the dependencies between the variables in this series also have to be considered; namely, we must take into account that the value of variable y_i at time i is likely to be highly dependent on the variable y_{i-1} that preceded it. This is especially the case when considering financial markets where prices at consecutive times are highly dependent and hence have complex interrelations that must be correctly modelled to make accurate predictions. Furthermore, the dependencies between prices are often non-linear, meaning they cannot be represented by a simple linear map $A = p + qB + \varepsilon$. The presence of such a dependency between variables means certain regression models cannot be used, such as linear regression; additionally, non-linear dependencies are known to exacerbate biases within the dataset (Phillips & Yu 2007), posing a challenge for any remaining model.

For those variables that are linearly related, the strength of their linear dependency can be measured through their *covariance* and *correlation*. If two variables A and B are linearly independent, it's

given that the *joint probability distribution* $P(A, B)$ of A and B (i.e. the distribution over conjunctions of the two variables) is simply the product of the probability distributions of each variable $P(A)$ and $P(B)$ (known as the *marginal probability distributions*). This is not the case if the variables are linearly dependent, as the joint distribution is given by $P(A, B) = P(A) * P(B \mid A)$ (where $P(B \mid A)$ is the probability of B conditional on the value of A). Hence, to quantify the linear dependency between two variables, we can measure the distance between the value of their joint distribution $P(a, b)$ and the value that the joint distribution would take if the variables were independent (i.e. $P(a) * P(b)$) over all values $a \in A, b \in B$. This is known as the covariance $Cov(A, B)$ and is computed through Equation 2.

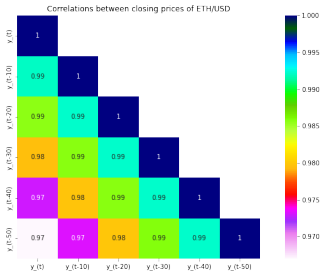
$$Cov(A, B) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} \left[P(a, b) - P(a) P(b) \right] dxdy \tag{2}$$

Since the size of the covariance between variables A and B depends upon the individual scales of these variables, we can a more general measure of dependency can be constructed by normalising the covariance by the product of each variable's standard deviation σ. This is known as the *correlation* between variables A and B and is shown in Equation 3. Because of this normalisation, correlation ranges between -1 and 1: $Corr(A, B) = -1$ indicates the variables are perfectly anti-correlated, $Corr(A, B) = 0$ indicates there's no linear correlation, and $Corr(A, B) = 1$ indicates a perfect correlation.

$$Corr(A, B) = \frac{Cov(A, B)}{\sigma_A \sigma_B} \tag{3}$$

Typically variables in a model dataset have to be *independent and identically distributed* (i.i.d)—i.e. independent variables drawn from the same probability distribution—to ensure modelling can be conducted. For example, both linear and logistic regression requires the assumption that all observation variables are independent of each other, and the residual errors of each are completely uncorrelated. This means that these methods cannot be used on correlated data, as they would produce an unrepresentative distribution and significant errors in predictions. For simple datasets, dependency relations between variables can be mitigated by *shuffling* the data, randomising the map between variables. However, this is not possible for time series, as the sequential nature of variables is intrinsic to their distribution. Hence, correlations between variables can pose a significant challenge to modelling time series, especially in complex systems such as financial markets that can exhibit many complex linear and non-linear dependencies.

Figure 3: Correlations between closing prices

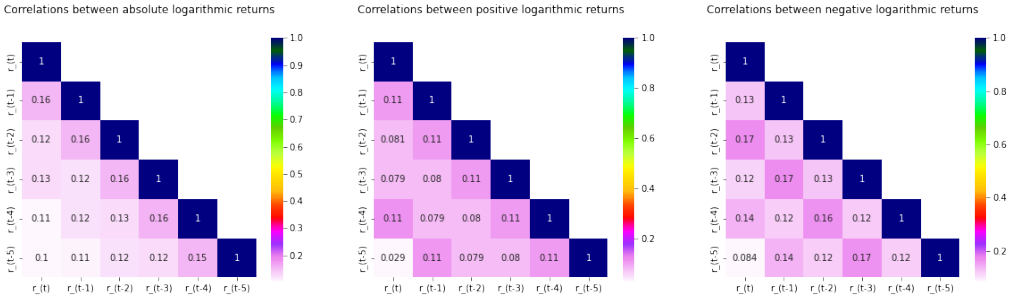


To exemplify the correlations between time-series data, and inform the model selection for this experimentation, the dependencies within the utilised ETH/USD dataset were checked using the *corr* method provided by the *pandas* library (Wes McKinney 2010) that produces a correlation matrix representing the dependencies between variables of the time series (produced though shifting the time series to different time lags). Figure 3 demonstrates this correlation matrix for the ETH/USD closing prices; the variable on each row and column of the matrix represents a time-lagged version of the time series, beginning with no lag (i.e. the original series) and incrementing in lags of 10 elements (e.g. y_{t-10} compares the correlation between variables and their predecessors 10 time steps previously in the series). The matrix demonstrates that even over 10 time step intervals the variables in the same series are highly positively correlated, with all correlation coefficients near 1, indicating there is a strong dependency between sequential elements of the time series, even significant time intervals between them (e.g. the correlation between each variable and the variable that precedes it by 50 time steps, or 250 minutes, is 0.97 indicating a high dependency). This interrelation is expected in financial time series, especially when explicitly considering prices, as

even in extremely volatile markets the price of established assets is unlikely to repeatedly change by a significant margin (with respect to the price at the previous time step) on a minute-by-minute basis.

Similarly to the issue of stationarity, the linear correlation between variables in a financial time series can be reduced by focussing on the log-returns of the asset instead of closing prices. This is because taking the logarithmic returns de-trends the dataset, reducing the extent of the linear relationship between variables, and making the development of models less challenging. The *pandas* library can similarly be used to exemplify the correlations between log-returns; the correlation matrix for the log-returns, positive log-returns, and negative log-returns is shown in Figure 4 (note time lags up to 5 time steps are used, a significant reduction from when closing prices were inspected as the interdependence of variables stretches much further into the past in that case). Upon inspecting the correlation matrices for the variables of this time series, it is clear the magnitude of the correlation between log-returns is significantly less than that of closing prices, and stretches less far back into the past, although a small positive correlation is still visible meaning the variables are still dependent upon each other. It is important to remember that despite this correlation being small, it is not zero, and so the variables within the log-return time series still exhibit a linear (and possibly non-linear) dependency. Hence, as our dataset is still not in i.i.d simple models like linear and logistic regression still cannot be used in this domain.

Figure 4: Correlations between log-returns



2.2 Gaussian Process Regression

2.2.1 Bayesian Statistics and Non-Parametric Models

A common practice when building a predictive probability distribution is to assume the type of distribution the data comes from *a priori* and then learn the optimal parameters of this distribution from the dataset; as discussed, this is known as *parametric modelling*. However, this requires knowing the distribution that our data comes from beforehand, or an extensive validation procedure to select the collect distribution. Thus, in many cases, it is preferable to construct a predictive model non-parametrically. This can be done by directly modelling the test points x^* from the testing dataset to predict the probability distribution over the possible test observations y^* without making any assumption about the underlying model of the dataset.

Bayesian statistics tells us that all variables v are drawn from some underlying probability distribution. Thus, we can predict the value a variable will take based on what prior information we know about it. For instance, say we have some probability distribution $p(v)$ specifying what we think we know about the variable v; this is known as the *prior distribution*. We can update the prior distribution based upon new information we get from observing training data points $D = (X, Y)$ that tell us more about our variable, inferring a new *posterior distribution* $p(v \mid D)$. This exact process can be done to model the predictive distribution over testing data in our time series; namely, we form the probability distribution $P(y^* \mid x^*, D)$ over possible observations y^* given the training data D and the observation time x^*. Hence, to make predictions of the next value y^* in a time series at the observation time x^* we can simply evaluate $P(y^* \mid x^*, D)$ to find the most likely value for our variable.

2.2.2 Gaussian Processes

A *Gaussian process* is a non-parametric model that uses Bayesian statistics to describe a probability distribution over a set of functions. This is done through the aforementioned concept of prior and posterior distributions. Initially, we begin with some prior distribution $p(w)$ over all possible Gaussian functions $f_{w \in W}$, where the function $y = f_w(x)$ is evaluated over a given domain through the Gaussian probability function shown in Equation 4. From Equation 4, it is clear how the function $f_w(x)$ can be manipulated by taking different values for the mean μ and standard deviation σ as parameter set $w = (\mu, \sigma)$. Hence, the prior distribtuion $p(w) = p\big(f_w\big(x \mid w = (\mu, \sigma)\big)\big)$ is constructed over all possible parameter values, with some intial probability associated with each function (e.g. a uniform distribution over all functions).

$$f\left(x \mid \mu, \sigma\right) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}} \tag{4}$$

To make predictions of the dependent variable y^* at test observation times x^* using this prior distribution, we must first construct the posterior distribution $P(w \mid X, Y)$ that describes a conditional probability distribution over the potential functions f_w that fit the data, given the dependent observation variables Y and observation times X from the time series in the training dataset. Specifically, under the distribution $P(w \mid X, Y)$ the probability of each possible function f_w is found through *Bayes' rule*, conditioning the dataset onto the parameterisation. This posterior distribution enumerates the probability that each function f_w, parameterised by the specific parameter set $w \in W, w = (\mu, \sigma)$, generated the time series (X, Y). This posterior distribution specifies the Gaussian process, describing a conditional distribution over the underlying Gaussian functions that explain a given set of observations.

From the posterior distribution $P(w \mid D)$ conditional on the training dataset $D = (X, Y)$, the predictive distribution $P(y^* \mid x^*, D)$ can be constructed by integrating the posterior distribution over all possible parameterisations $w \in W, w = (\mu, \sigma)$ of the Gaussian function f_w describing the data. Specifically, we compute the product between the probability distribution $P(y^* \mid x^*, w)$ over outputs for each Gaussian function f_w (parameterised by $w \in W$) upon input of the test observation x^*, and the prior distribution of each parameteriation $P(w|X, Y)$. This gives the *joint conditional probability* $P(y^* \mid x^*, w, X, Y) = P(y^* \mid x^*, w, D)$ specifying the distribution of outputs y^* on test point x^* given the parameterisation w and training dataset $D = (X, Y)$. Computing the entire predictive distribution $P(y^* \mid x^*, D)$ then involves computing this joint conditional probability over all possible parameterisations $w \in W$ of the Gaussian function f_w; this is implemented through an integral over w, shown in Equation 5.

$$P(y^* \mid x^*, D) = \int_{w \in W} \big[P(y^* \mid x^*, D, w)\big] dw = \int_{w \in W} \big[P(y^* \mid x^*, w) P(w \mid D)\big] dw \tag{5}$$

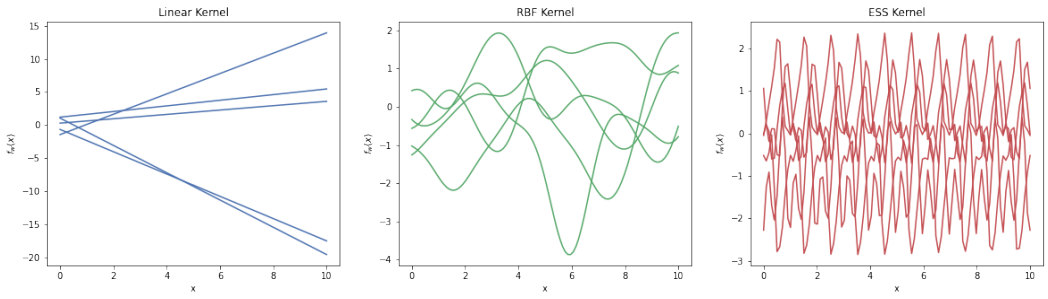
Hence, we have a probability distribution $P(y^* \mid x^*, D)$ that, after being trained upon some dataset D, can take input of a test observation time x^* and output not only the most likely value of the dependent variable y^* at this time, but a probability distribution over all values this variable could take, allowing us to compute an uncertainty interval around this predicted output. Conceptually, instead of predicting test outputs by forming a single map between values $x \in X \rightarrow y \in Y$ (the typical approach of regression), a Gaussian process makes predictions by taking every possible regression map that exists and weighting their outputs on the given input by the probability of this map being accurate to the true underlying distribution from which the data originates. Thus, functions that have a higher likelihood of fitting the data influence the prediction more than functions that aren't representative of the data, producing a result that is a weighted average over a multitude of different functions that could be close to the true distribution. Furthermore, since the individual parameters $w \in W$ have been marginalised out of the formula in Equation 5 through integration, the predictive distribution $P(y^* \mid x^*, D)$ does not depend upon any specific parameters, hence proving that the Gaussian process model is non-parametric.

2.2.3 Forming a Multivariate Gaussian Distribution

In reality, since each instance of a function in the set of varying parameterisations is Gaussian distributed (4), any linear combination of these instances is *jointly Gaussian distributed* (i.e. their joint distribution follows the Gaussian distribution), and therefore the predictive distribution $P(y^* \mid x^*, D)$ constructed through a linear combination of these Gaussian functions must itself be a multivariate Gaussian distribution (Alvarez et al. 2011). Hence, we can sample outputs of the Gaussian process with predictive distribution $P(y^* \mid x^*, D)$ by sampling from the equivalent multivariate Gaussian distribution $\mathcal{N}(\mu, \Sigma)$ parameterised by a multivariate mean function $\mu(x)$ and covariance matrix Σ_i, j. The covariance matrix is a positive semi-definite matrix (Rasmussen & Williams 2006) evaluating the covariance function $K(x_i, x_j)$ between all training and testing variables, indicating how correlated variables within the portion of the time series used for training are to each other and to variables within the testing portion of the time series. The hyperparameters μ and Σ are chosen before training and testing commence; whilst the mean function typically takes $\mu(x) = 0$, the covariance matrix is defined through a *kernel function*. This kernel function $K(x_i, x_j) = \Sigma_i, j$ describes the dependencies between variables in the time series (over the training dataset), completely defining the Gaussian process (and predictions it will infer during regression). Thus, once a kernel function has been selected the set of possible Gaussian functions $f_{w \in W'}$ can be reduced to a smaller set of possible parameterisations W', restricting the prior probability distribution $p(w)$ to only assign positive probability to functions that exhibit similar properties to the dataset (e.g. functions with a similar smoothness, roughness, or periodicity to the time series).

Therefore, determining the correct kernel for the time series at hand is vital, and consists of two steps. Firstly, the type of kernel must be selected based upon the properties of the time series being modelled: for example, the *radial basis function kernel* is commonly used for smooth time series such that similar inputs are highly correlated and thus generate similar outputs (Sollich & Williams 2005). A collection of different kernel types and the Gaussian functions they produce are shown in Figure 5; this exemplifies how the choice of kernel is crucial, as each function depicted has significantly divergent properties it can model.

Figure 5: A spectrum of different types of kernel function: the linear kernel, the radial basis function kernel, and the exponential sine squared kernel



Secondly, the hyperparameters controlling the behaviour of this kernel function must be learned from the dataset. To do this, the Gaussian process is fit to the training data such that we only retain Gaussian functions that explain the trend of the existing time series observations shown in the training data. This is typically implemented through *Maximum Likelihood Estimation* (MLE) where an optimal hyperparameter set $\hat{\theta}$ is found by evaluating the likelihood of a parameter set θ. Through Bayes' rule, the probability of the parameters θ defining the distribution exhibited in the dataset, namely $P(\theta \mid X, Y)$, is proportional to the probability $P(Y \mid X, \theta)$ of the training data being drawn from a distribution parameterised by θ. Hence, to find the optimal hyperparameter set we evaluate the likelihood $P(Y \mid X, \theta)$ over all parameter sets and find the values $\hat{\theta}$ that maximise this. Typically the log-likelihood $\ln[P(Y \mid X, \theta)]$ is maximised through Equation 6 instead of the pure probability, as this reduces the complexity of likelihood computations whilst preserving the same maxima (as the logarithmic function is monotonically increasing).

$$\hat{\theta} = \underset{\theta}{\operatorname{argmax}} \left[\ln \left[P(Y \mid X, \theta) \right] \right] \tag{6}$$

2.2.4 Making Predictions

Once we have our kernel function, the Gaussian process is completely defined as the multivariate Gaussian distribution $\mathcal{N}(\mu, \Sigma)$. Hence, to make a prediction y^* of the value taken by the dependent variable at time x^* we sample from the Gaussian distribution over all possible values output by the Gaussian functions on x^*. Specifically, to make the prediction y^*, we construct the probability distribution $P(y^* \mid Y, X, x^*)$ over possible values $y^* = f_w(x^*)$ and solve for the mean and variance of this distribution. If x^* is a single observation time, this will give a single mean value μ_{y^*} and variance value $\sigma^2_{y^*}$; the mean μ_{y^*} gives us our predicted value y^*, and the square root of the variance gives the standard deviation σ_{y^*} of the solution, allowing us to specify a confidence interval $\mu_{y^*} \pm 2\sigma_{y^*}$ around the prediction. We can also predict multiple steps into the future on input of the observation times vector $\mathbf{x}^*$; this would output the mean vector of predictions $\mathbf{mu}_{\mathbf{y}^*}$ and covariance matrix Σ used to specify the specifies the standard deviation and confidence interval of each prediction over the time series.

2.2.5 Advantages of GPR

Gaussian process regression is an effective tool when modelling time series, as it addresses many of the dataset challenges highlighted, allowing predictions to be made under circumstances that render other regression methods such as linear or logistic regression unusable. Namely, GPR can be conducted over time series that are not perfectly stationary, and datasets that exhibit dependencies and correlations between dependent variables. Moreover, whilst other regression techniques only output a prediction value without any context as to the likelihood of this prediction being accurate, GPR is effective at modelling uncertainty about the true value of a predicted variable, as the joint distribution over multiple Gaussian functions allows the computation of the confidence interval of predictions.

Non-stationarity of data is not an issue for GP models as the kernel function defining the relationship between data points can be chosen to exhibit any non-stationary trend (Bitvai 2016). For example, if a dataset demonstrated a linearly increasing trend we can construct a *composite kernel* (one that combines multiple different functions through summation or multiplication) including one function to model the trend—such as a *linear kernel* that models the relation between variables as the linear line—and another to model any fluctuations around this trend.

Additionally, GPR does not require data to be i.i.d as the kernel function can model correlations between variables explicitly through the kernel function, enumerating the correlations between all variables in the time series through the covariance matrix Σ (Roberts et al. 2013). Hence, optimising the hyperparameters of the kernel function allows for the manipulation of covariances in Σ, defining the dependencies between variables. Furthermore, the modelling of non-linear dependencies is also possible through the correct choice of kernel function (Bitvai 2016).

It is important to note that time series derived from financial markets are typically not Gaussian distributed (Bitvai 2016). However, due to the *Central Limit Theorem*, financial market prices are often modelled as Gaussian distributed. This theorem, proved by (LAPLACE 1810), states that the sum of a collection of i.i.d random variables tends to a Gaussian distribution. However, this is an oversimplification as the Central Limit Theorem only holds for random variables with finite variances (which is not the case in complex systems such as financial markets). Instead, a generalisation of this theorem is made, stating that the sum tends to a *Levy-stable distribution mainardi-2007*. Gaussian processes can be used to model these *Levy processes* as the model used is non-parametric, and the choice of kernel allows the model to take account for non-smooth fluctuations in value.

2.2.6 My Methodology

The aim of this report is the prediction of future values of the time series representing logarithmic returns of the cryptocurrency pair ETH/USD. Due to the non-stationarity and correlations in this

dataset, it is a perfect application for Gaussian process regression. Hence, my approach implemented a Gaussian process with zero mean and a composite kernel function consisting of the white noise, constant, and squared exponential kernel functions (Equation 7).

$$GP \sim \mathcal{N}(0, K_{white} + K_{const} * K_{ESS}) \tag{7}$$

This kernel utilises the *exponential sine squared function* (ESS) (Mackay 1998) (Equation 8), a commonly chosen kernel for time series regression as it permits the modelling of repetative but noisy functions: for example, Rasmussen & Williams (2006) and Roberts et al. (2013) both explored its use when modelling time series. Additionally, the kernel function can introduce a non-linear warping over its domain (Rasmussen & Williams 2006), making it well suited for modelling financial markets with non-linear dependencies. The function has two hyperparameters $\theta = (p, l)$ that we must optimise over: the periodicity p and lengthscale l.

$$K_{ESS}(x_i, x_j) = e^{-\frac{2}{l}\sin^2\left(\frac{\pi}{p}|x_i - x_j|\right)} \tag{8}$$

The periodicity p regulates the distance between repetitions in the function being modelled, and the lengthscale l determines the magnitude by which the kernel function can fluctuate around its general trend to fit fluctuations in the data. These hyperparameters are optimised through MLE; a particular challenge of this is optimising the lengthscale l, where a compromise has to be struck between smaller values that allow more detailed, rapid fluctuations around the general trend (allowing a close fit to the dataset), and larger values that give a smoother function. The high volatility of financial data means that a large the lengthscale is necessary to accurately represent the rough properties of data. However, if the lengthscale is too large, the training dataset will be *overfit*, where noise in the training data is modelled with too fine detail, reducing the generalisation performance to unseen testing data. Thus, a compromise between a good fit and good generalisability is necessary.

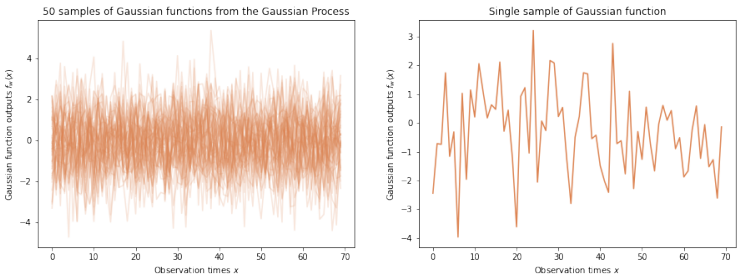
In my implementation, this ESS kernel function is multiplied with a *constant kernel* (Equation 9) to scale down the magnitude of the resulting composite kernel such that the GP outputs values within the correct range. Finally, a *white noise kernel* (Equation 10) is added to the composite kernel, which uses a random noise term $\mathcal{N}$ to allow the modelling of noise within the signal.

$$K_{const}(x_i, x_j) = c \tag{9}$$

$$K_{white}(x_i, x_j) = \mathcal{N} \tag{10}$$

This composite kernel was chosen as relying upon composite kernels is a common practice when developing GPs, as it allows models to be more generalised and robust when applied to unseen testing data. The properties of the Gaussian functions produced by this composite kernel are exemplified in Figure 6; this demonstrates the ability of this kernel to model significant and frequent fluctuations in value as the Gaussian functions produced exhibit a rough pattern of spikes and troughs of varying magnitude (similar to what we would expect from financial returns).

Figure 6: A sample of Gaussian functions from the implemented Gaussian process (after hyperparameter of the kernel have been learned, but before model training).

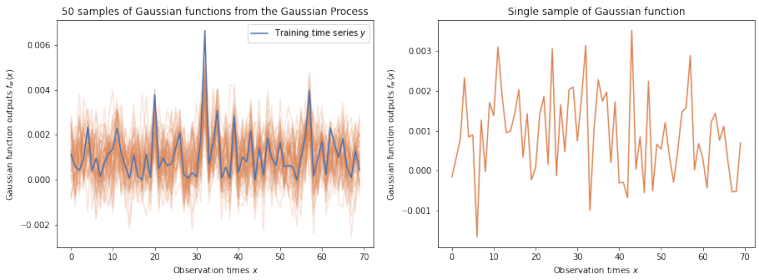


3 Results and Discussion

3.1 Fitting the Dataset

Before regression can be conducted, the model must undergo training where we fit the GP to the dataset. First, we must split this dataset into separate training and testing datasets, partitioning the time series of ETH/USD log-returns $T_{return} = (X, Y)$ into subsets separately used for training the model and testing its performance at unseen observation times. For this report, a $70:30$ train-test split was implemented, using the first 70% of the time series for training the GP and saving the final 30% as the testing set. During training, the set of possible Gaussian functions of the GP is constricted to those that describe the time series depicted in the training data. Figure 7 demonstrates this adaptation in the distribution over underlying Gaussian functions f_w; the plot shows a collection of samples drawn from the fitted multivariate Gaussian distribution of our trained GP.

Figure 7: A sample of Gaussian functions from the implemented Gaussian process (after all training).



Upon inspecting the multivariate distribution of the GP model after training, we can see how the set of possible Gaussian functions has been restricted to those that follow the trend exhibited in the time series; the remaining functions (orange lines) exhibit similar properties to the time series used to train the model (blue line), mimicking its fluctuations. Where significant increases in the dependent variable are seen (e.g. at observation times $x = 20, 32, 57$) the remaining Gaussian functions also spike in value. Comparing this distribution (Figure 7) to the distribution of functions before the model had been fit to the training dataset (Figure 6) further highlights this adaptation. The samples in Figure 6 show a wide range of different functions, including those with large amplitude, with outputs approximately ranging between $[-4, 4]$. If we compare this to the samples from the constricted function set in Figure 7 it's evident that the amplitude of functions has been significantly reduced, with most function outputs ranging between approximately $[0.005, -0.002]$. This much more closely emulates the values exhibited in the training time series, where values range between $[0.0, 0.0066]$. The plot also demonstrates that our kernel function is able to represent the rapid fluctuations of financial markets.

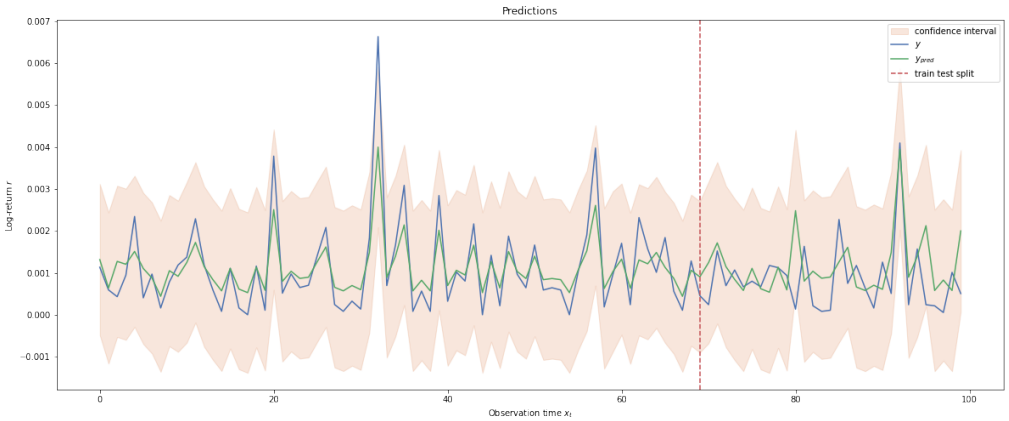
3.2 Predictions

After training, the GP model can be applied to unseen testing data to conduct GPR. This involves evaluating the multivariate Gaussian distribution at unseen observation times x^*, using the constrained set of Gaussian functions to output a mean function and covariance matrix that specify the predicted value and confidence interval of the dependent variable.

Figure 8 demonstrates this regression, showcasing the predicted mean function y_{pred} over the entire time series (both training and testing datasets). The left-hand side of this figure demonstrates the compromise reached during the training process; one can see how the general trend of the prediction function of the GP (shown in green) follows the trend of the training data (shown in blue), emulating the same peaks and troughs. However, the lines do not perfectly align, demonstrating how the GP has avoided overfitting the training data, preserving generalisability. The right-hand side of Figure 8 shows how this predicted function performs over the testing dataset. The trend of the mean function y_{pred} (in green) with respect to the true distribution y (in blue) demonstrates that the model has accurately captured the fluctuating behaviour of the time series, following similar peaks and troughs. If the trend of the predicted time series is observed closely, it can be seen how GPR has correctly

predicted the general increases and decreases of log-returns on a number of occasions: for example, a correct prediction of the spike in value at test time $x = 71$ (although overestimating the magnitude slightly), and decrease at time $x = 93$ (again, overestimating the final value's magnitude). The most impressive result occurs at time $x = 92$, at which point the GP correctly predicts the significant jump in log-return, accurately predicting the magnitude of increase. This is likely due to the similar large peak in the training dataset (at training time $x = 32$), which has caused the model to expect significant jumps in value, proving the implemented model has successfully modelled how volatile the financial time series can be. Furthermore, one can see how the GP has captured the general trend of the time series well in retained distribution of functions as the true distribution of the time series consistently remains within the confidence interval of the model (shown through the orange band), indicating that the function set has been constrained well to those that represent the time series. However, it is important to note that the model does not correctly predict the behaviour of the time series on all occasions, demonstrating the limitations of this GP. Most notably, at time $x = 80$ GPR incorrectly predicts that the log-return would spike in value, whereas the true distribution drops at this time to a value much lower than that predicted by the model. This indicates that the model has not perfectly captured all relations between variables, likely due to the complexity of the time series. This is likely due to the volatility of financial markets resulting in the distribution of values often not behaving in a predictable manner, meaning models will never predict fluctuations in value in all circumstances.

Figure 8: The predicted function y_{pred}, and its confidence interval, over training and testing data.



3.3 Error measures

3.3.1 Coefficient of Determination

Besides visual inspection of predictions, the model's performance can be tested through a range of statistical metrics quantifying the residual error ε of the regression. One such measure is the *coefficient of determination* R^2 that compares the estimated variance of the residual ε to the estimated variance of the dependent variance Y. The formula for R^2 is depicted in Equation 11 as one minus the ratio between the squared distance between the dependent variable y_i and its predicted counterpart $\hat{y}_i$, and this distance between y_i and the sample mean of the dependent variable $\bar{y}$. The resulting quantity indicates the proportion of variables y correctly represented by the predicted regression line y^*, with the value R^2 tending towards the probability of a new data point falling on this regression line.

$$R^2 = 1 - \frac{\sum_{i=0}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=0}^{n}(y_i - \bar{y}_i)^2} \tag{11}$$

$$\bar{y}_i = \frac{1}{n}\sum_{i=0}^{n} y_i \tag{12}$$

The coefficient of determination ranges between $[-\inf, 1]$, with 1 indicating a perfect score and values below zero indicating the predictions $\hat{y}_i$ are no better than the *constant model* which uses the sample mean (Equation 12) to predict values. When applied to my GP model, a score of $R^2 = 0.739$ was generated over the training dataset, suggesting a successful fit to the data (but not too well as to imply overfitting). However, when applied to the predictions made over the testing dataset, a score of $R^2 = -0.0601$ was computed. This suggests that our GPR is performing no better than the constant model. Part of this result can be explained by the variations in volatile financial time series being well described by a constant mean function (as fluctuations on either side of the mean are common and hence balance out); however, this disappointing result may partly be down to the flaws of R^2 as a measure of model performance, as the metric does not inform of whether a model is a good approximation of the true underlying distribution of the time series, it only indicates whether the model performs noticeably better than the constant model over this specific dataset.

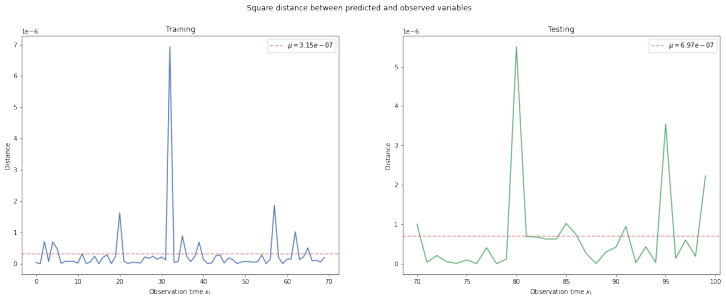
3.3.2 Root Mean Squared Error

For this reason, several other measure quantifying the residual error ε have been explored. The first of these is the *root mean squared error* (RMSE), a popular regression metric exploited in work such as Roberts et al. (2013) and Bitvai (2016). RMSE measures the magnitude of deviations of predictions $\hat{y}_i$ from the true values of the dependent variable y_i, computed as the square root of the average square distance between the true and predicted values of the dependent variables (Equation 13). Hence low RMSE values indicate a model performs well, generating predictions close to the true distribution.

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=0}^{n}(y_i - \hat{y}_i)^2} \tag{13}$$

When the RMSE is computed between the true time series used in this report and the predicted function over these observations, an impressively low score of 0.000562 is output; evaluating the RMSE of this predicted function over the testing dataset generates a score of 0.000835. Not only are these scores impressively small, with their near-zero values indicating that the model produces outputs that are very similar to those of the true underlying distribution, but the error over the testing dataset has not increased significantly from the training (a relative increase of only 1.59×). This suggests that the GP model has successfully leant to follow the general trend of the time series, and thus generalises well to unseen observation times. These results are shown in Figure 9, visualising the RMSE of each prediction made of the dependent variable during testing; this plot demonstrates how neither the range of nor the mean error value do not increase significantly during testing.

Figure 9: RMSE over training and testing instances.



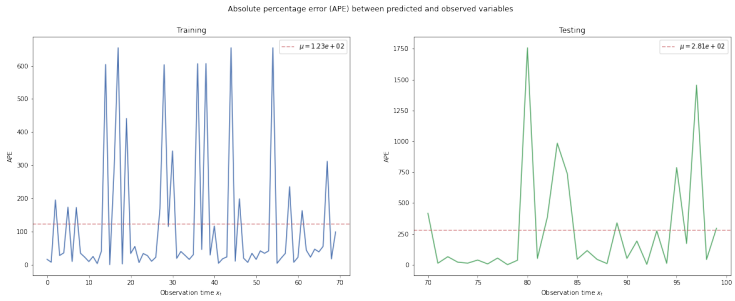
3.3.3 Mean Absolute Percentage Error

However, the RMSE metric is not perfect either; not only is RMSE sensitive to outliers in the dataset, but it is also incredibly dependent on the scale of the data, with the error increasing in magnitude alongside the increase in the magnitude of data points. This is partly the reason why the RMSE appears so low over our data, as the small log-return values (ranging between $[0.0, 0.0066]$) mean that any error in predictions will also be small in scale; thus, a scale-independent measure is preferable.

$$MAPE = \frac{1}{n}\sum_{i=0}^{n}\left|\frac{y_i - \hat{y}_i}{y_i}\right| * 100\% \tag{14}$$

The *mean absolute percentage error* (MAPE) is one such metric, computed as the absolute difference between the true values y_i and predictions $\hat{y}_i$ relative to y_i as a percentage average (Equation 14). Over the training dataset, the implemented model produced a MAPE of 122.68%; this increased to 280.74% over the testing dataset. Conceptually, this means that the average difference between the predicted value of the dependent variable and the true value is 122.68% during training and 280.74% during testing. This illustrates that our model maintains a relatively high margin of error during both training and testing; however, when we can also see that the MAPE during testing is only 2.29 times that of training, which is not a dramatic increase, meaning that performance according to this metric is still shown to generalise well to unseen data. The differences between training and testing are also exemplified in Figure 10; this plot enumerates the MAPE for each observation time during training and testing. The plot shows that whilst a few instances in the testing dataset exhibit a noticeably larger error (e.g. $x = 80$ and $x = 97$), in general, the MAPE during testing is of the same order of magnitude as that produced during training.

Figure 10: MAPE over training and testing instances.



4 Limitations and Further Work

Whilst the performance benefits of GPR have been shown, the results presented in this report clearly suggest that the GPR predictions of log-returns are not perfect. This is largely due to the difficulty in predicting the behaviour of financial markets due to their extremely high-frequency variations (Liu et al. 2020). For this reason, predictions were only made over a short window of future observation times within the time series. Furthermore, GPR can incur long training times if using a large dataset of many observations, as the covariance matrix will be large, increasing complexity (Bitvai 2016). This meant that to conduct GPR in a timely manner, only a short time series between *13:45* and *22:00* on *01-01-2022* was used, limiting the amount of data available to train the model to higher accuracy. Hence, to test the utility of GPR further a larger time series will need to be utilised for training and testing, giving a more general picture of how effective GPs are at predicting financial markets.

Additionally, there are a number of known limitations of GPs, including numerical instabilities introduced by the covariance matrix and reduced efficiency when dealing with high-dimensional feature spaces (Bitvai 2016). To address these issues, many adaptations to the basic model of GPR have been presented; this ranges from the inclusion of *jitter* in the covariance matrix (Basak et al. 2021) to the development of *deep Gaussian processes* (Damianou & Lawrence 2012) that construct a neural network where each layer is modelled by a GP, elevating the ability to learn the non-linear functions over complex datasets.

5 Conclusions

Hence, this report has shown that exploiting Gaussian processes to conduct Gaussian process regression over financial time series allows for predictions to be made over complex domains like financial markets

that prove challenging to other regression methods (such as linear and logistic regression). By relying upon a multivariate Gaussian distribution, this method can model the complex linear and non-linear dependencies between variables in datasets such as the logarithmic returns of the cryptocurrency pair ETH/USD, allowing it to accurately capture the highly fluctuating values of varying magnitude and direction typical of this volatile domain, producing a trend line that captured the noisy signal of the time series and generalised well to unseen observation times.

References

Alvarez, M. A., Rosasco, L. & Lawrence, N. D. (2011), 'Kernels for Vector-Valued Functions: a Review', *arXiv e-prints* p. arXiv:1106.6251.

Baidoo, E. & Priestley, J. L. (2016), An analysis of accuracy using logistic regression and time series, *in* 'Grey Literature from PhD Candidates'.

Basak, S., Petit, S., Bect, J. & Vazquez, E. (2021), 'Numerical issues in maximum likelihood parameter estimation for Gaussian process interpolation', *arXiv e-prints* p. arXiv:2101.09747.

Bitvai, Z. (2016), Predicting financial markets using text on the web.
URL: *https://etheses.whiterose.ac.uk/19505/*

Brahim-Belhouari, S. & Bermak, A. (2004), 'Gaussian process for nonstationary time series prediction', *Computational Statistics and Data Analysis* **47**(4), 705–712.

Damianou, A. C. & Lawrence, N. D. (2012), 'Deep Gaussian Processes', *arXiv e-prints* p. arXiv:1211.0358.

Girard, A., Rasmussen, C. E., Candela, J. Q. n. & Murray-Smith, R. (2002), Gaussian process priors with uncertain inputs application to multiple-step ahead time series forecasting, *in* 'Proceedings of the 15th International Conference on Neural Information Processing Systems', NIPS'02, MIT Press, Cambridge, MA, USA, pp. 545–552.

Han, J., Zhang, X.-P. & Wang, F. (2016), 'Gaussian process regression stochastic volatility model for financial time series', *IEEE Journal of Selected Topics in Signal Processing* **10**(6), 1015–1028.

LAPLACE, P. (1810), 'Memory on the approximations of formulas which are functions of very large numbers and on their applications to probabilities'.
URL: *https://ci.nii.ac.jp/naid/10017470740/en/*

Lin, K., Lin, Q., Zhou, C. & Yao, J. (2007), Time Series Prediction Based on Linear Regression and SVR, *in* 'Third International Conference on Natural Computation (ICNC 2007)', Vol. 1, pp. 688–691.

Liu, B., Kiskin, I. & Roberts, S. (2020), 'An overview of gaussian process regression for volatility forecasting', *2020 International Conference on Artificial Intelligence in Information and Communication (ICAIIC)* pp. 681–686.

Mackay, D. J. C. (1998), Introduction to gaussian processes.

Mushtaq, R. (2011), 'Augmented dickey fuller test', *Econometrics: Mathematical Methods & Programming eJournal* .

Phillips, P. & Yu, J. (2007), 'Maximum likelihood and gaussian estimation of continuous time models in finance', *Handbook of Financial Time Series* .

Rasmussen, C. & Williams, C. (2006), *Gaussian processes for machine learning*, Vol. 2, MIT Press.

Roberts, S., Osborne, M., Ebden, M., Reece, S., Gibson, N. & Aigrain, S. (2013), 'Gaussian Processes for Time-Series Modelling', *Philosophical Transactions of the Royal Society (Part A* .

Seabold, S. & Perktold, J. (2010), statsmodels: Econometric and statistical modeling with python, *in* '9th Python in Science Conference'.

Sollich, P. & Williams, C. K. I. (2005), Understanding gaussian process regression using the equivalent kernel, *in* J. Winkler, M. Niranjan & N. Lawrence, eds, 'Deterministic and Statistical Methods in Machine Learning', Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 211–228.

Wes McKinney (2010), Data Structures for Statistical Computing in Python, *in* Stéfan van der Walt & Jarrod Millman, eds, 'Proceedings of the 9th Python in Science Conference', pp. 56 – 61.