Comparative Analysis of Semi/Non-Parametric Regression Models on Bitcoin Price Prediction

Aviejay Paul, M.Sc. Computer Science (Data Science)

Dissertation report

In completion of the degree program in Masters in Computer Science (Data Science),

School of Computer Science and Statistics,

Trinity College,

University of Dublin, Ireland

Supervisor: Prof. Bahman Honari

August, 2021

Declaration

Aviejay Paul

August 31, 2021

Permission to Lend and/or Copy

Aviejay Paul

August 31, 2021

Acknowledgement

First and foremost, I would like to thank my supervisor, Prof. Bahman Honari, for his immense support, assistance and guidance, without which it would have been extremely difficult for me to complete the thesis in a planned and timely manner. The directions provided by Prof. Honari played a pivotal and a very significant role in churning out this piece of academic work.

Next, I would like to thank my father who always inspired me to pursue excellence of the highest order. His memories and words will always be my guiding star and pathfinder. I would also like to thank my mother and my brother for their unrelenting support in these troubled times that helped me surge ahead and produce this work.

Last but not the least, I would like to thank this esteemed institution, Trinity College Dublin, for accepting me and giving me the opportunity to pursue masters under its umbrella. I feel honoured to have been a part of such a prestigious institution and to have got the opportunity to learn from such distinguished and renowned professors.

Aviejay Paul
Trinity College, University of Dublin

August 31, 2021

In loving memory of my father, Mr. Ajoy Paul, who left us this year due to Covid.

Abstract

Bitcoin is a cryptocurrency which uses peer-to- peer technology for secure transactions based on block-chain technology. It is one of the most widely accepted crypto-currency in the world today. In this paper, we present a thorough comparative analysis of the performance of different predictive regression models (primarily non-parametric and semi-parametric) on Bitcoin price predictions. The models that we have used to perform the comparative analysis are K-Nearest Neighbours, Random Forest Regressor, Kernel Ridge Regression, Cubic Spline and Gaussian Process Regressor. We have also attempted to check which model is able to accurately predict the sudden rise in the bitcoin price using endogenous features related to price, followed by predicting the sudden slumps. We primarily make use of two columns from the dataset, which are date and closing price (USD). At the same time, we have also introduced two new features in the training data set. These new features are yesterday's Bitcoin price and the difference between yesterday's price and the price day before yesterday. Training of the model has been performed using data from the year 2013 to 2020 and the test set is the data from the year 2021. Now, since the Bitcoin price trend is extremely volatile and pattern less, it is understandably very difficult for prediction models to capture the behaviour of the data, let alone make predictions on it. Hence, we may even end up doubting the efficacy of a particular model in making predictions for time-series data. In order to avoid this, we have introduced an auxiliary time-series data set, which has repetitive patterns and can be considered an 'ideal' time-series data set. This will help us judge if a particular model is suitable to make time-series predictions or not, even if the model performance on the Bitcoin data set is poor. This data set is an electricity consumption data set that has date and electricity consumption in the columns. Also, the same kind of feature engineering as that in the case of the Bitcoin dataset has been done here. Here, the training set extends from the year 2008 to 2016, while the test set is that of 2017. We find that the Gaussian Process, when used with an appropriate kernel, and Kernel Ridge are the most effective of the studied models while predicting Bitcoin price using strictly endogenous features related to price.

Note: This work focusses on the prediction performance and analysis of different models rather than on optimization strategies used by the different models.

Keywords: Bitcoin, K-Nearest Neighbours, Random Forest Regressor, Kernel Ridge Regression, Smoothing Spline, Gaussian Process Regressor.

Table of Content

List of Tables Used

List of Figures

Chapter 1

Introduction

1.1 Overview

Distributed Ledger technology, most commonly known as block-chain technology, is a framework which enables secure transactions in a decentralized database and thus, eliminates the need for a central authority. This is the fundamental theory that works behind the implementation of Bitcoin. Bitcoin is a proper peer-to-peer electronic cash or a cash transfer system that was introduced to the world by *Satoshi Nakamoto* (pseudonymous) in his paper *"Bitcoin: A Peer-to-Peer Electronic Cash System"*. Bitcoin allows monetary transactions to take place without the intervention of any middleman financial institution. The bitcoin network is also considered safe from attackers and also does not require a lot of infrastructure. At the same time, the nodes present in a Bitcoin network can join or leave the network as and when required. Bitcoin also solves the double spending problem[1]. Double spending is a flaw present in digital cash transactions in which the same digital token is used more than once that leads to creating new currency that did not exist previously [2].

But the main problem is that the bitcoin price has been fluctuating very rapidly. This is because the price of Bitcoin, unlike that of stocks and shares, lacks the pillars that otherwise hold financial assets together and thus, lacks the support of any other ground that would make the price fluctuation more stable. Hence, any external factor that could easily change the demand and supply of Bitcoins would significantly affect Bitcoin prices. In the last few years, bitcoin price has been extremely volatile and the bitcoin market has been even more unpredictable due to Covid. The volatility of Bitcoin price can be understood from the fact that it shot up from around $10,000 to $65,000 in a span of only 3-4 months. More interestingly, the price fell as quickly after the peak.

Previously, various machine learning methods have been introduced to forecast bitcoin price. However, not many of the models have been very successful in accurately predicting the Bitcoin price. Time-series forecasting involves correctly predicting future behaviour of time-series data by carefully analysing it. The kind of models (viz. nonparametric and semi-parametric) that are to be implemented depend on what information we have about the model function. It is to be noted that parametric regression models are the ones that are most frequently used. These models can be both linear and non-linear and are comparatively easy to work with since there is a sufficient amount of prior information regarding the function that will be used. However, in this project, we will concentrate on the performance of different predictive non-parametric and semi-parametric regression models in predicting Bitcoin price. We will present a comparative analysis of the model performances along with reasoning as to why certain models perform better than the others.

1.2 Motivation

Bitcoin is one of the most widely used cryptocurrencies today. However, the amount of its market volatility and the amount of non-periodic fluctuations in its price, caught my attention. Time series predictions can be made comparatively easily for data that has a periodic trend. However, Bitcoin price has an extremely uneven trend and does not follow any specific pattern. As a result, making correct predictions on Bitcoin prices is not an easy task. If Bitcoin has to be studied as an investment asset, it is very important to be able to correctly predict the future trend of its price. Most of the earlier Bitcoin studies have focussed on the technical aspects of Bitcoin and its network. In contrast to that, this work will focus on looking at Bitcoin from a financial aspect and will attempt to introduce some degree of predictability to it.

At the same time, it will be very interesting to note the performance of various predictive regression models for Bitcoin price prediction. Different models work on different principles and not all model will have a high degree of performance while making Bitcoin predictions. Also, it is to be noted that non-parametric models capture the behaviour of the data directly from the training set and not by using any pre-defined function. Hence, it will be very interesting to note which model is able to capture the behaviour of the data most accurately. Hence, a motivation for me to work on this project is that if a high level of certainty can be brought about in the Bitcoin price predictions, this work could contribute in the direction of making Bitcoin a global currency. There have been many researches in predicting Bitcoin price, some of which have been discussed later.

On a personal note, I have been very keen to work on this project owing to my tremendous interest in implementing Data Science concepts in the domain of finance. Since, I take a keen interest in stocks and debentures, I have always wondered if some sort of accurate predictability can be built on stock prices. Since, most stock price data show similar kind of non-seasonality behaviour as that of bitcoin prices, the knowledge, experience and findings from this project will help me tremendously to understand and build models that would be able to make accurate stock price predictions, even though there are several economic and political factors that affect stock prices, unlike in the case of Bitcoin. This project will also help me understand the world of cryptocurrencies better from an investment asset point of view and the extensions of this project will also help me in identifying more stable cryptocurrencies and the factors that lead to this stability, a work that I wish to pursue in the future. At the same time, I am looking at non-parametric and semi-parametric models in this project because I have extensively used different parametric models in the past and have a good idea on what kind on predictions will these models make on Bitcoin prices. However, I have not explored the other 2 kinds of regression models too much on time series data and hence, would like to find out more about their performance on Bitcoin price data and understand whether these kinds of models are better at making predictions on non-periodic time-series data than on data with some seasonality. In a nutshell, this work can be considered the first building block of my future research on cryptocurrency and stock price predictability.

1.3 Objectives

The primary objective of this work is to analyse which of the selected predictive regression model is most suitable to make predictions for Bitcoin prices, while strictly using endogenous features related to price. The aim of this study is to serve as the ground work for my future research work on how non-parametric and semi-parametric models can be used to predict the behaviour of crypto currencies and stocks. As a result, the simplest of the available datasets have been chosen to understand how maximum information can be extracted from minimalistic data. Once this is done, identifying other relevant endogenous and exogenous features and adding them to our model will hugely limit our future challenges. We will try to look at which model is able to precisely extract the behaviour of the training data and is then able to make precise predictions. The models that we will be using for evaluation are K-Nearest Neighbours, Random Forest Regressor, Kernel Ridge Regression, Smoothing spline and Gaussian Process Regression (GPR). In case of GPR, we will also be testing the model performance using various kernels. It is to be noted that the period of the year 2021 will not be fed into the model for training purpose and will simply be used as the validation set. This will give us an authentic idea about the model performance. Also, model parameters will be selected using proper cross validation and not blindly.

Another important objective of this project would be to check if the models are able to predict the sudden rise and the subsequent plunge in the Bitcoin prices in the year 2021. The increase in the Bitcoin price and then the sudden crash is called a bubble. We will also try and figure out which of these models is able make precise predictions regarding the formation of a bubble. Having this capability is very important if we have to study the Bitcoin behaviour from an investment asset perspective.

1.4 Dissertation Structure

The following structure will be followed in the thesis:

1 **Chapter 1:** In this chapter, we will introduce the thesis and explain what the thesis will be all about. This includes explaining the overview, motivation and objectives of the thesis.

2 **Chapter 2:** In this chapter, we will look into the background of the thesis, that will involve understanding Bitcoin concepts from an investment and a technological point of view and looking into the difference between parametric, non-parametric and semi-parametric regression models. At the same time, some of the related previous work that were looked into as part of the literature review will be presented in this chapter.

3 **Chapter 3:** In this chapter, we will look into the algorithms and theoretical concepts behind the different chosen models.

4 **Chapter 4:** In this chapter, we will look into the methodologies that will be implemented for getting the predictions from the different models and also the model results.

5 **Chapter 5:** In this chapter, we will look into the scope for future work and the conclusion drawn from the project.

Chapter 2

Background and Literature Review

2.1 Cryptocurrency, Blockchain and Bitcoin

The concept of cryptocurrency was introduced to the world way back in 1998 by a person named *Wei Dai*. As the name suggests, cryptocurrency is nothing but currency that is encrypted and was developed to be used in the form of digital currency. It has today become an important financial commodity in the international market. Many cryptocurrencies have been developed since the development of Bitcoin in 2009. But the prices of all these cryptocurrencies are extremely volatile and keep changing dramatically. This affects the behaviour of the cryptocurrency investors. Even though many researchers have tried to develop an in-depth understanding of the cryptocurrency market, there is still a long way to go, especially compared to our understanding of other financial assets and the way they work [5].

However, what we do know with absolute certainty is that this kind of currency is only virtual and does not really exist in the form we know currencies to exist in. In other words, cryptocurrency is a form of digital currency and because it is encrypted, it is very difficult to be tampered with. These currencies do not find the backing of any recognised financial institution or person. These are also not issued in the manner in which stocks or bonds are issued. These currencies simply exist and are generated according to the algorithm that generates them. Cryptocurrencies are still considered an asset because they have some financial value, which can be exchanged for some other cryptocurrency or some real currency. At the same time, cryptocurrency units are called coins and their ownership can be transferred from one account to another. Cryptocurrencies enable unique monetary transactions anywhere across the world without having money to be physically sent or received [6].

When these currencies move between accounts, they result in financial transactions. These transactions are recorded in ledgers called "Blockchain". Bitcoin is a type of cryptocurrency that works on Block chain technology. Blockchain is a distributed public ledger and all transactions that have taken place and committed are stored in list of blocks. As newer transactions take place, new blocks are appended to the chain. These blocks use cryptography as means of providing security and use distributed consensus algorithms for maintaining ledger consistency. Blockchains are decentralised, maintain anonymity and can be audited, Since, blockchains allow financial transactions to take place without the intervention of any middleman institution, this technology can be used as digital financial assets, remittances and online payments, among other fields such as IoT and security services. The transactions, once packed into a blockchain block, cannot be changed and thus, can provide a high level of security. Also, because the blockchain network is distributed, a single point of failure can be avoided [3].

However, there are certain issues that blockchains still face. First and foremost, scalability is an issue with blockchain. When the rate of transaction information coming in increases, block size will have to be increased that could lead to more storage space. This will eventually lead to slower network transmission. Next, in case of Bitcoin mining, miners find ways to hide the mined Bitcoin blocks so that they could be sold once the prices have increased further. This could result in frequent branching and hinder the development of the blockchain. Also, Bitcoin may not be as fool proof from a security point of view as it may seem. It has been shown that information leakage could take place when transactions are taking place [3].

In Fig. 1, we can see that blockchain is formed by attaching one block after the other using the parent block hash. Each block header stores the previous block hash and thus contains the information regarding the parent block. These blocks store all the transaction records. The first block in a blockchain is called the genesis block [3].

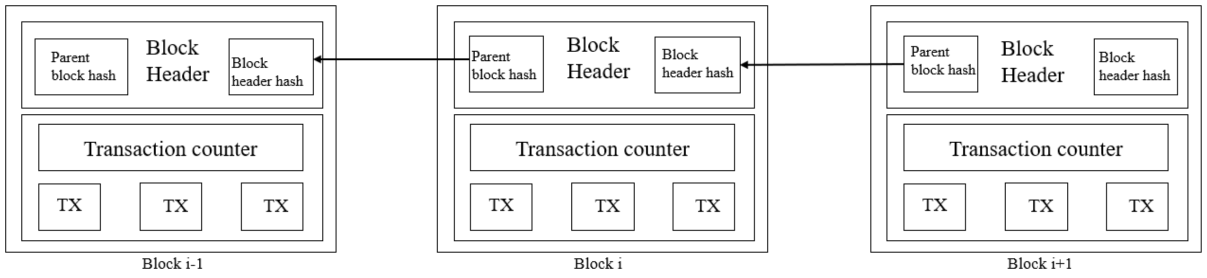

Figure 1[3]: Diagram showing blockchain

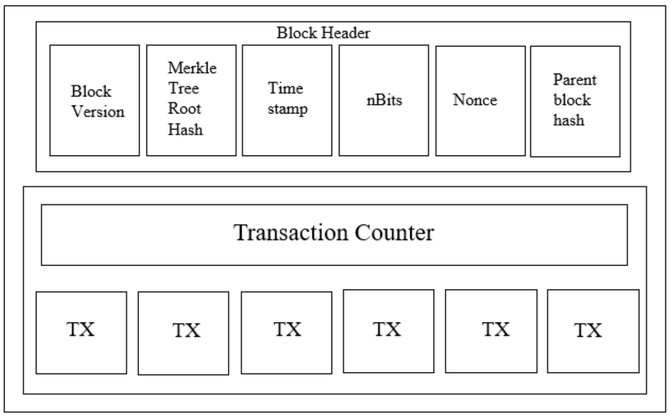

Figure 2[3]: Diagram showing a block structure

In Fig. 2 above, we can see that a block header constitutes the following information [3]:

i. **Block version**, which decides the set of block validation rules to be followed and is dependent on what the blockchain will be used for such as cryptocurrency, blockchain for industry.

ii. **Merkel Tree Root Hash**, which contains the hash value of all the transaction records present in the box.

iii. **Timestamp**, which contains the current time, is used as proof of the time at which that particular block was used.

iv. **nBits**, which identifies the complexity and the computation power that would be need to mine in a particular network.

v. **Nonce or number only used once**, which is of 4-bytes and increases from 0 with every hash calculation. This information is used by the miners during the mining process.

vi. **Parent block hash**, which is a 256-bit hash value that points to the previous block.

Each transaction is signed using digital signatures, which uses public key cryptography. The digital signatures help to establish the authenticity of the account owner and also the owner's authorisation to spend the funds. Also, integrity of the transaction data can be maintained using these digital signatures. Each user in the blockchain network has a pair of public key and private key. The public key of each user, which is derived from his/her respective private key using some mathematical function, is available to everyone on the network. In the signing phase, the message is signed using the sender's private key. Now, because the public key of the sender is available to all the nodes in the network, any node in the network can verify the digital signature and thus, can verify the authenticity of the sender. Also, the integrity of the message can be verified using the digital signature since the signature is mathematically bound to the message and also to the private key of the sender. This signature will change if the data has been tampered with and validation of the data integrity will fail.

At the same time, the sender encrypts the data using the public key of the receiver. Upon receiving the encrypted data, the receiver decrypts the data using the own private key.

There are 3 types of Blockchain categories: 1. Public blockchain, 2. Private blockchain and 3. Consortium blockchain. This division has been broadly created on the basis of how the network nodes are allowed to join the consensus process. The consensus process in blockchain is the procedure using which the nodes connected to the network at a certain point in time reach a common agreement about the current state of the ledger. In this manner, trust can be established between the network peers and also every new block added to the blockchain network can be agreed to be the one true block. In case of public blockchain, all the peers present in the network can take part in the consensus process. Also, all the transaction records are visible to all the peers. In case of private blockchain, the consensus process would be open to only those nodes that come from one particular organisation. This is a centralised network and is under the complete control of the organisation. Lastly, in case of consortium blockchain, the network is built by several organisations and only a small group of pre-selected nodes would be allowed to join the consensus process [3].

There are many algorithms such as PoW (Proof of Work), PoS (Proof of Stake), PBFT (Practical Byzantine Fault Tolerance) and many more using which consensus can be established in the network Different cryptocurrencies use different consensus algorithms. However, in this study, we will concentrate only on PoW since this algorithm is the one that is used in the Bitcoin network. In this algorithm, the node that wants to publish a transaction needs to perform a huge amount of task [3]. This huge amount of task comes in the form of a complex and a computationally intensive mathematical puzzle [4]. This is done so that the block created can be included in the ledger using the block header. This process is called mining and the nodes that carry out the mining task are called miners. The mathematical puzzle is such that the resultant answer is a hash value of a block header. The nonce, present in the block header, is constantly changed by the miners to arrive at different hash values. Once the miner has arrived at the correct hash value, the block is immediately broadcasted to other nodes in the network. These nodes then verify if the hash value is correct or not. Once the hash value is declared to be correct, the miner receives a certain number of cryptocurrencies. At the same time, this block also gets appended into the respective blockchains of the other miners [3]. The number of bitcoins available to miner per block halves every four years. As of 10th August, 2021, there are 6.25 million bitcoins available per mined block.

There are many theories behind the inventor of Bitcoin, *Satoshi Nakamoto*, not revealing his identity. One important theory that comes to light is that Nakamoto wanted to instil people’s faith in his creation. Nakamoto did not want people to think that the creator of Bitcoin would be earning profits out of it. Bitcoin was launched right after the global financial crisis of 2008-2009 when people had lost faith in centralised banking institutions. This could be the reason behind why Nakamoto made the

Bitcoin network decentralised [7]. Bitcoin provides many advantages over normal currency because it can be easily transferred anywhere in the world through the internet and also maintains anonymity of the account owners. At the same time, Bitcoin is super transparent because every node in the network has a copy of the public ledger. However, *Kaushal et al.* [7] identifies volatility and degree of acceptance as the major factors hindering the growth of Bitcoin. *Pavel et al.* [8] analysed whether Bitcoin had the potential to become a global currency and found that price volatility stops it from widespread market acceptance.

Bitcoin works using the “Blockchain technology” and the “Proof of Work (PoW)” consensus algorithm. Bitcoins are created through mining in which transactions are verified by the nodes (or users) in the network and recorded into a public ledger called blockchain. The users are in turn rewarded with bitcoins and transaction fees for their service. The computation problem in the PoW algorithm that the miners try to solve using huge computational power is given as [7]:

Find nonce such that:

H(nonce || prev_hash || tx1 || tx2 || ……... || txn) = Output hash with some leading zeros

where, H is the hash function, pre_hash is the hash of the previous block, tx1- txn are the transactions that have not yet been included in the block. Bitcoin resolves the problem of double spending since it uses Blockchain technology. This problem occurs in digital currency transactions when the same digital currency can be spent twice. However, if the original digital currency transaction details can be copied and broadcasted, the same Bitcoin can be used multiple times. Since, Bitcoin maintains transaction logs in the form of blockchain, each transaction can be authenticated and double spending is prevented.

Bitcoin has a financial value since it can be used to purchase goods and service if it accepted by the seller. Bitcoins can be bought using standard global currencies if approved by the respective governments. Since Bitcoins are decentralised, Bitcoin transactions can take place without the intervention of an intermediate financial institution. However, there are conflicting views among economists and financial institutions regarding whether Bitcoin satisfies the criteria of being a currency [8].

2.2 Data Analytics of the Price Volatility in Bitcoin

Bonneau et al. [9] correctly points out that we still have very limited understanding of Bitcoin in a way that we still might not be in a position to create a better cryptocurrency than Bitcoin. At the same time, it is still unclear if a different consensus design would be able to provide more stability and efficiency to Bitcoin as a functional currency. There are many such Bitcoin parameters that still needs more research. In a way, it is difficult to judge whether Bitcoin's success, though limited, is due its design or simply because it is the first cryptocurrency. *Liang etal.*[10] analyses the clustering structures for cryptocurrency, stocks and fiat currency (or foreign exchange such as U.S. Dollars, Indian Rupee, Euro) in order to classify the economic activities of these financial assets and identify which of these assets dominate the financial market. They observe that the foreign exchange clusters are consistent with the geographical regions, meaning that currencies belonging to the same geographical region fall in the same cluster. In case of stocks, the clustering is formed with respect to their business domains. However, no particular clustering rule was observed in case of cryptocurrencies. Also, the cryptocurrency clusters kept changing more frequently than the other two financial assets. Thus, foreign exchange and the stock markets are more robust with respect to the cryptocurrency market. This raises doubts regarding the stability of cryptocurrencies, including Bitcoin.

There are many financial and economic factors, including price volatility, relating to Bitcoin that needs to be investigated before it can be considered a global currency. However, this work will focus on the price volatility of Bitcoin since we are trying to understand the nature of the Bitcoin price and make predictions on it. Also, price volatility is the feature where Bitcoin differs from global currencies and other more popular financial assets the most. Price volatility of Bitcoin is very high within short spans of time. This can be seen from Fig.3 which shows the manner in which Bitcoin prices have been changing since mid-2017 (It is to be noted that the y-axis scales by 10,000$). This kind of massive volatility decreases the effectiveness of Bitcoin as a currency because this would lead to direct or indirect losses to businesses. On the other hand, global currencies such as Indian Rupee has on changed from around 0.016$ in 2013 to 0.013$ in 2021. In the economics world, such a small change too is considered high volatility, so one can understand the situation with Bitcoins. It is to be noted that the Bitcoin price volatility prior to mid-2017 is also considerable. However, because of the large scaling in the y-axis, this volatility is not getting captured and highlighted in figure 3. Businesses using Bitcoin as a mode of transaction would have to frequently change the value of their goods and services in order to keep the prices relevant with regards to the Bitcoin price changes. Of course, there are measures such as market exchange pricing and instantaneous exchange facilities that can reduce the risk to businesses due to Bitcoin price volatility [8].

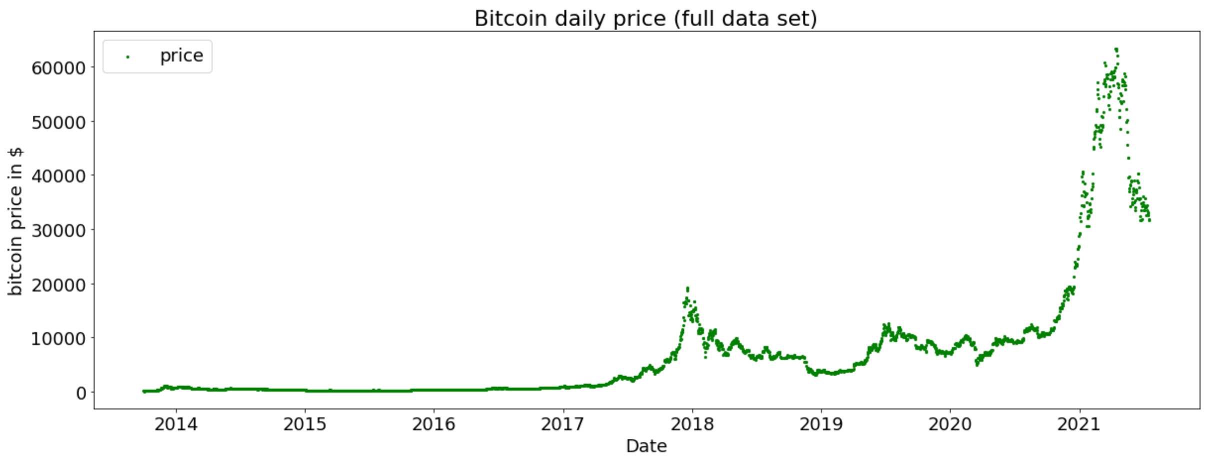

Figure 3 showing price volatility in Bitcoin

So, in order to understand the reasons for such high price volatility in Bitcoin, we need to understand the drivers of Bitcoin prices. *Pavel et al.* [8] mentions 3 such drivers that affect bitcoin prices:

1 **Supply and Demand of Bitcoin**: Bitcoin demand is estimated using 2 factors: 1. Volume of Bitcoin usage in exchanges and 2. Rate at which Bitcoin is circulating. This rate is a measure of how frequently is a unit of Bitcoin used to buy goods and services. As the rate of bitcoin circulation increases, the Bitcoin price also decreases. But as the Bitcoin usage increases in the exchanges, the Bitcoin price subsequently increases. Also, the demand for Bitcoin depends on how dependable it is as a medium for exchange. Because Bitcoin is a virtual currency and does not exist in reality, it does not have any intrinsic value. Therefore, the demand for Bitcoin depends only on its estimated future valuation and not on any kind of intrinsic value. However, this demand can be artificially changed easily by changing the future expectations from Bitcoins, such as by declaring more acceptance for Bitcoin in the future. On the other hand, the supply of Bitcoin is estimated by determining the amount of Bitcoin in circulation, which is declared publicly. This, in turn, depends on how frequently is Bitcoin getting mined.

2 **Attractiveness of Bitcoin:** The more attractive Bitcoin is in the investment market, higher will be the price. There are several factors that determine how attractive Bitcoin is at a certain point in time. These are:

i. Attractiveness of Bitcoin can be affected by the risk involved in the entire system. As mentioned earlier, Bitcoin does not have any intrinsic value. Thus, its future valuation will depend on the belief that Bitcoin will remain relevant as a financial asset in the future. If Bitcoin loses its credibility in any manner, its price will fall drastically.

ii. Next, the Bitcoin network has to be safe from cyberattacks. Many cyberattacks have occurred on the Bitcoin network in the past and this is a major hindrance in uplifting the confidence of participants in the financial market on Bitcoin since these attacks have the potential to de-stabilise the Bitcoin system.

iii. Bitcoin attractiveness can also be impacted greatly by its portrayal in the news and social media. Positive portrayal will definitely boost the Bitcoin prices and vice-versa. Also, the mere presence of Bitcoin in media can affect its price because it reduces the information search cost of potential investors. At the same time, attractiveness towards of a financial asset can be both positive and negative with respect to the kind of sentiments floating in the news regarding the financial asset. Hence, attractiveness can influence Bitcoin prices both ways in this regard.

3 **Global Economics and Financial Developments:** Global economics parameters such as oil prices, currency exchange rates can also affect Bitcoin price. Such parameters, if present in suitable conditions, can push the use of Bitcoins in commerce, resulting in higher demand and eventually higher price. At the same time, parameters such as stock price can have both positive and negative impact on Bitcoin price. If the stock prices decrease, this could lead to investors pulling out money from the stock. If Bitcoin seems to be a viable investment option at that point in time, this money could be pumped into buying more Bitcoin and thus, increasing the demand. This could increase the Bitcoin prices and hence, is a negative relationship with stock returns. At the same time, if stocks yield good results, investors could take some risk and put in some of the profits in Bitcoin, thus increasing the Bitcoin price again. This shows a positive relation between stock returns and Bitcoin price.

Thus, *Pavel et al.*[8] proposes the following model that takes into account the above 3 drivers to get the Bitcoin price. The model is shown below:

$$p_t = \beta_0 + \beta_1 p_t^{G\&S} + \beta_2 y_t + \beta_3 v_t + \beta_4 b_t + \beta_5 a_t + \beta_6 m_t + \varepsilon_t,$$

where, subscript t denotes any time 't', p_t is the price of Bitcoin at the time 't' in dollars, $p_t^{G\&S}$ is the price of the goods and services in the economy at time 't',y_t is the volume of Bitcoin usage in the exchanges at time 't' (or size of the Bitcoin economy),v_t is the rate at which Bitcoin is circulating at time 't', b_t is the total amount of Bitcoin in circulation in the market at time 't'. These variables account for the demand and supply of Bitcoins in the market. a_t is the parameter that captures the attractiveness of investing in Bitcoin at time 't', m_t is the parameter that captures the global economics and financial parameters at time ‘t’ and ε_t is an error term. It is to be noted that the given model is clearly a parametric model, whose details we will see later [8].

Now, higher the prices of goods and services, higher is the Bitcoin price. This is because a higher amount of Bitcoin will be used in the transactions, thus inflating the demand for Bitcoins. Thus, there is a positive correlation between price of goods and services and Bitcoin price and hence, β_1 is expected to be positive. Next, bigger the Bitcoin economy, higher is the demand for Bitcoin that would lead to increase in Bitcoin price. Hence, here too is positive correlation between volume of Bitcoin usage in the exchanges at time ‘t’ and Bitcoin price at time ‘t’. Therefore, β_2 can also be expected to be positive. We have already seen that as the Bitcoin circulation rate and the amount of Bitcoin circulation in the market increases, the Bitcoin price decreases. This is a negative correlation and hence, β_3 and β_4 should be negative. Now, it is to be noted that the amount of Bitcoin in circulation is largely pre-defined and the variable is exogenous too since it depends on how many Bitcoins are being mined and brought into the market. Hence, β_4 will be such that it is statistically insignificant. β_5 captures the behaviour of Bitcoin attractiveness in the model. As discussed above, attractiveness can positively or negatively impact Bitcoin price and hence, β_5 can be either positive or negative. Lastly, β_6 represents Global economics and Finance. Similar to a_t, we have seen that m_t also affects Bitcoin prices both positively and negatively. As a result, β_6 can be both positive and negative [8].

There are certain intrinsic issues with the data, as analysed by *Pavel et al.*[8], that are handled separately. The first issue is that the explanatory variables are mutually correlated. Therefore, there will be the issue of endogeneity. Endogenous time series models are those in which an input variable is affected by other variables in the system. Next is the issue of stationarity in the data. This means that the data does not have any kind of seasonality. And making predictions on stationary data could lead to misleading results. These issues are tackled using various techniques, which are not in the scope of this study. $p_t$and $p_t^{G\&S}$are determined in terms of dollars. The number of Bitcoins mined has been taken as the total number of Bitcoins b_t in circulation. Next, in order to determine the size of the Bitcoin economy, y_t, total number of unique Bitcoin transactions per day and total number of Bitcoin addresses used per day are estimated. Then, instead of using the rate of Bitcoin circulation directly, an alternative is used. This proxy is the number of Bitcoin days destroyed for a given transaction that is calculated by multiplying the number of Bitcoins in a given transaction and the number of days since the involved coins were last used. In order to capture the data for Bitcoin attractiveness, a_t, the number of online views about Bitcoin in Wikipedia is taken into consideration. At the same time, the number of new members and new posts on online Bitcoin forums are extracted from *bitcointalk.org*. Finally, to account for global economics and financial developments, m_t, oil prices are used since oil price is one of the most (or the most) important factor that drives global market [8].

With regards to the result of the impact of the drivers, it is found that demand has more impact on the Bitcoin prices than supply. Supply is characterised by number of Bitcoins in circulation whereas, demand for Bitcoin is determined from total number of unique Bitcoin transactions per day, total number of Bitcoin addresses used per day and number of Bitcoin days destroyed. The following observations were noted for the price drivers through the respective coefficients of the variables [8]:

1 Matching the expectations, it is observed that as the Bitcoin stock (supply) increases, the Bitcoin price decreases. On the other hand, as the size of the Bitcoin economy (demand) increases, the Bitcoin decreases. However, it is observed that the number of transactions, which forms a basis for the Bitcoin economy, has very less significance on the Bitcoin price.

2 Next, Bitcoin attractiveness has a significant impact on Bitcoin price. As mentioned above, this driver is measured using number of online views about Bitcoin in Wikipedia, and using the number of new members and new posts on online Bitcoin forums. Surprisingly, it is observed that new members have a negative effect on the Bitcoin price. This means that members in the Bitcoin forums ought to be information driven. At the same time, number of new posts has a positive impact on Bitcoin price, implying that more discussion on Bitcoin results in more trust in Bitcoin among investors, resulting in higher demand and price. It is also observed that number of Wikipedia views on Bitcoin too has a positive impact on its price. In this case, there could be multiple interpretations. An increasing number of views about Bitcoin on Wikipedia could mean that this was fuelled by some positive or negative new regarding Bitcoin in the news media. However, since the impact is positive, it means that the positive sentiments in the new media overtake the negative sentiments, which are usually regarding security of the Bitcoin network. Number of Wikipedia views could also mean an increasing interest amongst investors in Bitcoin, thus showing more acceptance of the asset. However, it is to be noted that these investors who go into Wikipedia to search for more information on Bitcoin would be new investors since experienced Bitcoin investors are more likely to search for information in websites that are more specialised on Bitcoin. In a nutshell, the positive impact of a greater number of Wikipedia views regarding Bitcoin could either mean verifying Bitcoin related positive sentiments by users or simply greater interest in Bitcoin amongst new investors.

3 Increase in global oil price or exchange rates could lead to inflation and thus less money amongst investors to invest in Bitcoin. This leads to less demand and lower Bitcoin price. However, it is observed that the impact of global economy and finance indicators do not have any significant impact on Bitcoin price.

Thus, is can be said conclusively that Bitcoin is a sentiment driven financial asset, unlike stocks and foreign exchange. Other factors such as demand and supply and global economics do not have a significant impact on Bitcoin price. This means that there is a high possibility that Bitcoin price will remain quite volatile even in the future since sentiment is a parameter that is very sensitive and does not take long to alter given the strength of social and new media today to influence market opinion.

2.3 Previous Work on Bitcoin Predictions

There has been a significant amount of study done on Bitcoin price prediction. The researchers have studied different model and approached and have studied different factors that could be used to predict Bitcoin prices. *Rathan et.al* [12] discusses decision tree and linear regression techniques that can be used to predict bitcoin prices. They managed to attain accuracies of over 95% in both the models and conclude that the linear regression model outperforms the decision tree model. The features that they use to make predictions are Time Stamp, opening price, Bitcoin High price, Bitcoin Low price, closing price, Bitcoin volume traded and Bitcoin value traded. However, no analysis of the variables has been presented. *Velankar et.al* [11] attempts to study the daily bitcoin trends with a focus on the different factors that affect bitcoin prices. The features that they use in their model are block size, total bitcoins mined, daily high and low values of the bitcoin, total number of unique bitcoin transactions and trade volume. The models that they propose to study the trends are Bayesian Regression and Random Forest. But the authors present no significant result or conclusion. *Joshila et al.* [13] use SVM to make Bitcoin predictions and attain an accuracy of about 70% on the test data. *Rane et al.* [14] study the Bitcoin data under numerous models such as Auto-Regressive Integrated Moving Average (ARIMA), Linear Regression, Binomial Generalised Linear Model (BGLM), Support Vector Machine (SVM), Long Short Term Memory Network Model (LSTM) and Non-linear Auto-Regressive with Exogenous Input Model. The authors get accuracy levels in the range of 51%-57%. Again, not much analysis of the models and that of the results have been presented. *Phaladisailoed et al.*[15] use different regression models such as Theil-Sen and Huber regression models and deep learning regression models such as LSTM and GRU models. The deep learning regression models run for 480 epochs with batch size of 160. It is observed that the accuracy of all the 4 models is very high with MSE in the order of 10^{-6}.The variables used in study are same as those used in [12]. *Ferdiansyah et al.* [16] use LSTM model to make Bitcoin prediction. They train the data for different numbers of epochs and the best result comes out for 500 epochs with RMSE = 288.59, which is declared not good. The researchers attribute this to local and global political and economic issues which have not been considered in the study. *Tandon et al.* [17]use LSTM with the motive of studying the effect of k-cross validation on the Bitcoin data and observed that a 10-cross validation does help in improving accuracy of the model by a good amount. *Roy et al.*[18] studies the performance of ARIMA, Auto Regressive model (AR) and Moving Average model (MA) on Bitcoin price data, with focus on ARIMA. The researchers carry out the Augmented Dickey Fuller test in order to check the stationarity of the data with the NULL hypothesis as “data has a unit root and is non-stationary”. Upon receiving a p-value of 0.999, they conclude that the data is non-stationary. This is contradictory to our initial understanding of the data and observation. The study does not produce details of the test and hence, the result of the test will not be considered in our study. The researchers observe that the accuracy of the ARIMA model, which is a combination of the AR and the MA model, is better than the other two. *Nithyakani et al.* [19] evaluate the performance of a Bi-directional LSTM and have come up with a MAPE of 13%. *Hashish et al.* [20] propose a hybrid model of Hidden Markov model and LSTM. The researchers here use a set of features that can be categorised into 1. Orders and Trades, which are features that have been extracted from the raw data and 2. Technical indicators, which have been computed from the historical data to predict the price movement. The authors observe that the performance of the hybrid model is very good with MAE of 2.5 and better than the performance of the standard time-series models such as LSTM and ARIMA. *Wu et al.*[21] perform an Augmented Dickey Fuller test on the data to check its stationarity. The researchers observe that the test produces a unit root and hence conclude that the data is stationary, which is at par with our initial observation regarding the stationarity. Hence, here we reject the conclusion of the ADF test of the test done by *Roy et al.*[18]. [21] conclude that LSTM with Auto Regressive AR(2) model perform better than the convectional LSTM model. Auto Regressive AR(2) model is the kind of model in which the present value in the time series is calculated from the previous 2 values. *Adegboruwa et al.* [22] studies the manner in methods in which time series inputs can be modelled in LSTM. These are some research works that mostly study the endogenous factors that determine the Bitcoin price trends. *Anupriya et al.* [32] use ARIMA to make Bitcoin predictions. Mean error in this case has been found to be less than 6%. But it was observed that almost all of these studies do not present their research in detail or provide a plot of the prediction over the test data, something that would have further helped to visualise the accuracy of these works. At the same time, none of these works concentrate completely on analysing the performance of non-parametric regression models in predicting Bitcoin price, which is the aim of our study. *Albariqi et al.* [31] is one of the few works that present a thorough representation of the task performed. The researchers use Multi-Layer Perceptron and RNN in their study. To select the hyperparameter values of learning rate, number of hidden node and learning algorithm, cross validation has been performed and the results have been provided. Also, a comparative study has been performed to select the optimisation strategies amongst Adam, AdaDelta, RMSProp, and Stochastic Gradient Descent (SGD). The researchers conclude that the models achieve better accuracy with long term predictions rather than short term predictions. At the same time, the researchers also conclude that MLP achieves better accuracy than RNN.

The next few consulted papers examine text to perform sentiment analysis in some way or the other to form endogenous features that can be fed into predictive models. *Pavlyshenko et al.* [23] analyses how combining a term that describes the dynamics of model deviation from real values with a regression model can help to make better Bitcoin price predictions. This is done using expert's opinion where an expert, who understands investor behaviour, defines the points of the local extremes that serve as the pivots for the deviation. Also, the Bayesian approach has been investigated to use probabilistic concepts in order to capture the outliers in the Bitcoin price data set. This study also takes into account google search trends. *Yao et al.* [24] studies the influence of news articles on Bitcoin price. They perform a sentiment analysis and conclude that such articles do influence Bitcoin price. They also propose a new technique to extract features from the news articles called sentigraph and conclude that this approach yields better results than the conventional techniques. *Sattarov et al.* [25] study the impact of public opinion on twitter on Bitcoin price. The researchers conclude that Bitcoin price can be predicted with around 63% accuracy when analysed with Twitter feed using Valence Aware Dictionary and Sentiment Reasoner (VADER) and with historical Bitcoin (closing) price. The researchers found a strong correlation between Bitcoin price and Twitter sentiment using a Random Forest Regressor model. *Pant et al.* [26], too, perform sentiment analysis on Twitter. However, they use Recurrent Neural Network (RNN) to build the prediction model and use Random Forest to choose between the different text analysis techniques. The researchers obtain an accuracy of 77.62% in the prediction. *Serafini et al.* [27] study the influence of network sentiments by analysing Tweets related to Bitcoin and financial features along with statistical and deep-learning methods such as Auto-Regressive Integrated Moving Average with eXogenous input (ARIMAX) and LSTM on Bitcoin price prediction. The researchers conclude that asset sentiment plays a significant factor in determining Bitcoin price, an observation that is consistent with our initial study. Also, another conclusion derived from the study is that ARIMAX achieves better prediction accuracy (very accurate prediction with MSE of 0.00030187 on test data) than LSTM. *Radityo et al.* [30] uses different Artificial Neural Networks models, such as Backpropagation Neural Network (BPNN), Genetic Algorithm Neural Network (GANN), Genetic Algorithm Backpropagation Neural Network (GABPNN), and Neuro Evolution of Augmenting Topologies (NEAT) in order to make predictions on Bitcoin price. The researchers conclude that BPNN produces the best accuracy with MAPE of 1.998 ± 0.038%. The researchers use some popular market trend indicators such as William %R that serves as a momentum indicator and Exponential Moving Averages (EMA). Different periods of these indicators have been taken into consideration for study. Feature selection has been done using Greedy Forward Selection approach. The features that the researchers study are open, high, low, close, volume, EMA 12, EMA 26, %R 5, %R 14.These are some studies that perform text analysis in order to incorporate sentiment as a feature in the models. The conclusions drawn from these studies are pretty much in line with our expectation that Bitcoin is a sentiment driven financial asset.

Apart from these works, some more researches have been studied and referred to. These researches study the effect of other social and economic factors on Bitcoin price. *Aggarwal et al.* [28] analyses the effect of gold price on Bitcoin using models such as Convolution Neural Networks (CNN), LSTM and Gated Recurrent Unit (GRU). After bringing in tweet sentiment as a feature, the researchers conclude that LSTM performs much better than before with RMSE increasing from 151.67 to 32.98. The researchers also raise doubt on the correlation between gold price and Bitcoin price. *Luo et al.* [29] study the effect of Covid-19 data (confirmed case, recovery and death) on Bitcoin price. Luo combines this with sentiment analysis using Twitter and concludes thatCovid-19 data does not have any impact on Bitcoin price. Another important conclusion of this study is that investors refer to information from last 5 days before investing in Bitcoin. The researcher uses Random Forest, Decision tree, AdaBoost and SVM to make predictions and find that SVM is not very suitable to make Bitcoin predictions.

2.4 Parametric, Non-parametric and Semi-parametric Regression Models

The different types of predictive regression models with respect to their model function are parametric, nonparametric and semi-parametric. The model that is to be implemented depends on what information we have about the function [33].

The type of model that is most frequently used is a parametric regression model. These models can be both linear and non-linear and are comparatively easy to work with since there is a sufficient amount of prior information regarding the function that will be used. Hence, a fair amount of precise estimation and correct interpretation can be done regarding parametric regression models. In these models, value estimation of a certain number of model coefficients is done. Accuracy of the prediction depends on whether the correct model form has been used and how well the model parameters have been tuned. These models also train pretty fast. So, if we have the data for the independent variable $X = (x_1, x_2, x_3, \dots, x_k)$ where $x_1, x_2, x_3, \dots, x_k$ forms the feature vector, and for the dependent variable Y, the model function required to capture the relationship between X and Y can be written as: $Y = f(X, \beta) + \epsilon$, where X is the feature vector and the β forms the coefficient vector and ϵ is the distribution of the error. $f(X, \beta)$ should be correctly known in order to make correct predictions. Some examples of common parametric regression models are linear regression, logistic regression. Parametric models can be 2 types: 1. Linear models and 2. Non-linear models. Linear models are those in which the final model is just a linear combination of the predictors x_1, x_2 and so on. For example:

$$Y = \beta_0 + \beta_{10}x_1 + \beta_{11}x_1^2 + \beta_{20}x_2 + \beta_{21}x_2^2$$

In this case, estimation methods such as least square method or maximum likelihood method can be used. Also, it is interesting to note that in parametric regression models, unlike in the case of non-parametric models, we do not come across the curse of dimensionality [34]. Curse of Dimensionality basically means that the accuracy of a model does not increase any further or even decreases after a threshold number of features has been fed into the model. Non-linear models are those in which the final model is a non-linear combination of the predictors. For example:

1. $Y = \beta_0 + \beta_1 x_1 + \beta_2 e^{(\beta_3 x_2)} + \epsilon$
2. $Y = \frac{e^{\beta_0 + \beta_1 x_1}}{1 + e^{\beta_0 + \beta_1 x_1}} + \epsilon$

In this case, non-linear estimation methods such as Newton's method, Gauss-Newton algorithm can be used [33].

On the other hand, non-parametric models, which is our focus of study, do not require the assumption of any linearity and is used when no part of the model function can be known in prior. The function in this case can only be extracted from the data set that is fed into the model. Nonparametric models do not have a fixed number of parameters. Since the model learns the model form directly from the training set, the number of parameters can be infinite. This means that the relationship between the predictors and the dependent variable is unknown. The model function can take any form, which can be linear or non-linear. However, the model user will have no clue about it. This means that in case of nonparametric models, $f_X(x)$ is an arbitrary function and is unknown. As a result, the function does not have a finite number of parameters to capture the characteristics of the data. This is just the opposite of parametric functions in which the structure is known. Some models that learn directly from the training data are KNN, Kernel Ridge regression, decision trees, smoothing splines and many more. A researcher should be able to understand when to use non-parametric models [33]. If the data complexity is really high and introducing more features would only increase model complexity undesirably, introducing the curse of dimensionality and the artificial correlation between the inputs and outputs, resulting in lower accuracy, in that case the researchers should opt for non-parametric models [35]. It is to be noted that non-parametric models do not require building statistical model of the underlying data. In this type of models, the training set is not generalised, unlike in the case of parametric models. Rather, a comparison is carried out between the input test samples and the training set that result in predictions [41].

Semi-parametric regression models are those models in which only a part of the model function is known. It is basically a hybrid of parametric and non-parametric regression models. Semi-parametric models are used when we are looking to estimate only some of the parameters. These models can take many different structures. In a particular case of semi-parametric models, some of the predictors may not take any pre-determined form. On the other hand, the rest of the part takes known forms with the dependent variable. In a model that is given by: $Y = \beta_0 + \beta_1 x_1 + f(x_2) + \epsilon$, the relationship between predictor x_1 and Y is linear. On the other hand, the relationship between x_2 and Y is governed by 'f', which is unknown. In another case called Single Index Model (SIM),the form of the model is given by $Y = f(X\beta) + \epsilon$, where f is an unknown function, $X = (x_1, x_2, x_3, \dots, x_n)$ is a matrix of dimension n X k of predictors, β is k X 1 matrix of parameter values. $X\beta$ is called a single index and is a scalar. Error ϵ is supposed to be normally distributed around 0 with constant variance σ^2 [37]. Even though f in this case is unknown to the researcher, it is still semi parametric because the fact that f, which is known, is specified as a linear combination of X and β[33]. Semi-parametric models adapt better to different data sets and have better interpretive capability [36]. Profile kernel and spline methods come under this category [38].

<u>Chapter 3</u>

Architecture of Models Used

In this chapter, we will look into the architecture of the different models used. These are the models that will be used to predict Bitcoin prices in the next chapter and hence, it becomes important to know and understand how these models work. The models such as KNN, Kernel Ridge, Random Forest Regressor and Gaussian Process Regression are all non-parametric. The only semi-parametric model that we will be analysing is Cubic Smoothing Spline.

3.1 K-Nearest Neighbour

KNN, which stands for K-Nearest Neighbours, is a non- parametric regression model that learns the data directly from the training set. The distance between each training data point $x_i$is calculated and then the nearest training points are taken into consideration. The number of data points that will be involved to make a prediction for a particular point is determined by the parameter 'K'. Once the K nearest data point have been figured out, the average of the predictions $y_i$for the K Nearest training data is calculated and is used as prediction. The distance between point x and $x^{(i)}$in KNN is usually calculated using the Euclidean distance formula, which is given by$d\left(x^{(i)}, x\right) = \sqrt{\sum_{j=1}^{n}\left(x_j^{(i)} - x_j\right)^2}$, where 'j' keeps a count of the next reference point with which the distance will be calculated. K number of nearest neighbours is then selected based on this distance. These K points are then given weights with respect to their distance from the referenced data point for which the prediction is being calculated. The kind of weights that are applied on these points are 1. Uniform weights, where all the nearest neighbours are given a weight of 1, and 2. Gaussian weights, where the weights of the nearest neighbours are decided using the formula$w^{(i)} = e^{-\gamma d(x^i,\ x)^2}$. As a result of the Gaussian weight, the weight of the neighbours decreases as the distance from the referenced data point increases. The weighted mean that becomes the prediction for the referenced data point is given by:$\hat{y} = \frac{\sum_{i \in N_k} w^{(i)} y^{(i)}}{\sum_{i \in N_k} w^{(i)}}$, where N_k defines the number of points in the neighbourhood of x_i, y_i is the value of associated with each neighbouring point and w_i is the weight associated with each neighbouring point.In case of uniform weights, the weight assigned is 1 and hence the prediction is derived from $\hat{y} = \left(\frac{1}{N_k}\right) \sum_{i \in N_k} y_i$. The rate at which this increase or decrease in weight happens is determined by the factor γ. Higher the value of γ, faster is the decrease in the weight. Also, the value of K, which represents the number of nearest neighbours considered while making the prediction, helps to control overfitting and underfitting. As the value of K increases, the number of nearest neighbours considered also increases. The model is then able to capture a more generalised behaviour of the data and thus, the model starts to become less sensitive to the training data. A combination of $\gamma$and K can be used to control overfitting and underfitting. Figure 4 shows how γ can be used to control overfitting and underfitting with K equalling the data set size. A general feature of the KNN model is that it often fails to capture the data behaviour towards the end of the datasets and makes really inaccurate predictions in areas where the data is not present since the model trains directly from the dataset.

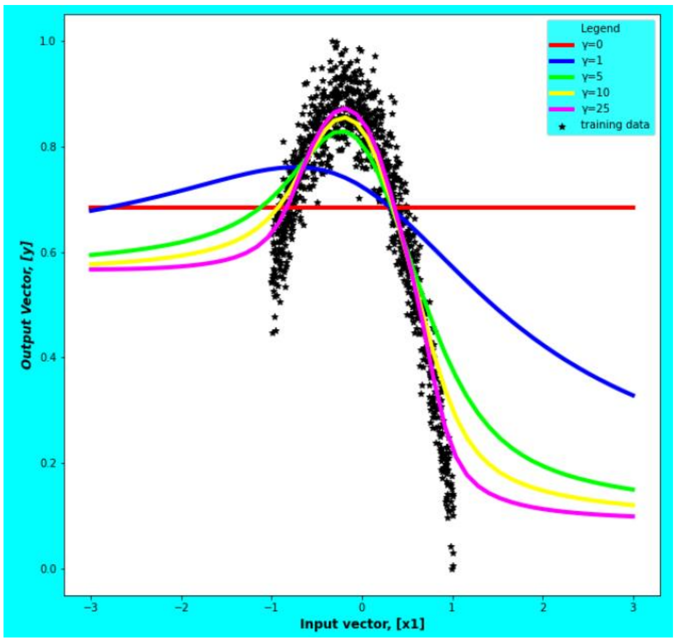

Figure 4 showing how γ can be used to control overfitting and underfitting

The performance of the KNN model is dependent on the value of K and γ selected. The selection of the best value needs to be done using cross validation. Model complexity of KNN models is given by n/k. However, [39] propose a better estimate for KNN model complexity, given by $d = \frac{n}{k.n^{\frac{1}{5}}}$, where d is the degree of freedom, which is statistical term for model complexity. It is to be noted that KNN can also be used for prediction in classification problems. In this case, the referenced point will be classified with respect to the majority classification vote of its neighbours. This is given by $y_i = \arg\max_k \sum_{x_j} y(x_j, c_k)$, where x_j is one of the neighbouring points, $y(x_j, c_k)$ indicates whether x_j belongs to class c_k. KNN is fast and requires less space when training for classification problems. However, it fails to yield good results if the data set is unevenly distributed among the classes [40].

3.2 Random Forest

Before diving into the Random Forest Regressor model, it is very important to look into the building blocks of Random Forest, which are Decision Trees and Regression trees. Though, Regression Tree is our point of interest since we are dealing with Random Forest Regressors, a look at Decision Trees, which are used in classification problems, is important before delving into Regression Trees and eventually Random Forest Regressors. A decision tree, as the name suggests, is a tree that branches (or classifies) based on a decision. The classification decision at a particular level is based on the answer to the question that the model receives at that level. An example of a decision tree that solves a classification problem has been show in Figure 5.

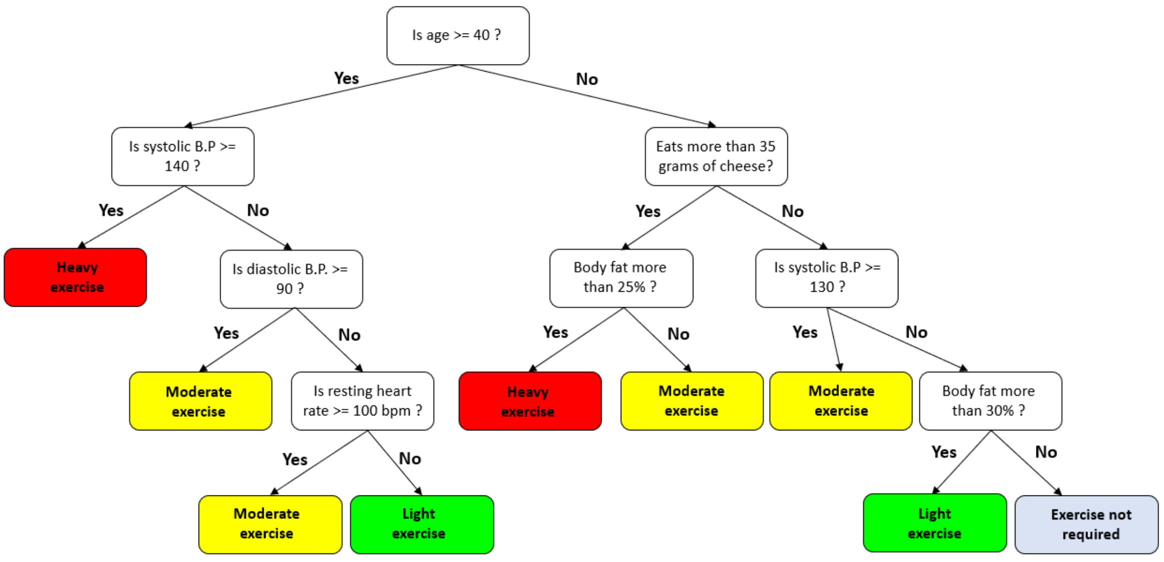

Figure 5 showing a decision tree

Figure 5 shows a decision tree that can be used to predict the level of exercise that a person needs to do every day. It can be seen that the decision is arrived using different questions at different levels. The top of the tree is called the root node and the nodes in which the branching ends are called leaf nodes. The leaf nodes are the ones in which the final classification is made. The sample data from which the decision tree can be built has been shown in table 1.

Age >= 40?	S.B.P. >=130?	S.B.P. >=140?	Cheese Consumption > 35 g?	D.B.P. >=90?	R.H.R.>=100 bpm?	Body fat %age?	Exercise level
No	No	No	No	No	No	25	No
Yes	Yes	No	No	Yes	Yes	30	Moderate
Yes	Yes	Yes	Yes	Yes	No	40	Heavy
No	No	No	No	No	No	35	Light
No	Yes	No	No	No	No	35	Heavy
Yes	Yes	Yes	Yes	Yes	No	20	Heavy

Table 1: Sample data to build the decision (classification) tree in Figure 5

Consider that there are 300 such observations. The first thing that we need to figure out is which of the variables should be the root node. For this, we need to start classifying the exercise levels with respect to the first variable in the table that is age of the person. This is done for all the persons in the study. This process is repeated for the other variables in the study such as Systolic Blood Pressure, Cheese Consumption, Diastolic Blood Pressure, Resting Heart Rate and Body Fat percentage. From the sample data, it can be seen that none of the variables make the classification perfectly, meaning that a particular category of exercise cannot be recommended with absolute certainty if the age of person in greater than greater than 40. Similarly, it also cannot be said that if the systolic blood pressure of a person is greater 140, he/she definitely needs to go for heavy exercise. Since, none of the variables manage to make classifications with absolute certainty on its own, these variables are considered impure. Hence, it cannot be directly determined which variable would be the best choice for root node. Thus, in order to determine which variables makes the best classification, there are different techniques such as Gini Index, Entropy, Chi-Square and so on. Using any of these techniques, the first split and the subsequent splits can be determined, thus building the decision tree. Note that the variable Body Fat %age is numeric whereas the other variables in table 1 are categorical. Hence, we need to find cut offs for the Body Fat %age variable that will help us classify the exercise levels with respect to this particular variable. This can also be done using the techniques mentioned above. Once the decision tree has been prepared, making prediction becomes easy for a new sample that comes in with some combination of the variables.

Regression Trees on the other hand are used when the dependent variable does not have a linear relationship with the predictors. This means that a straight-line model will not work in this case and we need to introduce polynomial features into the model. In such a scenario, Regression Trees can also be used. Regression Trees are a type of Decision Trees. In a Regression Tree, each leaf node represents a numeric value, unlike Decision Trees in which the leaf nodes represent categorical values. Figure 6 shows increase in life expectancy when a 100ml of different concentrations of a certain drug has been administered in humans. It can definitely be said that there is no point in building a Regression Tree for only one predictor since prediction can be made simply by visualisation. However, Regression Trees become very useful when more predictors come into the model and prediction using visual aid become almost impossible.

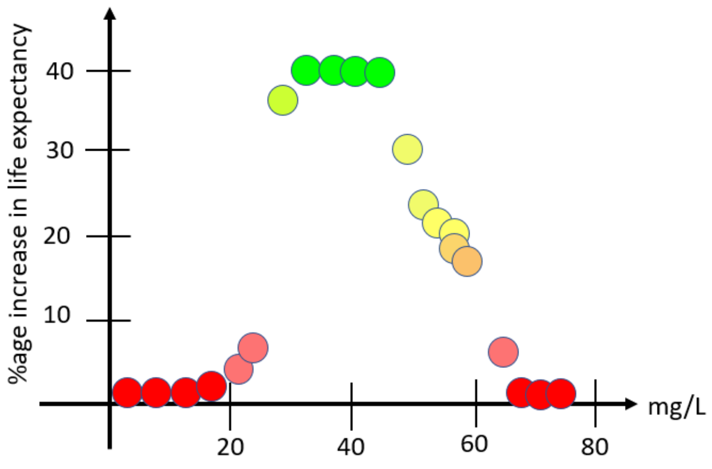

Figure 6 showing increase in life expectancy with variation in drug concentration

The average increase in the life expectancy from the 2 smallest dosages (dosage < 10 mg/L) is about 2% and the average increase in the life expectancy of the points on the greater side of 10mg/L is 18%. Therefore, a simple tree that splits the observations based on "dosage < 10 mg/L?" will have leaf nodes of 2% (left leaf) and 18% (right leaf). The values in each of these leaves will be the prediction that this tree will make in case of a dosage < 10 mg/L split. Even though the deviation in the prediction from the actual value of the points less than 10 mg/L is almost NILL, the deviation in the prediction of the points for dosage > 10 mg/L is quite high. For example, deviation for dosage = 40 mg/L is about 40%-18% = 22%.It is noticed that the sum of squared residuals of the deviation of the prediction from the actual value of all the points on either side of dosage = 10 mg/L is pretty high value if the split is made at this point. Hence, this dosage = 10mg/L is not an ideal split. The point along the x-axis at which the sum of squared residuals is the lowest will be the first split point and thus, the root node. Say that the root node is made at dosage = 25 mg/L. Now, if we look at the points to the left of dosage = 25 mg/L, we can further split these points using the same technique that we used above. We can make such splits down to the last point until no change in the prediction value can be observed on the left nodes (corresponding to answer 'Yes'). However, as the number of splits increases, the model becomes more and more sensitive to the data and this would definitely lead to overfitting. The parameter provided by Scikit-Learn to control the number of splits and hence the depth of the tree is max_depth. This parameter determines the depth to which a tree will be built.There are many other techniques that can be employed to prevent overfitting. Another common technique is to make a split only when there are a minimum number of sample available. This is controlled by a parameter called min_samples_split provided by the Scikit-Learn library. In this case, let this minimum number be 6. Figure 7 shows the Regression Tree built for the data set. For points less than 25 mg/L, the average increase would be 5% (say). Hence, this will be the prediction of the model for points less than 25 mg/L, which is not too bad. Also, since there are only 5 points to the left of 25mg/L, we will not make any further split. So, we need to find the ideal split for points to the right of 25mg/L since we have more than 6 observations on this side. This is done in the similar way the root node was found and the initial split was made. The smallest sum of squared residual is found for the point to the right of 25mg/L and the next split is made at 59 mg/L (say) with the average value of the points to the right of 45 mg/L equalling to 2.5%. Again, since the number of sample points to the right of 59mg/L is less than 6, no further splits are made in this region. The next split is made in the region between 25mg/L and 59 mg/L. This process continues until the number of sample points left in a particular region does not become less than 6.

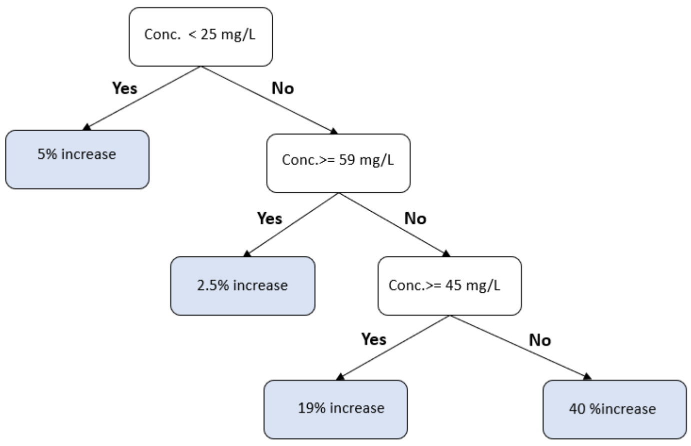

Figure 7 showing a sample regression tree with only one variable

This becomes a little trickier when more than one predictor comes into the picture. Table 2 shows a sample of the data when other predictors come into the picture. Just like earlier, we will try different values of concentration and check at which point do we find the lowest sum of squared residual. The point (say concentration = 25 mg/L) at which we find the lowest sum of squared residuals becomes a candidate for the root. This process is repeated for age and the point (say at age = 50), at which the lowest sum of squared residuals is found, becomes another candidate for the root node. This process is repeated for the variable sex. However, for this variable, there can only be one split and that is between males and females. We use this split to calculate the sum of squared residuals, which becomes the 3rd candidate. And the average of the percentage increase in case of males and the average of the percentage increase in case of females in considered as the predictions. The candidate for which the sum of squared residual is the minimum is selected to be the root node. In this case say the second candidate that is age = 50 has the lowest sum of squared residuals. Hence, this is considered the root node. The tree is grown with the next branching considered with the second candidate (second smallest sum of squared residual) that is concentration = 25 mg/L (say) and repeated for the last candidate. The sample Regression Tree built in this case is shown in figure 8.

Concentration	Age	Sex	%age increase
2	30	M	10%
5	10	F	25%
15	25	F	15%
18	45	M	34%
22	60	M	40%
25	75	F	37%

Table 2 sample data to build the Regression Tree with multiple predictors

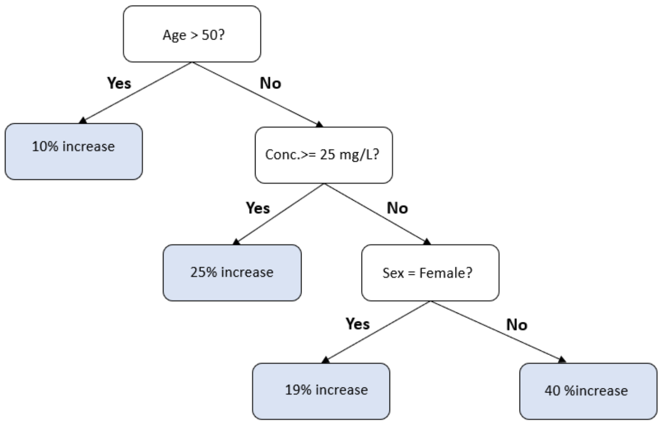

Figure 8 showing the sample Regression Tree formed using the variables in table 2

Hence, we see that the basic principle to construct a Regression Tree is to divide the original space into 2 separate regions and construct a binary tree. Hence, if we consider a dataset like: $\{(X_1, y_1), (X_2, y_2), \ldots, (X_i, y_i), \ldots, (X_n, y_n)\}$, where X_i is ap-dimensional vector with 'p' number of features. The steps to build a Regression Tree would be as follows [43]:

1 Solve the objective function, which is given as follows:
$\min\limits_{j,s}[\min\limits_{c_1} \sum_{x_i \in R_1(j,s)} (y_i - c_1)^2 + \min\limits_{c_2} \sum_{x_i \in R_2(j,s)} (y_i - c_2)^2]$, where c_1 and c_2 are the predictions of each region, R_1 and R_2 are the 2 separate regions created after a node splits a region,j represents the best segmentation variable and s represents the best segmentation point from the variable j, meaning the variable that is used to split the region and the point at which the region is split respectively. The 2 regions R_1 and R_2created using the selected segmentation variable 'j' and the segmentation point 's' of the variable j is given as follows:
i. $R_1(j, s) = (X|X^j \leq s)$
ii. $R_2(j, s) = (X|X^j > s)$, where X^j is the j^{th} feature

2 The prediction $c_m$for regions noted in i and ii aboveis as follows:
$c_m = \left(\frac{1}{n}\right) \sum_{x_i \in R_m(j,s)} y_i,\ x \in R_m, m = 1, 2$ since the space is divided into 2 sub-regions.'n' is the total number of input data points in each region. This basically means that the prediction in each region will simply be the average of the actual values in each region.

3 Steps 1 and 2 are repeated until the stop condition is met. The stop condition could be defined by the number of samples required to make a split.

4 The final step is to segregate the input space into 'p' regions in order to build the Regression Tree, where p is the total number of input features.

Random Forests are built using Decision trees (or using Regression Trees in case of Random Forest Regressor model). Decision Trees are not very accurate with their predictions. Hence, a collection of Decision Trees in the form of a Random Forest is used to increase the prediction accuracy by a huge margin. Random Forest also increases the flexibility of Decision Trees. Creating a Random Forest requires the following steps:

1 The first step in building a Random Forest is to create a **Bootstrap dataset**. In order to create a bootstrapped dataset, samples are randomly selected from the original dataset. Also, the same sample can be selected more than once from the original dataset. It is to be noted that the size of the Bootstrap dataset has to be the same as that of the original dataset.

2 The next step is to create a Decision Tree using the bootstrapped dataset. This is done by first randomly selecting only a subset of the variables from original dataset. For example, say that out of a total of 5 variables (A, B, C, D, E) in the original dataset, any 2 variables (say) have

been picked randomly. Of these 2 variables selected (say B and E), the variable that does the best job in separating the samples needs to be figured out using techniques such as Gini Index, Chi Square, Entropy and many more. The variable that has the least impurity becomes the root node say variable B. Out of the remaining 4 variables (A, C, D, E), 2 variables (say A and D)are again randomly selected and the variable with the least impurity makes the next node (say D).This is how the tree is built as usual by randomly selecting a fixed number of variables from the remaining variables at each step. Regression Trees are used in case of Random Forest Regressors. The only difference is that the sum of squared residuals technique is used to select the nodes as discussed earlier instead of the techniques such as Gini Index as used to build Decision Trees. The best number of random variables is selected using cross validation.

3 Step 1 and Step 2 are repeated many hundreds of times and many different Decision Trees (or Regression Trees) are built and hence, a Random Forest (or Regressor) is built.

Once the Random Forest is built, the new data sample is passed through it and the predictions of all the individual trees are aggregated. In case of classification, the value to which the test data is classified the maximum number of times is taken as the final prediction for the sample data. Similarly, in case of regression, the average of the predictions of all Regression Trees is taken as the final prediction for the sample data. This process of bootstrapping the data and then using the aggregate to take the final decision is called **Bagging**. Random Forests make it possible to capture the non-linear interactions between the dependent and independent variables [42].It is generally observed that about 33% of the samples from the original data set are left out during bootstrapping. This data is called **Out of bag** data and can be used as test data to measure the prediction accuracy of the model. The flow diagram that represents the working of a Random Forest Regressor model is shown in Figure 9 [42]:

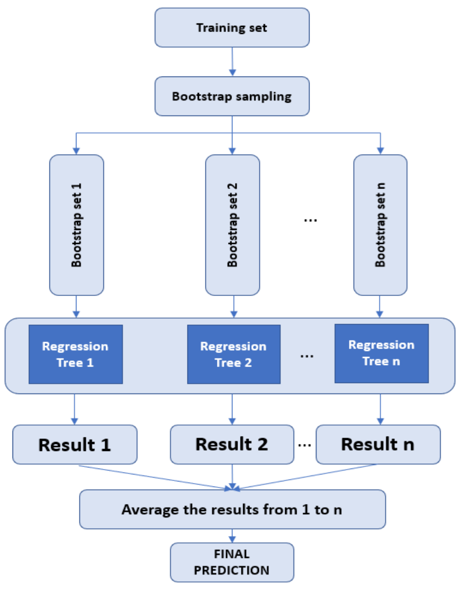

Figure 9: A schematic view of the Random Forest Regressor model

3.3 Kernel Ridge Regression

Kernel Ridge Regression is a kind of non-parametric regression model, which merges ideas from KNN and linear regression with L2 regularization. The basic idea behind this model is to associate a feature with every training data point and then use a linear model. This basically means that there will be a lot of features in this model and thus, the feature vector will be large. This is done by introducing a kernel function. Consider a generalised form of the training data, which is given by: $(x^{(i)}, y^{(i)})$, where $x^{(i)}$ is the input feature of the i^{th} data point and $y^{(i)}$ is the corresponding output of the i^{th} training data point. Now, since every training data point is associated with a feature, the i^{th} feature will be derived from the training data point $x^{(i)}$ using the function $y^{(i)}K(x^{(i)}, x)$, where x is the point for which the prediction is being calculated. The $K(x^{(i)}, x)$ function, which is called a Kernel, gives a sense of the distance between $x^{(i)}\ and\ x$ in a way that K is more of a weighting function of the distance between $x^{(i)}$ and x. If the distance between $x^{(i)}$ and x is very less, then K will be close to 1 and when $x^{(i)}$ and x are far apart, K will be close to 0. The kernelized regression model that comes up from here is:

$$\hat{y} = \theta_0 + \theta_1 y^{(1)} K\left(x^{(1)}, x\right) + \theta_2 y^{(2)} K\left(x^{(2)}, x\right) + \cdots + \theta_m y^{(m)} K\left(x^{(m)}, x\right)$$

So, we can see that the Kernel Ridge Regression model is more or less a kernelized version of linear regression. It is to be noted that kernels can also be used for classification and the model that is used in that case is:

$$\hat{y} = sign(\theta_0 + \theta_1 y^{(1)} K\left(x^{(1)}, x\right) + \theta_2 y^{(2)} K\left(x^{(2)}, x\right) + \cdots + \theta_m y^{(m)} K\left(x^{(m)}, x\right))$$

The model parameters$\theta_0, \theta_1, \ldots, \theta_m$need to be estimated and can be learnt by minimising the cost function just like in the case of linear models with non-linear features, where instead of using polynomials, we can use the Kernel weights. Since, each training data point gets associated with a feature, the vector of parameters is huge in this case. As a result, Kernel Ridge Regression ends up occupying a lot of space for its calculations. As the distance between the training data points and the reference point gets bigger, the farther points become insignificant and the model gets reduced to the "K nearest neighbours". Hence, a simpler way to calculate the prediction is to take the weighted average of the nearest neighbours which is given as: $\hat{y} = \frac{1}{N}\sum_{i=1}^{N} \omega_i y_i$, where N is the number of nearest neighbours, ω_i is the weight calculated for each of the neighbouring points and y_i is the actual value of the nearest neighbours. The weights are decided by the Nadaraya-Watson estimator, which is given as $w_i = \frac{K\left(\frac{x_i - x}{b}\right)}{\sum_{j=1}^{N} K\left(\frac{x_i - x}{b}\right)}$ [45], where K is a non-negative, real valued function such that K(x)=K(-x) that is it is symmetrical for all values of x and $\int K(x)dx = 1$.The denominator of the estimator always results in 1. The selection of kernels must satisfy the conditions given by: 1. $K(x, x_i) > 0$, 2. $\int xK(0, x)dx = 0$ and 3. $0 < \int x^2 K(0, x)dx < \infty$ [46]. Some commonly used kernel functions are uniform kernel, Gaussian kernel and Epanechnikov kernel. The numerator put higher weights on the points that are closer to the reference point x. 'b' is the bandwidth or the "smoothing parameter" that controls how quickly the weight falls as the training data points move away from x. The smoothing parameter value is chosen using the Jackknife cross validation given by: $CV(b) = \left(\frac{1}{N}\right) \sum_{i=1}^{N} [y_i - \widehat{y_{[i]}}]^2$, where$\widehat{y_{[i]}}$ is the prediction for x when one observation is left out. The value for 'b' needs to selected very carefully since the model is very sensitive to this parameter. The smoothness of the model increases as the value of 'b' increases. A sample model built using Kernel Ridge has been shown figure 10. The model is able to correctly capture the behaviour of the data and also, is able to make correct predictions where the data is not present.

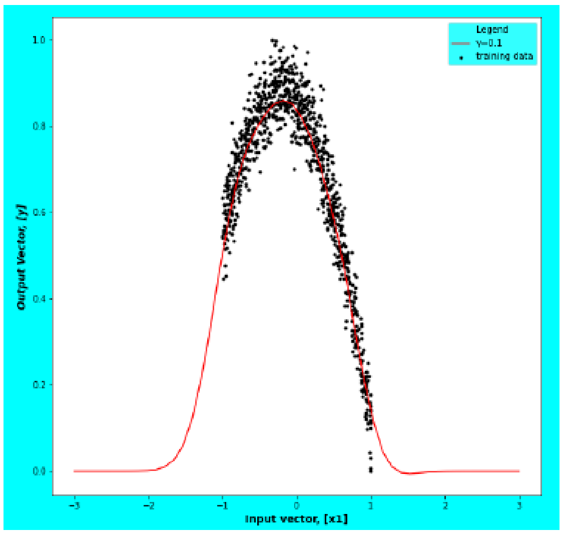

Figure 10 showing a sample Kernel Ridge Regression model.

3.4 Natural Cubic Splines

Smoothing splines is a semi-parametric algorithm that successfully manages to fit non-linear data. The idea behind smoothing spline is to divide the data set into different segments and then capture the behaviour of every segment separately. This regression approach is different from the rest of the models because it does not aim to capture the behaviour of the entire dataset in a single polynomial model, but combine separate models to form one final model. Figure 11 shows how segments selected can be used to fit separate models. The breakpoints have been selected to be at 2006-08, 2007-01 and 2007-08. These breakpoints are called knots. And between each knot, a linear model has been fit. This technique is referred to as piece-wise regression. It is to be noted that figure. 11 does not show the actual model, but is only a descriptive representation.

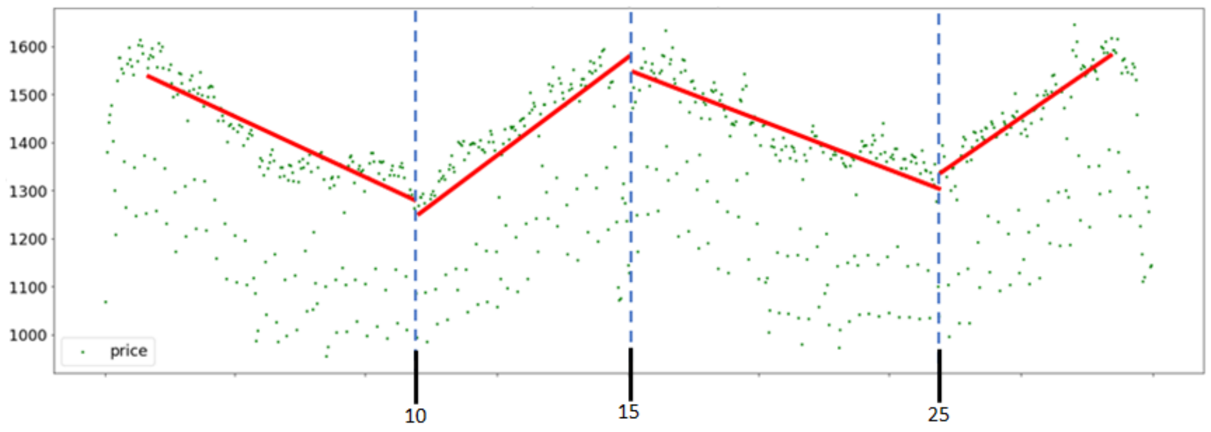

Figure 11 showing a descriptive representation of piece-wise regression

However, it can be clearly seen from fig. 10 that these linear models are breaking in at the knots and are discontinuous. So, for $x \leq 10$, the model is given by $y = \beta_0^{(1)} + \beta_1^{(1)}x + \varepsilon^{(1)}$. Similarly, for $10 \leq x \leq 15$, the model is given by $y = \beta_0^{(2)} + \beta_1^{(2)}x + \varepsilon^{(2)}$. For $15 \leq x \leq 25$, the model is given by $y = \beta_0^{(3)} + \beta_1^{(3)}x + \varepsilon^{(3)}$ and so on. In order to build a continuous model, we use truncated functions. Thus, the model is given by:

$$y = \beta_0 + \beta_1 x + \beta_2(x-10)_+ + \beta_3(x-15)_+ + \beta_4(x-25)_+ + \varepsilon,$$

where a truncated function is given as: $(x-k)_+ = \begin{cases} 0, if\ x < k \\ x-k, if\ x \geq k \end{cases}$. Thus, we can see that between 0 and 10, the model simplifies to $y = \beta_0 + \beta_1 x$. Between 10 and the 15, the model becomes $y = \beta_0 + \beta_1 x + \beta_2(x-10)_+$ and so on. Thus, we can clearly see that the number of parameters has decreased because earlier we had separate intercept terms for each interval, whereas now there is a single intercept term and the intercept of every segment is affected by equation of the line formed in the previous segment. Thus, we get a continuous model comprising of separate linear models called linear splines. A descriptive representation of linear splines has been shown in figure 12 [44].

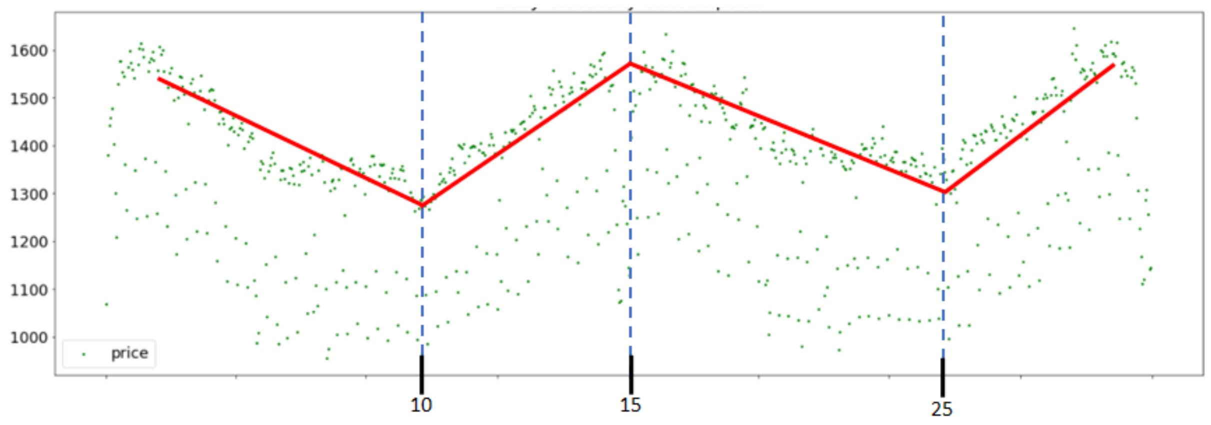

Figure 12 showing a descriptive representation of linear splines

Now, instead of having straight forward lines between the segments, we can have polynomial models that would represent the data behaviour better. Hence, we need to introduce polynomial features into the model up till order 3. This would be given as: $y = \beta_0 + \beta_1 x + \beta_2 x^2 + \beta_3 x^3 + \beta_4 (x-10)_+ + \beta_5 (x-10)^2{}_+ + \beta_6 (x-10)^3{}_+ + \beta_7 (x-15)_+ + \beta_8 (x-15)^2{}_+ + \beta_9 (x-15)^3{}_+ + \beta_{10} (x-25)_+ + \beta_{11} (x-25)^2{}_+ + \beta_{12} (x-25)^3{}_+ + \varepsilon$. When polynomial features are introduced, the model looks like as shown in figure 13 [44].

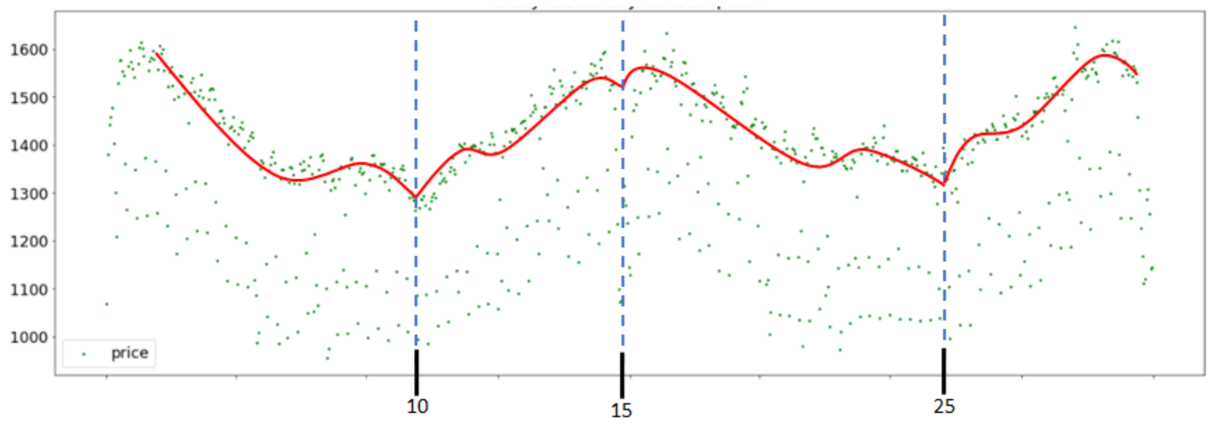

Figure 13 showing descriptive representation of polynomial features introduced with linear splines

However, it is can be noticed clearly that at the knots, the function is non-differentiable. Hence, we can better this model by removing these sudden changes in the model. The idea to achieve this comes from the fact that the first and second derivative of a function needs to exist in order for the function to be differentiable. Also, the first and the second derivative of a function play a significant role in determining the smoothness of the function. The first derivative is given by: $y' = \beta_1 + 2\beta_2 x + 3\beta_3 x^2 + \beta_4(x > 10)_+ + 2\beta_5(x - 10)_+ + 3\beta_6(x - 10)^2{}_+ + \beta_7(x > 15)_+ + 2\beta_8(x - 15)_+ + 3\beta_9(x - 15)^2{}_+ + \beta_{10}(x > 25)_+ + 2\beta_{11}(x - 25)_+ + 3\beta_{12}(x - 25)^2{}_+ + \varepsilon$ and the second derivative is given by: $y'' = 2\beta_2 + 6\beta_3 x + 2\beta_5(x > 10)_+ + 6\beta_6(x - 10)_+ + 2\beta_8(x > 15)_+ + 6\beta_9(x - 15)_+ + 2\beta_{11}(x > 25)_+ + 6\beta_{12}(x - 25)_+ + \varepsilon$. Hence, it can be seen that the first and the second derivative are discontinuous at the knots that are given by the associated intercept terms. The trick here is to simply drop the truncated function terms with order less than 3. This is because if these terms are dropped, the first and the second derivative of the functions will become continuous [44]. When this is done, the final model becomes:

$$y = \beta_0 + \beta_1 x + \beta_2 x^2 + \beta_3 x^3 + \beta_4(x - 10)^3{}_+ + \beta_5(x - 15)^3{}_+ + \beta_6(x - 25)^3{}_+ + \varepsilon.$$

The final model is shown in figure 14. It is to be noted that regression splines can use any degree of polynomials, but cubic polynomials are the most effective often since these have the best capability to capture most of the data behaviour. The models in which the highest order is 3 are called cubic splines [44].

As the number of knots increases, the flexibility of the model increases and hence, the residual sum of squares decreases. However, as the number of knots increases, the sensitivity of the model increases and this could lead to overfitting. Model accuracy will also depend on the location of placing the knots. The idea behind the placement of these knots is that the number of knots in the model will more or less be uncontrolled. However, a penalty will be introduced in the residual sum of squares to control the roughness of the model, which is introduced by the more number of knots. This penalty will be introduced in the form of a second derivative of the function because second derivative essentially gives a sense of the rate at which the slope of the curve is changing, which in turn is a measure of how rough the fitted curve is. Hence, the residual sum of squares, including the penalty parameter called roughness penalty, is given by:

$$\sum_{i=1}^{n} \left(y_i - f(x_i)\right)^2 + \lambda \int [f''(x)]^2$$

Interestingly, $\hat{f}(x)$, which is the optimised solution of the above problem is nothing but a cubic spline with knots at every training data point x_i. However, it is not practical to put a knot in the place of every training data point. Instead, a large number of knots is used and the roughness penalty parameter is chosen using cross validation. Also, the knots are placed, by default, such that the number of data points between each knot is the same [44].

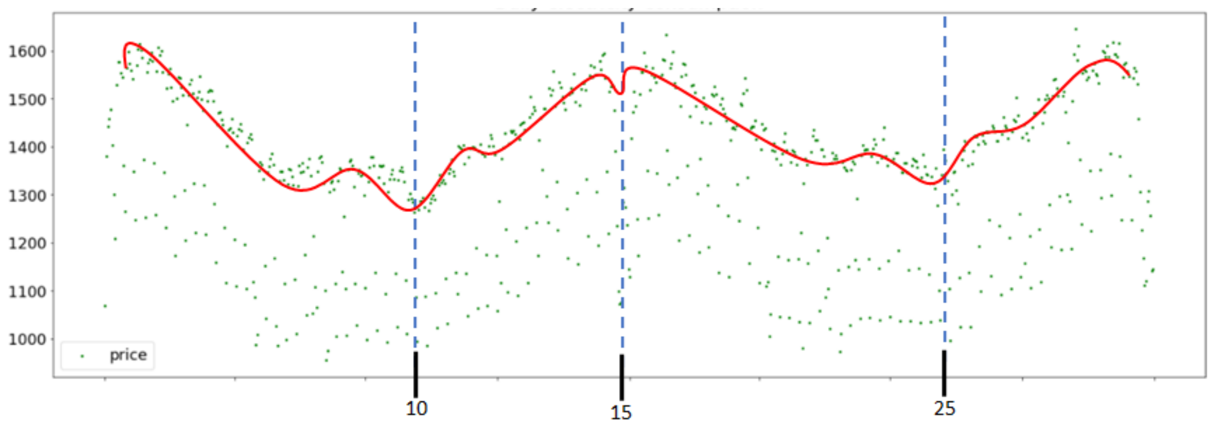

Figure 14 showing a descriptive representation of smoothing splines

3.5 Gaussian Process Regressor

Gaussian Process Regressor is a probabilistic, non-parametric regression model that works on the principles of Bayesian Inference, including the concepts of prior and posterior. In this case, prior knowledge or kernels are extensively used to make predictions. The probability distribution function in form of Gaussian distribution of a random variable X that assumes values $x_1, x_2, \ldots, x_n$ is represented as $P_X(x) \sim N(\mu, \sigma^2)$ and is given by:

$$P_X(x) = \left(\frac{1}{\sqrt{2\pi\sigma}}\right) \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)$$

When more than one correlated variables $(X_1, X_2, \ldots, X_D)$ comes into the system, we use a multi-variate normal (MVN) as the probability distribution, where D represents the total number of variables in the system or the dimension of the system. The probability distribution in this case is given by:

$$P(x|\mu,\Sigma) = \frac{1}{(2\Pi)^{\frac{D}{2}}|\Sigma|^{\frac{1}{2}}} \exp\left[-\left(\frac{1}{2}\right)(x-\mu)^T\Sigma^{-1}(x-\mu)\right]$$

Where x represents the vector of the random variables $[X_1, X_2, \dots, X_D]$, μ represents the vector of means of the individual variables $[\mathrm{E}(X_1), E(X_2), \dots, E(X_D)]$, Σ represents the covariance matrix, which measures how the values of the random variables $X_1, X_2, \dots, X_D$ change with respect to each other. $|\Sigma|$ represents determinant value of the covariance matrix and Σ^{-1}, which itself is a matrix represents the inverse of the covariance matrix. It is to be noted that a random vector x which represents a collection of multiple correlated variables is multivariate normal if any linear combination of the variables, given by $a_1X_1 + a_2X_2 + \cdots + a_DX_D$, has a univariate normal distribution for any value of $a_1, a_2, \dots, a_D$ [48]. Figure 15 shows a representation of the normal distribution of 2 variables (for easier visual representation) both when the variables correlated and when the variables are independent. It can be clearly seen that because the 2 variables X and Y are correlated, the probability distribution of the points of X and Y makes an elliptical shape. On the other hand, when the variables X and Y are non-correlated, the probability distribution of the points makes a circle. The probability distribution will look like a bell with highest probability towards the middle and lowest probability towards the lower end of the curve. However, for simplicity, the bell has been replaced with a colour index. A MVN is expressed as:

$$\begin{bmatrix} x_1 \\ x_2 \\ \dots \\ x_D \end{bmatrix} \sim N\left(\begin{bmatrix} \mu_1 \\ \mu_2 \\ \dots \\ \mu_D \end{bmatrix}, \begin{bmatrix} \sigma_{11} & \sigma_{12} & \dots & \sigma_{1D} \\ \sigma_{21} & \sigma_{22} & \dots & \sigma_{2D} \\ \dots & \dots & \dots & \dots \\ \sigma_{D1} & \sigma_{D2} & \dots & \sigma_{DD} \end{bmatrix}\right) \sim N(\mu, \Sigma)$$

Where, x_1, x_2, …, x_D represent values that the random variables $X_1, X_2, \dots, X_D$ take. Similarly, μ_1, μ_2, …, μ_D are the mean values of the random variables $X_1, X_2, \dots, X_D$ and Σ is the co-variance matrix that represents the correlation between every pair of the random variables [48]. This covariance matrix will later be replaced by the kernel matrix in Gaussian Process Regression.

For a vector of input of random variables $[X_1, X_2, \dots, X_D]$, if the output vector f(x), given by $[f(x_1), f(x_2), f(x_3), \dots, f(x_D)]$, where x_1, x_2,…, x_D are the values taken by random variables $X_1, X_2, \dots, X_D$, is Gaussian distributed, then f would be referred to as a Gaussian process.

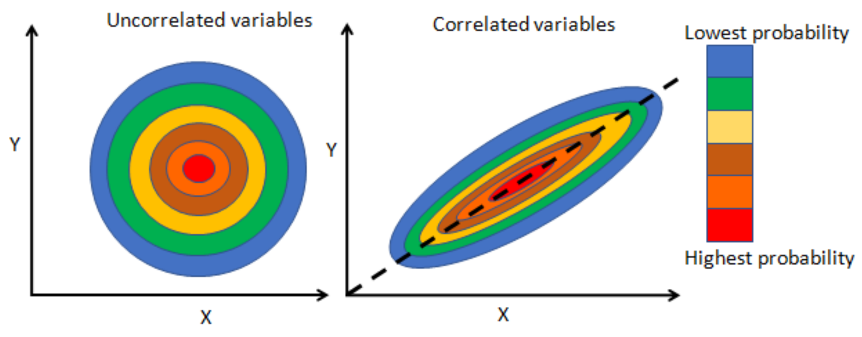

Figure 15 showing the normal distribution of 2 variables X and Y

The Gaussian process, given as $f\ (or\ y) \sim GP(\mu, k)$ just like in the case of MVN, is specified using the mean function $\mu(x)$, defined by the mean vector $[\mu(X_1), \mu(X_2), \mu(X_3), \ldots, \mu(X_D)]$ and D X D covariance or kernel matrix. A kernel is a function that describes the co-variance between each pair of random variables or the relationship between each pair of the random variables used in the Gaussian process as $k(x, x')$. These kernels help us to use our prior knowledge about the characteristic of data while building the model [47]. A kernel matrix looks like:

$$K = \begin{bmatrix} k(x_1, x_1) & k(x_1, x_2) & \ldots & k(x_1, x_D) \\ k(x_2, x_1) & k(x_2, x_2) & \ldots & k(x_2, x_D) \\ \ldots & \ldots & \ldots & \ldots \\ k(x_D, x_1) & k(x_D, x_2) & \ldots & k(x_D, x_D) \end{bmatrix}$$

where $k(x_1, x_2)$ is the co-variance between the values of random variables 1 and 2 and is measured using the kernel chosen and so on. Also, a kernel function must be symmetric and positive definite. Kernel functions are used to introduce some degree of smoothness in the models. Different kernels are used with respect to the designer's prior information about the data. If it is known that the data is not very smooth, in that case the squared exponential kernel (or the RBF kernel), given by $k(x, x') = \exp\left(-\frac{(x-x')^2}{2}\right)$ can be used since this kernel brings in a tremendous amount of smoothness in the model. The periodic kernel, given by $k(x, x') = \sigma^2 \exp\left(-\frac{2sin^2(\pi|x-x'|/p)}{l^2}\right)$, where p is the distance between the repetitions in the function and *l* determines how long the curves will be in the function, is used when there is some periodicity in the data.

One of the most important differences between the polynomial regression models and the Gaussian process is that the polynomial regression techniques output only a single model, whereas there could be multiple ways to connect the training data points and build the model. This is facilitated using the

Gaussian process as shown in figure 16. For simplicity of visualisation, figure 16 uses only one random variable to depict the probability distribution over functions. Gaussian process enables us to build distributions over models (or functions) and the mean of these functions is then selected as the ultimate model as shown by the thick black line in figure 16. The wider these distributions are, the more uncertainty is in the actual prediction. Gaussian process can also be used to model functions that have some amount of noise in them. That is, these functions do not necessarily pass through the given points but pass very close to them [47].

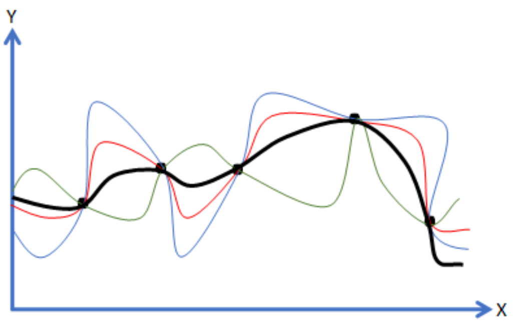

Figure 16 showing the mean function of the different functions generated by GPR

The task at hand now is to calculate the different distribution of the models. This is where Bayesian concepts of posterior and prior comes in. The posteriors, which are the different functions, are nothing but the prior that has been combined with the sample data points. Hence, it can be seen from figure 16 that all the functions pass through the same point where the observed data point is present. However, in regions where the observed data point is not present, the same functions vary a lot since the posterior has not been combined with any data point in that region. Also, when there are greater number of observed data, the functions get updated and the final prediction becomes accurate. Also, the variance in the probability distributions over the functions decreases as shown in figure 17.

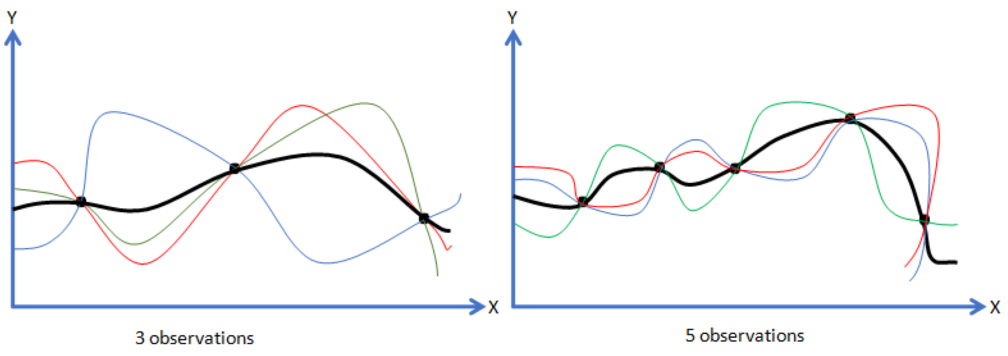

Figure 17 showing change in the function as more data points are added

The form of the multivariate regression model is given by: $P(f|x) = N(f|\mu, K)$, where $x \in \mathbb{R}^D$ is the observed data point, $y = f = [f(x_1),\ f(x_2), \dots, f(x_D)]$, where $x_1, x_2, \dots, x_D$ are the values taken by the random variables $[X_1, X_2, \dots, X_D\ ]$, μ is the mean vector given by $[\mu(X_1), \mu(X_2), \mu(X_3), \dots, \mu(X_D)]$ and K is the kernel matrix [47]. In Gaussian process, there are majorly 2 types of observations, viz. noise free observation and noisy observation. In case of noise free observations, the sampled function (or the observed training label y) will be present exactly at point x. This is given by $y_i = f(x_i)$. Let X represent the training set that is a matrix of P number of points having D dimensions and let X_* represent the test set that is a matrix of Q number of points having D dimensions. The joint distribution of the actual output, given by f and prediction of the test points, given by f_* is [49]:

$$\begin{bmatrix} f \\ f_* \end{bmatrix} \sim N\left(0, \begin{bmatrix} K(X,X) & K(X,X_*) \\ K(X_*,X) & K(X_*,X_*) \end{bmatrix}\right)$$

where $K(X, X_*)$ is the P X Q matrix of covariances calculated for every pair of the training and test data point. Similar are the interpretation of the other matrices $K(X, X)$, $K(X_*, X)$ and $K(X_*, X_*)$. Using these matrices, the prior knowledge that the training data provides about the function can be incorporated to build the posteriors. From here the posterior distribution can be calculated by conditioning the predictions of the Gaussian process on the training set and test set matrices along with the actual output, f. This conditional probability distribution is given by [49]:

$$f_*|X_*, X, f \sim N(K(X_*,X)K(X,X)^{-1}f, K(X_*,X_*) - K(X_*,X)K(X,X)^{-1}K(X,X_*))$$

Thus, the output values f_* of the test data, that can be plotted as functions, can be sampled from the above normal distribution with mean = $K(X_*,X)K(X,X)^{-1}f$ and variance = $K(X_*,X_*) - K(X_*,X)K(X,X)^{-1}K(X,X_*)$ [49]. On the other hand, in case of noisy observations, which is a more practical case, the function values are not available directly. Only a distorted version of the true function values is available and is given by $y_i = f(x_i) + \in$, where $\in$ is considered to be some Gaussian noise given by $\in \sim N(0, \sigma^2)$. The joint distribution of the actual output, given by y and the prediction of the test points, given by f_* is [49]:

$$\begin{bmatrix} y \\ f_* \end{bmatrix} \sim N\left(0, \begin{bmatrix} K(X,X) + \sigma^2 I & K(X,X_*) \\ K(X_*,X) & K(X_*,X_*) \end{bmatrix}\right)$$

where the matrices of the covariances have the same meaning as in the earlier case and I is the identity matrix. The posterior of the predictions in this case is given by [49]:

$$f_*|X_*, X, y \sim N(K(X_*, X)[K(X, X) + \sigma^2 I]^{-1} y, K(X_*, X_*) - K(X_*, X)[K(X, X) + +\sigma^2 I]^{-1} K(X, X_*))$$

Thus, in both the cases, different samples can be created from the posterior distribution of functions. The average of these functions is considered the prediction of the model as shown in fig. 17.

Chapter 4

Methodology and Results

4.1 General Idea of the Data Sets and Feature Selection

The focus of this work is to examine how future price trends of Bitcoin can be predicted from past price trends using different models. Hence, the simplest of the available Bitcoin price datasets have been considered in this research. A sample of the Bitcoin dataset that has been downloaded from https://www.coindesk.com/price/bitcoin has been used in this research has been shown in table 3:

Currency	Date	Closing Price (USD)	24h Open (USD)	24h High (USD)	24h Low (USD)
BTC	01-10-2013	123.65499	124.30466	124.75166	122.56349
BTC	02-10-2013	125.455	123.65499	125.7585	123.63383
BTC	03-10-2013	108.58483	125.455	125.66566	83.32833
BTC	04-10-2013	118.67466	108.58483	118.675	107.05816
BTC	05-10-2013	121.33866	118.67466	121.93633	118.00566

Table 3: sample of the original Bitcoin data set

From this dataset, only the columns ‘date’ and “closing price (USD)” have been used and rest of all the other columns have been dropped. This is because the other columns in the data set won’t make any difference in the prediction since they would be passing the same kind of information and similar values to the model. Also, our motive is not to provide any hint to the model regarding the price of a particular date while feeding the test data in the model. If the dropped columns are fitted into the model for training, they will also have to be fed into the model as part of the test data and hence, the model will be fed with a major clue regarding the closing price prediction. Thus, prediction of the models will exceptionally high, which will be misleading. The data set contains Bitcoin price data from November 2013 till mid-July 2021. The Bitcoin price trend has been shown in figure 18.

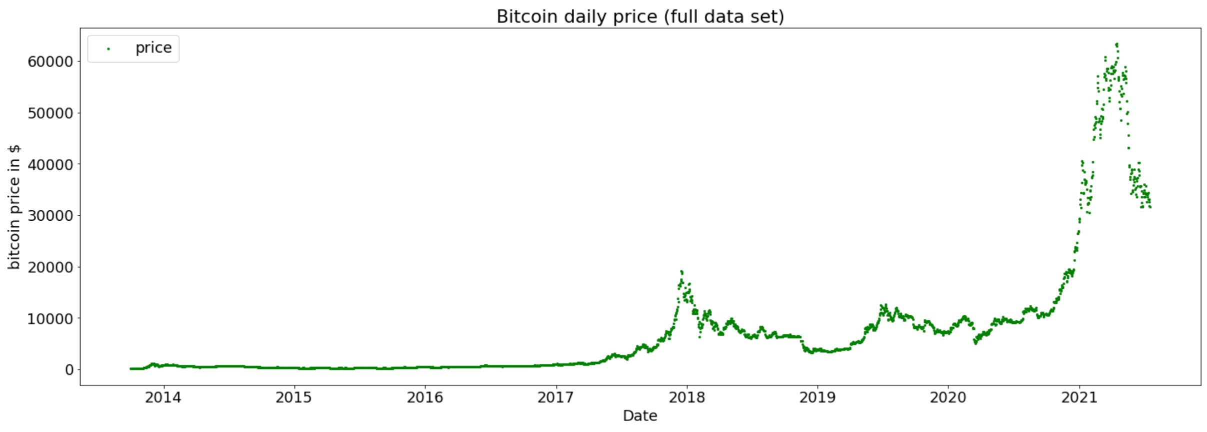

Figure 18 showing Bitcoin price trend from Nov-2013 till Jul-2021

At the same time, since Bitcoin price does not have any kind of seasonality as evident from the trend in figure 18, making accurate price predictions based on such less information (using history price data) will be extremely difficult for any model. As a result, this could lead to doubts regarding the general efficacy of the model in predicting time-series data. Hence, in order to analyse whether prediction of the models is low due to model architecture or due to the data trend, another data set has been used as an auxiliary data set. This data set is the electricity power consumption dataset for Germany from the year 2006 to 2017 and has been downloaded from https://www.kaggle.com/mvianna10/germany-electricity-power-for-20062017. A sample of the data set has been shown in table 4:

Date	**Consumption**
01-01-2006	1069.184
02-01-2006	1380.521

03-01-2006	1442.533
04-01-2006	1457.217
05-01-2006	1477.131

Table 4: sample of the electricity power consumption data set

As can be seen from table 3 and 4, the data that is being fed into the model is very similar. Also, there are no missing data in both the datasets. Both are time-series data with date in the daily date format. And in both cases, apart from date, only that quantity is being fed into the model whose prediction will be made by the models, that is Bitcoin price in the Bitcoin data set and consumption in the electricity power consumption data set. The only differences that exists between both the data sets are that spread of the data around near-by dates is almost nil except in the region after Jan-2021, whereas there is considerable spread in the data throughout the electricity consumption data set and that the electricity consumption data set has a considerable seasonality as can be seen from figure 19. The reason why this data set has been chosen is that it will provide a very good analysis and comparison ground between the model predictions on these 2 datasets.

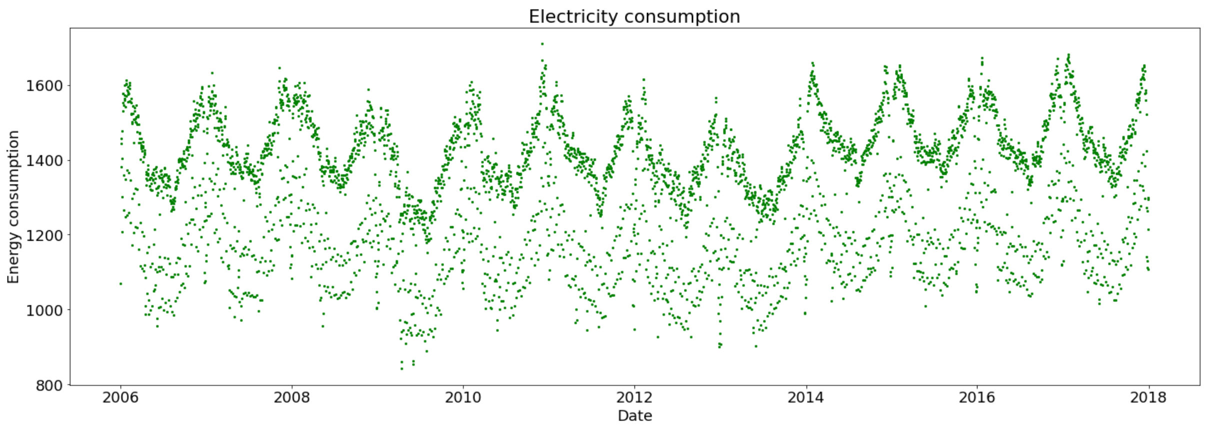

Figure 19 showing energy consumption trend from Jan-2006 till Dec-2017

4.2 Feature Engineering and Cross-Validation

We have already seen that the only information that is being sent into the model as part of the training set is the Bitcoin price corresponding to each date. Thus, in order to increase the amount of information getting into the model, 2 more features are introduced. These new features are built using the available Bitcoin price. The features that being added are: yesterday’s price and the difference between yesterday's price and the day before yesterday's price. As a result, the model gets a reference of the Bitcoin price for 3 days in total. Hence, the predictors are: 1. Date, 2. yesterday's price and 3. yesterday's price – day before yesterday's price. It is to be noted that exactly the same feature engineering has been carried out for the auxiliary data set as well.

The cross-validation technique that is used for the train-test split is called the Forward Chaining Strategy. This technique particularly produces better results in case of time series data than k-fold cross validation. In fact, it is not wise to use the k-fold cross validation technique in case of time-series data. This is because in case of k-fold cross validation, the test set that is formed by data that comes later in the data set can be used to train the model and can be used to make predictions for the data that comes earlier in the data set. However, this technique will not give us a good estimation of the model performance in case of time-series data. This is because, in time-series data, it does not make sense to train the model on future data and make predictions for past dates. To counter this problem, forward chaining strategy has been used. The difference between k-fold cross validation and forward chaining strategy with 5 splits has been shown in figure 20, where the red blocks resemble the training set and the yellow blocks resemble the validation set. It can be clearly seen that the training set is formed only in a forward direction with respect to time and the predictions are always made for future dates in the forward chaining strategy.

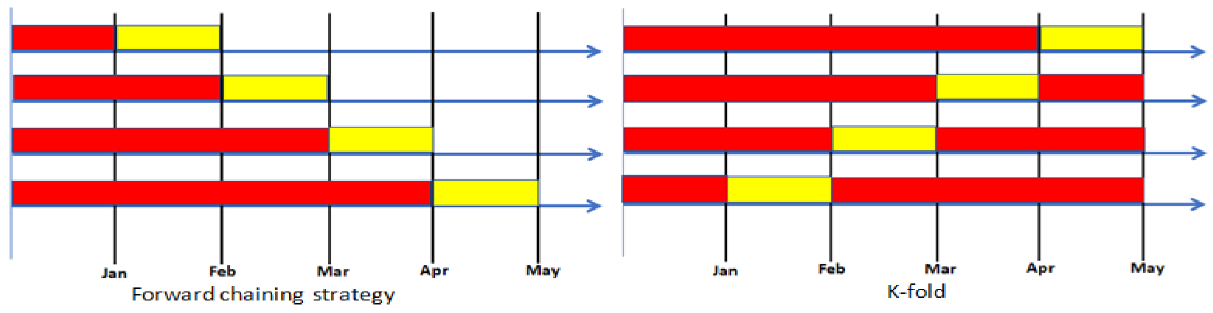

Figure 20 showing the difference between Forward chaining strategy and k-fold

On performing a forward chaining cross validation (splits = 10) spot check comparison between the KNN, Random Forest, Kernel Ridge Regression and Gaussian Process Regression models using the default parameters available in the sklearn library for these models, it is observed, from the mean absolute error box plots in figure 21, that though the performance of the KNN, Random Forest and Kernel Ridge models on the Bitcoin price data is similar with respect to the median of the error, the Kernel Ridge model performs better overall since the mean error is the least. Performance of the Gaussian Process Regression model, as evident from the box plot, is the least impressive. When the same comparison, as shown in figure 22, is done on the electricity comparison data set, it is observed that the KNN and the Random Forest models perform much better. Again, performance of the Gaussian model is very poor. This check, in a way, gives us some idea on what to expect in the comparison analysis between these 3 models when predictions would be made using the best model parameters and hyper-parameters. These models also perform as individual baseline models.

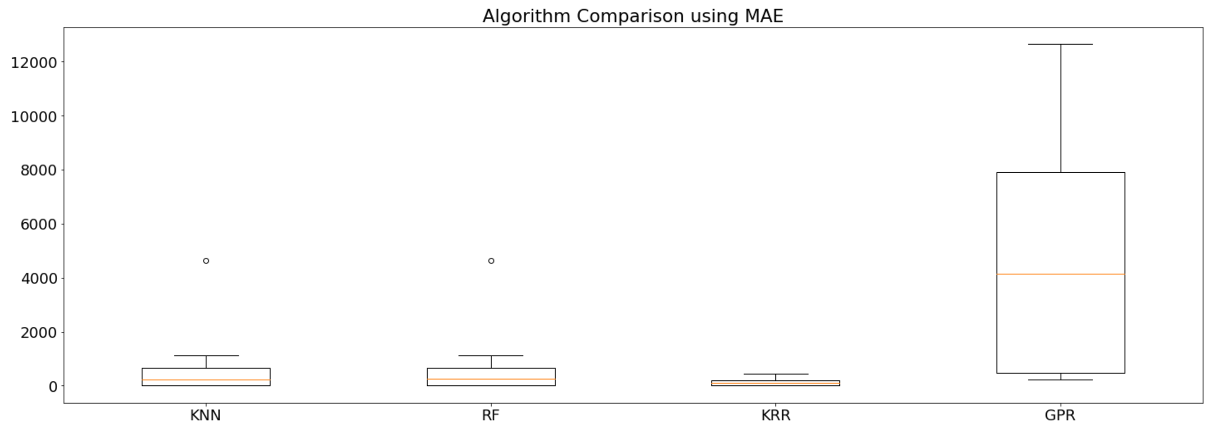

Figure 21 showing the box plot comparison for prediction errors on the Bitcoin price dataset

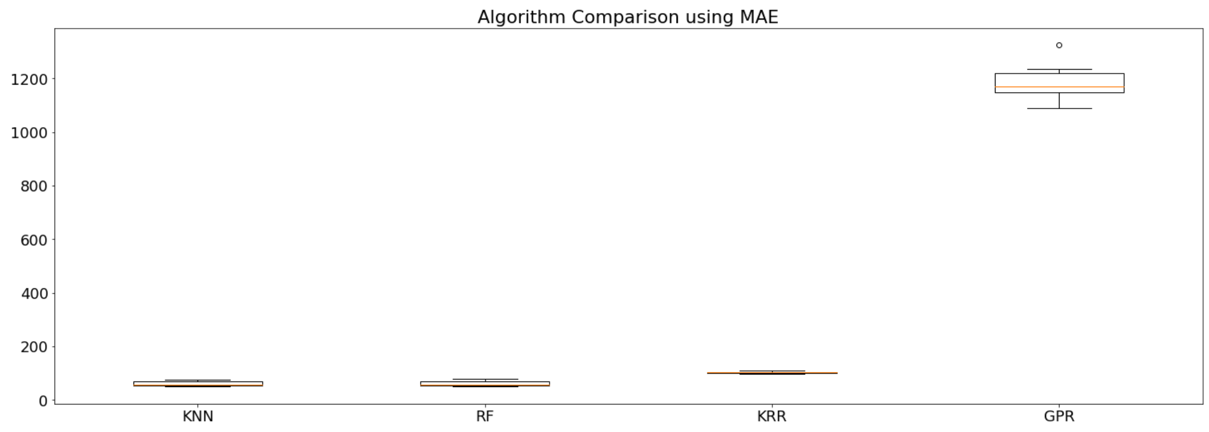

Figure 22 showing the box plot comparison for prediction errors on the electricity consumption dataset

On performing cross validation using model parameters, the best parameters values for the model are selected. Only the important parameters that can majorly influence the performance of a model have been considered. After performing the forward chaining cross validation, the data set is divided into training and test sets. The data set is divided in such a way that all data till December 31st, 2020 will be used for training and the data after that till July 17th, 2021 will be used as the test data set. In the Bitcoin data set, there are 2844 data points in total, of which 2646 data points are included in the training set and 198 data points are included in the test set. Similarly, of the electricity consumption data set, data till December 31st, 2020, which has 4016 data points will be included in the training set. On the other hand, the test set contains data for the 2017 and has 365 data point.

4.3 Measure of Model Comparison

The measure chosen to compare the models is Mean Absolute Percentage Error (MAPE). This is given by:

$$\frac{\sum_{i=1}^{n}\left[\left(\frac{|y_i-\hat{y}_i|}{y_i}\right)*100\right]}{n}$$

Where, n is the total number of samples, y_i is the actual value of the dependent variable and $\hat{y}_i$ is the prediction against y_i. Here, MAPE has been chosen because it is the best measure when a comparative analysis of different model or of different data sets. This is because measures such as root mean square error, mean square error or mean absolute error cannot be used to compare model performance when used on different data sets since these measures will be on different scales when used on different data sets.

4.4 Performance Evaluation of the Random Forest Regression Model

The grid of parameters and parameter values to select the model have been shown below.

```
'n_estimators': [20, 30, 50, 100],

'max_features': ['auto', 'sqrt', 'log2'],

'min_samples_split': [20, 25, 30, 50],

'min_samples_leaf': [ 20, 25, 30, 50],

'max_depth' : [i for i in range(5,15)],

'max_samples': [30, 50, 75, 100, 200]
```

In the grid above, ‘n_estimators’ is the number of trees in the forest, ‘max_features’ is the number of features that will be considered while making the split. This number will either be the actual number of features present, or the square root of the total number of features or the log of the total number of features depending on the algorithm chosen (auto, sqrt, log2 respectively). ‘min_samples_split’ is the number of the samples required to split a node, ‘min_samples_leaf’ is the number of samples required for a node to be considered a leaf node, ‘max_depth’ is the maximum of depth of the tree, that is the number of level till which the tree will be drilled down and ‘max_samples’ is the number of samples that will be pulled out from the actual data set to build the trees in case of bootstrapping. The bootstrap parameter takes in Boolean values and the default value is True. The same grid has been used for both the data sets. The measure that is used to compare the quality of the cross validation split is mean squared error [50].

The best model that comes out of the cross validation on the grid parameters for the Bitcoin price prediction data set is: `bootstrap = True, max_depth = 10, max_features = 'auto', max_samples = 200, min_samples_leaf = 20, min_samples_split = 30, n_estimators = 20`. The best model for the electricity consumption data set is: `bootstrap = True, max_depth = 13, max_features = 'auto', max_samples = 200, min_samples_leaf = 20, min_samples_split = 30, n_estimators = 30`. On making the test train split, the MAPE of the best model on the Bitcoin price prediction data is 69.62%, which is definitely not a good accuracy. The prediction on the test data has been shown in figure 23. On the other hand, the MAPE of the best model on the electricity consumption data is 5.84%. The prediction on the test data of the electricity consumption data set has been shown in figure 24.

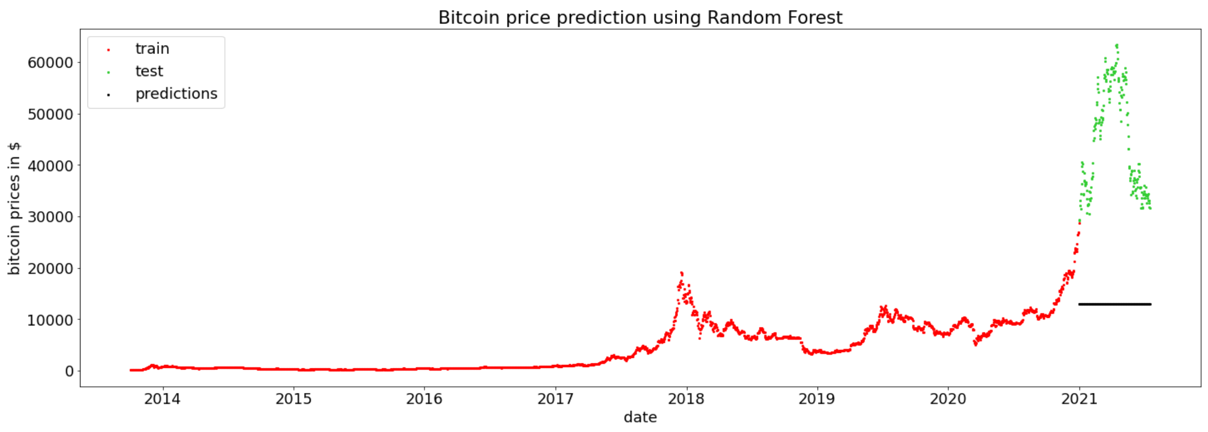

Figure 23 showing the prediction on the test data of the Bitcoin price data set using Random Forest

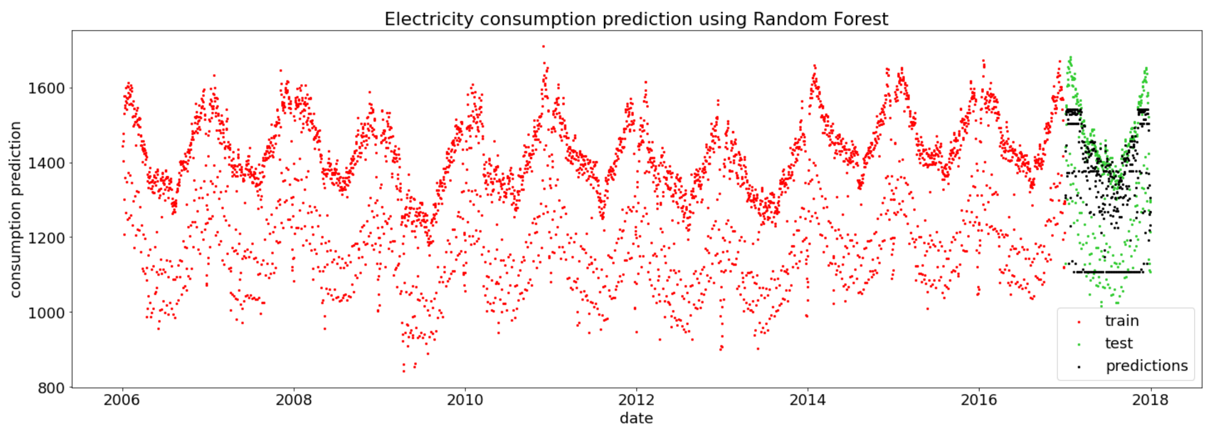

Figure 24 showing the prediction on the test data of the electricity consumption data set using Random Forest

It can be seen from figure 23 and figure 24 that Random Forest Regressor has a tendency to make flat predictions. In the electricity consumption data set, since the data is spread out, the model makes flat predictions on multiple levels that balance out each other and hence the overall error is less. On the other hand, in case of the Bitcoin price test data set, the data does not have any 'overall' significant spread at all and hence, the prediction is flat in only one level, thus adversely affecting the accuracy of the model. It is unlikely that seasonality in the electricity consumption data plays any role in determining accuracy of the model since Random Forest Regressor does not have any parameter that captures the seasonality in the data. Because of the tendency of the model to make flat predictions, Random Forest might not be a good choice to work with time-series data.

4.5 Performance Evaluation of the K-Nearest Neighbour Model

The grid of parameters and parameter values to select the model have been shown below.

```
'n_neighbors': [2, 5, 7, 10, 20, 30, 50, 100],

'weights': ['uniform','distance'],

'p': [1,2]
```

In the grid above, 'n_neighbours' is the number of neighbours that will be considered to make the prediction for a given training data point, 'weights' is amount of contribution of each nearest neighbour towards calculating the prediction of a given training data point. The amount of contribution of these points may vary with distance from the given training data point based on the type of weight selected. When weight = 'uniform', the weight of the nearest neighbours remains constant with increase on distance. When weight = 'distance', the weight of the nearest neighbours decreases as the distance increases from the reference training data point. Here too, the same grid has been used for both the data sets. The measure that is used to compare the quality of the cross-validation split is mean squared error. 'p' represents the kind of distance that will be measured. p = 1 represents Manhattan (l1) distance and p = 2 represents Euclidean (l2) distance [51].

The best model that comes out of the cross validation on the grid parameters for the Bitcoin price prediction data set is: `n_neighbors = 5, p = 2, weights = 'uniform'.` The best model for the electricity consumption data set is: `n_neighbors = 20, p = 2, weights = 'distance'.` On making the test train split, the MAPE of the best model on the Bitcoin price prediction data is 36.37%, which is definitely not good, but almost twice as good as the Random Forest Regresor. The prediction on the test data has been shown in figure 25. On the other hand, the MAPE of the best model on the electricity consumption data is 4.02%. The prediction on the test data of the electricity consumption data set has been shown in figure 26.

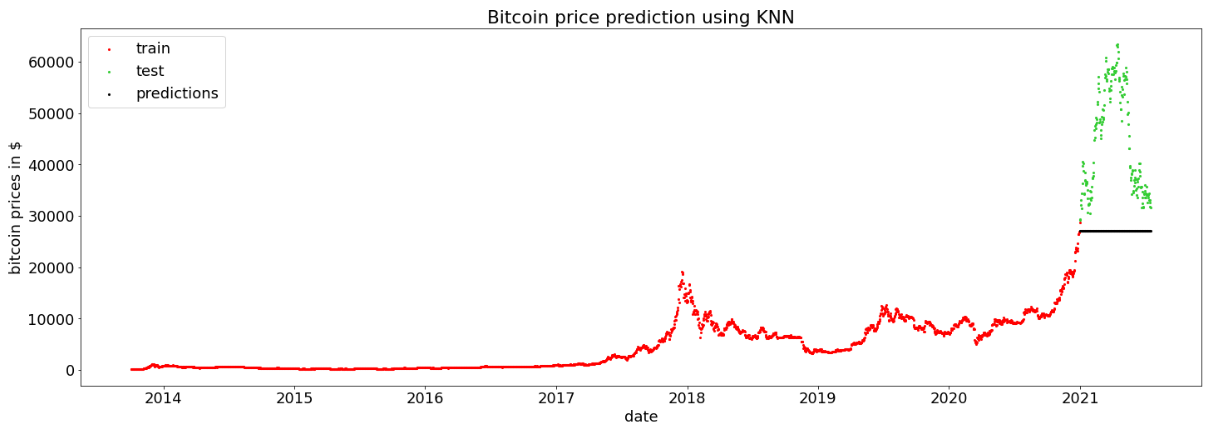

Figure 25 showing the prediction on the test data of the Bitcoin price data set using KNN

As can be seen from figure 25, the prediction is twice as good as in the case of Random Forest because the prediction level, even though flat, has risen almost halfway of the y-axis, compared to Random Forest. However, the flat predictions, as seen in the case of Random Forest Regressor, may not be characteristic of this model since there is no sign of any flatness in the in the model prediction for the electricity consumption data set. It can be seen from figure 26 that when the data has some spread in it, the prediction accuracy on the test data is pretty high. Hence, it can be said that KNN works well with time-series data, but only when there is some spread in the data as can be seen in the electricity consumption data set in figure 26.

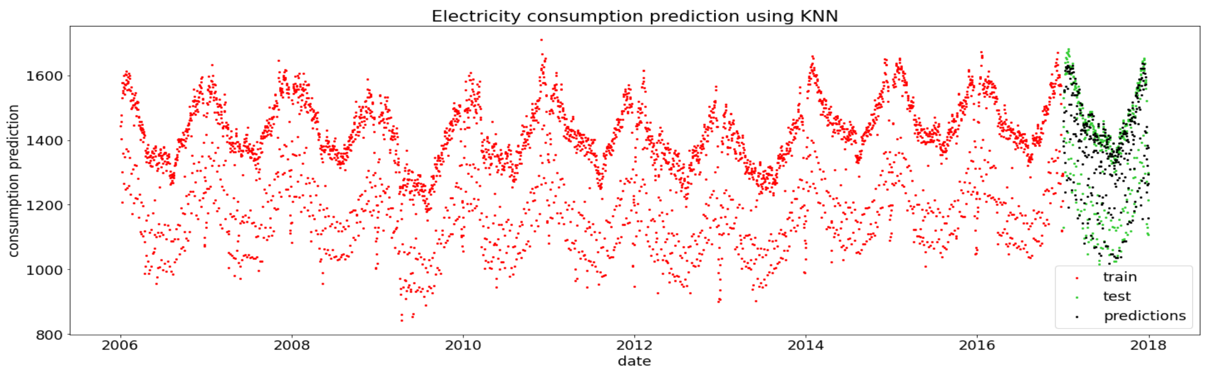

Figure 26 showing the prediction on the test data of the electricity consumption data set using KNN.

4.6 Performance Evaluation of the Kernel Ridge Regression Model

The grid of parameters and parameter values to select the model have been shown below.

```
'alpha': [10, 1, 0, 0.1, 0.01],
'gamma': [0,1,5,10,25],
'kernel': ['rbf', 'linear','laplacian', 'poly', 'sigmoid']
```

In the grid above, 'alpha' determines the amount regularisation that will be imposed on the model. This parameter is used to control overfitting and underfitting in the model. The formula for alpha is $alpha = \frac{1}{2C}$, where C in the penalty parameter. Gamma is the parameter that determines how fast the value of the kernel decreases as the distance of the nearest neighbours from the training data point increases. Kernel hold the kind of prior information that we are feeding into the model about the training data set. This is done by calculating the covariance between random variables or the predictor variables. There are different ways in which the covariance can be calculated and this is defined by the values passed in the kernel parameter. For example, the rbf kernel, also called the Radial Basis Function kernel, is given by $k(x, x') = \exp\left(-\frac{(x-x')^2}{2l^2}\right)$, where '$l$' determines how long will the curves be in the function generated. Linear kernel is given by $k(x, x') = x^T x'$, polynomial kernel is given by $k(x, x') = (x^T x' + 1)^d$, where d is the order of the polynomial [52][53][54]. Kernels can be manually designed functions too, but in this case we wanted to study the effect of the inbuilt kernels provided with Kernel Ridge. The manually designed kernel functions will be studies later under Gaussian Process Regression.

The best model that comes out of the cross validation on the grid parameters for the Bitcoin price prediction data set is: `alpha = 0.1, gamma = 0, kernel = 'linear'.` The best model for the electricity consumption data set is: `alpha = 10, gamma = 10, kernel = 'poly'.` On making the test train split, the MAPE of the best model on the Bitcoin price prediction data is 3.45%, which is a major improvement over Random Forest Regressor model and KNN. The prediction accuracy of the best model on the electricity consumption data is: 7.67%. It can be seen from figure 27 and 28 that the model performs exceptionally well on both the data sets. However, surprisingly the performance of the algorithm is better on the Bitcoin data set than on the electricity data set. It is particularly surprising to note that the linear kernel is able to push the model accuracy to such high extent in case of the Bitcoin price data set in spite of very limited information regarding Bitcoin being into the model. However, there is definitely a chance that customised kernel functions could perform even better than the built-in kernels on these data sets. Thus, we see that the initial trend that was observed between the skeleton of the KNN, Kernel Ridge and Random Forest models, where in the performance of the Kernel Ridge model is almost at par with that of KNN and Random Forest as shown in figure 23, does not hold anymore for the Bitcoin data. It can be clearly concluded that performance of KNN can be much better than of Random Forest and that the performance of the Kernel Ridge algorithm can be increased many folds when used with right kernels. At the same time, the Kernel Ridge model is able to properly predict the rise and fall in the Bitcoin price. In financial terms, this rise and fall is referred to as a bubble.

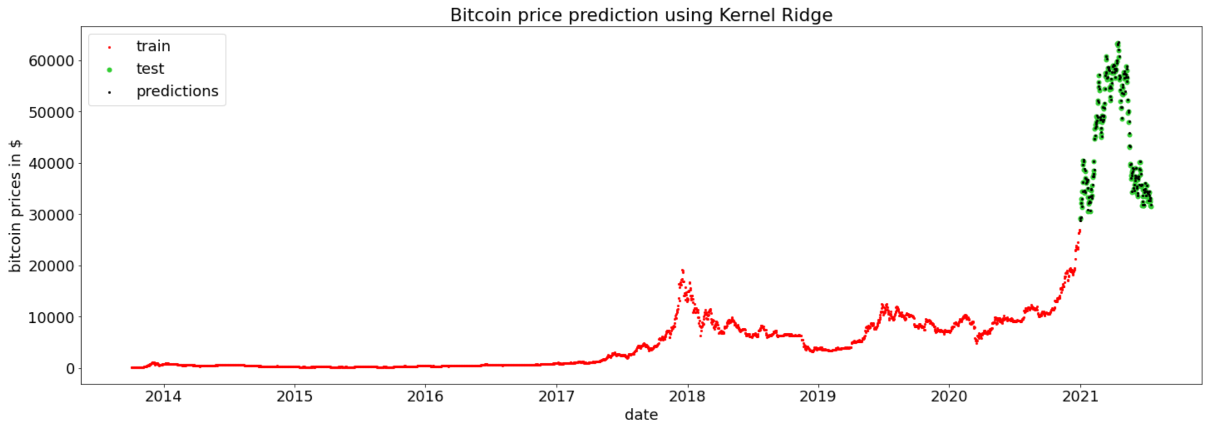

Figure 27 showing the prediction on the test data of the Bitcoin price data set using Kernel Ridge

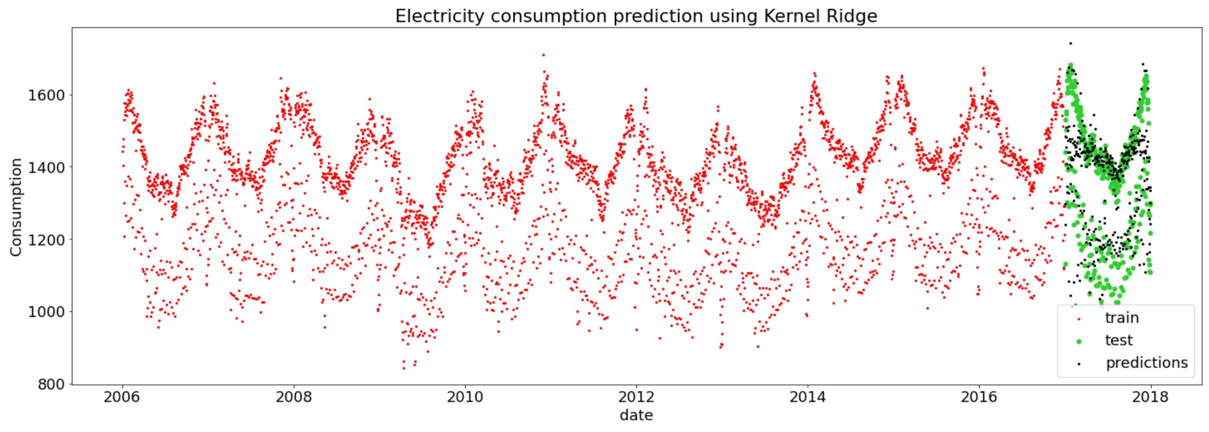

Figure 28 showing the prediction on the test data of the electricity consumption data set using Kernel Ridge

4.7 Performance Evaluation of the Natural Cubic Spline Model

The only parameter that is tuned in case of Natural cubic spline is the number of knots. This parameter controls overfitting in the model. Too many knots, though would reduce the MAPE on the training data, would also make the model over-sensitive to the training data. Hence, this MAPE would be misleading and we will have to select the best model by check the model prediction on the test data. The number of knots that the model is tested with are: 25, 50, 150 and 250. The model that produces the best MAPE on the test data is the one with number of knots = 50. The behaviour of the different models with different number of knots on the training data is shown in figure 29. The placement of the knots is equidistant throughout the range of the training data.

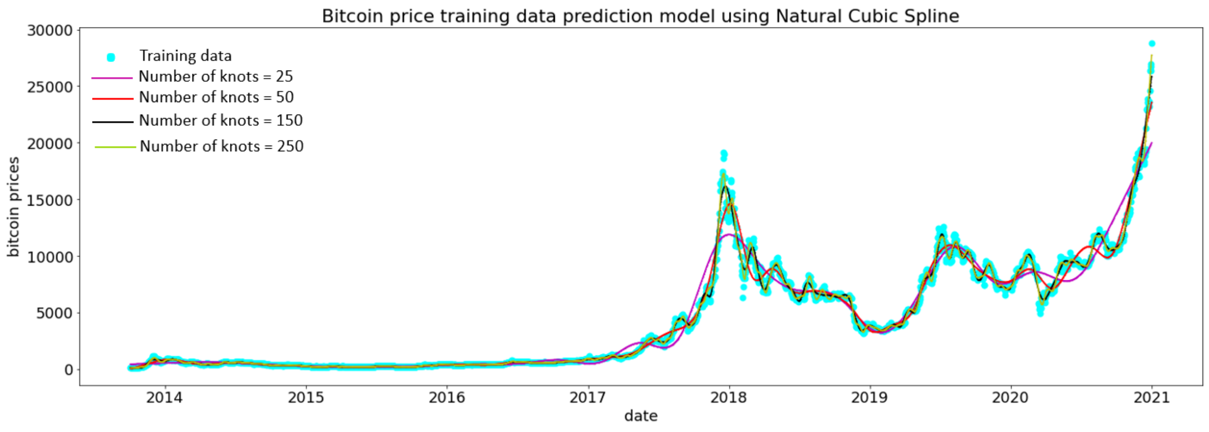

Figure 29 showing how models with different number of knots capture the behaviour of the training data of the Bitcoin data set

From figure 29, it is clear that the model with number of knots = 50 (red) would be the best model to make the prediction on the test data. This model is able to successfully capture the general trend of the data. Figure 30 shows the prediction of the best model both on the training data and on the test data. The MAPE of the model on the training data is 8.4%, which increases drastically to 31.9% on the test data. Also, clearly, though the model is able to capture the initial rising trend of the test data, it fails to capture the formation of the bubble and also the eventual slump in the Bitcoin prices. Also, it is observed that for none of the models, the prediction is in the form of a cubic spline. Rather, the model makes a linear prediction for all the different number of knots, which is not desired.

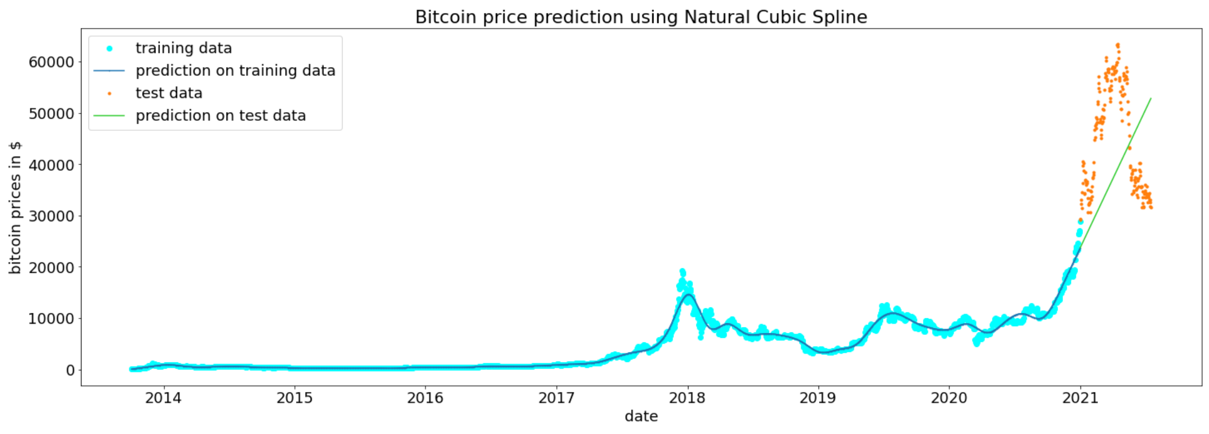

Figure 30 showing the prediction of the best model on the training and test data

Using the same methodology on the electricity consumption data set, the models developed for the same number of knots, as in the case of the Bitcoin price dataset, have been shown in figure 31. Based on the MAPE of these models on the test data, the best model comes out to be the one with number of knots = 25. The prediction of the model on the training and test data has been shown in figure 32. It is clearly seen that in the of this data set, the prediction of the model is a cubic spline. Hence, it can be said that it is difficult for cubic splines to make better predictions on data that does not have any spread in it and probably should not be considered for such kind of data. The MAPE of the best model on the test data is 11.92%. The model is also able to predict the wobbly trends in the test data to an extent. However, an observation here is that the prediction accuracy of the model increases even further when the number of knots is reduced below 25 and when model clearly under fits. This phenomenon has been shown in figure 33. The prediction of the model here is almost linear. The number of knots of the model in this case is 5, but the MAPE is 11.16%, which is less than that when the number of knots is 25 and hence, can be considered a better prediction.

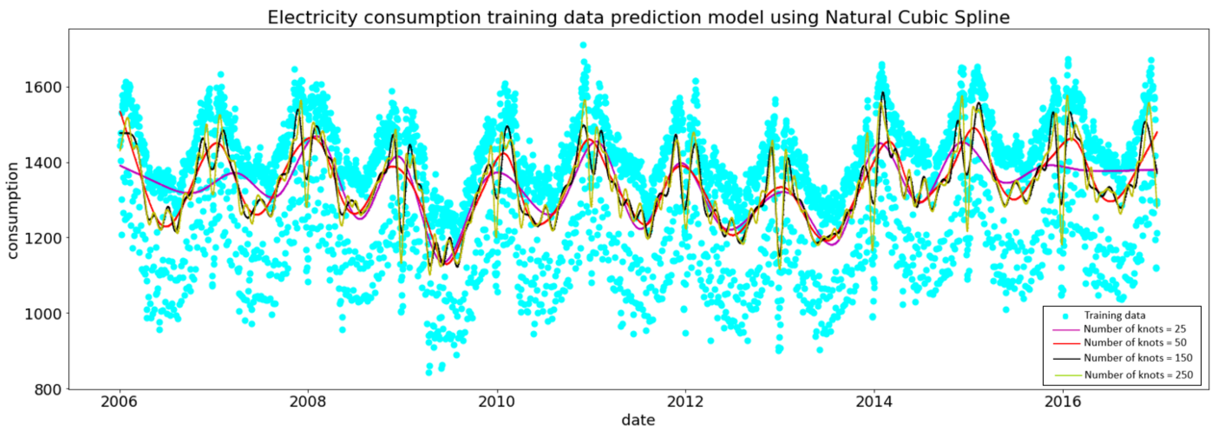

Figure 31 showing how models with different number of knots capture the behaviour of the training data of the electricity consumption data set

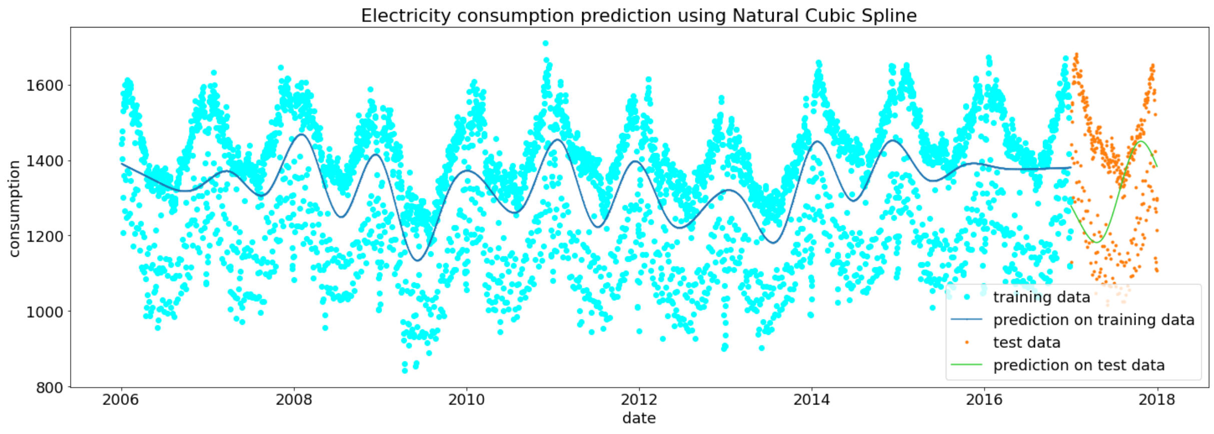

Figure 32 showing the prediction of the best model on the training and test data

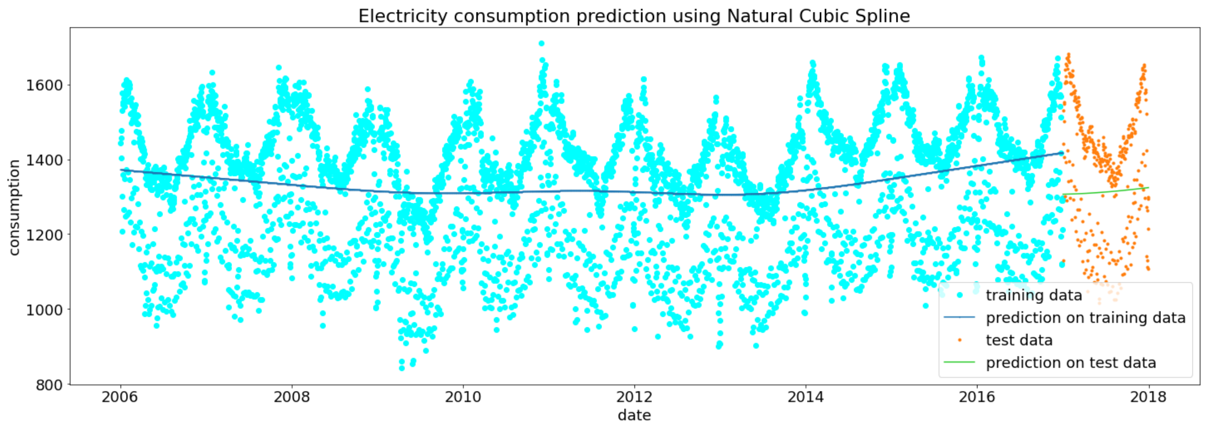

Figure 33 showing the under fitting in the model but decreasing MAPE

4.8 Performance Evaluation of the Gaussian Process Regression Model

While evaluating the performance of this model, 4 different kernel have been used. These kernels are all callable functions and are combination of individual kernels. The kernels that have been used are:

1. `ConstantKernel() + ConstantKernel() * RBF()  + WhiteKernel()`
2. `ConstantKernel() *  RBF()`
3. `DotProduct() + WhiteKernel()`

Performances of each of these models have been discussed in the next sub-section. However, before moving into the individual kernels, we will look into what these individual kernels are that combine to form the kernels noted above. A **Constant kernel** is nothing but a uniform prior that takes a constant value. This value is passed into the function in the form of an argument. This kernel is used to adjust the magnitude of other kernels or adjust the mean of the posterior functions. A constant kernel is given by: $k(x, x') = c, \forall x, x'$ [55]. A **RBF kernel** is also called a squared exponential kernel. It is given by: $k(x, x') = \exp\left(-\frac{d(x,x')^2}{2l^2}\right)$, where $d(x, x')$ is the Euclidean distance between $x, x'$and l determines the length of the curves of the function. The parameter is passed as an argument in the function. This kernel is used when the data is extremely rough and needs to be smoothened. The smoothening property of this kernel come from the fact that this kernel is infinitely differentiable [56]. A **White kernel** is used to add noise to the GP distribution. This noise is in the form of independent and identically distributed points and is given by: $k(x, x') = \sigma^2 I_n$, where I_n is the identity matrix and σ^2 is the variance in the noise. Thus, the covariance matrix that is formed has 0s everywhere except on the diagonal of the matrix. This is because there should not be any sort of correlation between the generated points [57]. The variance in the noise data is passed into the model as an argument. The last individual kernel that has been experimented with is the **Dot product kernel**. A dot product kernel is given by: $k(x, x') = x.x' + {\sigma_0}^2$. This is nothing but the dot product of $x \, and \, x'$. This kernel is clearly a linear combination of the product of the D dimensions of $x \, and \, x'$, along with a variance parameter ${\sigma_0}^2$. This parameter controls whether the kernel is homogenous or not, a concept that comes from the "systems of equations", which is divided into homogeneous systems given by Ax + By + Cz = 0 and non-homogeneous systems, given by Ax + By + Cz = P, where P is any constant. ${\sigma_0}^2$ is the variance of the normal prior that is imposed on the bias. In our research, we have used some of the common combinations of these kernels and their default parameter values.

GPR Model with ConstantKernel() + ConstantKernel()*RBF() + WhiteKernel (): This kernel has been formed using a combination of 3 different kernels that are in the form of callable functions. These kernels are constantkernel(), WhiteKernel() and RBF(). The GPR model with this kernel produces a MAPE of 70.27% while predicting on the Bitcoin price test data set and a MAPE of 3.97% while predicting on the electricity consumption data set. The prediction of model on the Bitcoin and electricity consumption, have been shown in figures 34 and 35 respectively. Figure 36 (a) and (b) capture the error in the prediction between the true value and the prediction for both the datasets.

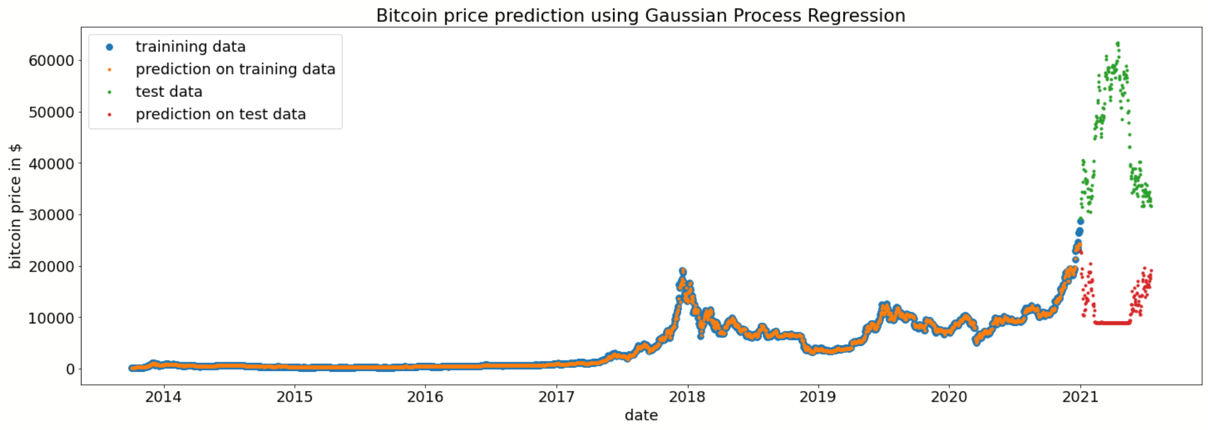

Figure 34 showing the prediction of the model on the Bitcoin price test data set

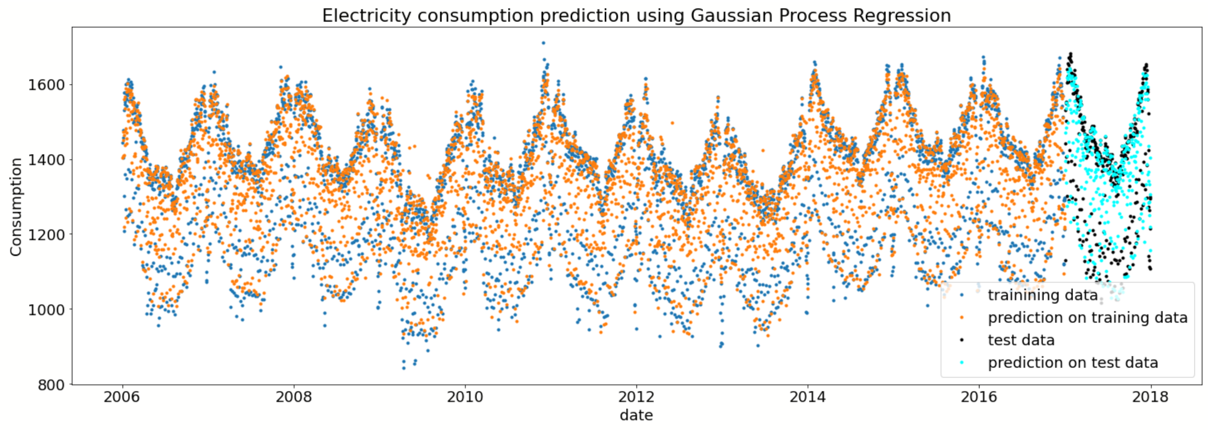

Figure 35 showing the prediction of the model on the electricity consumption test data set

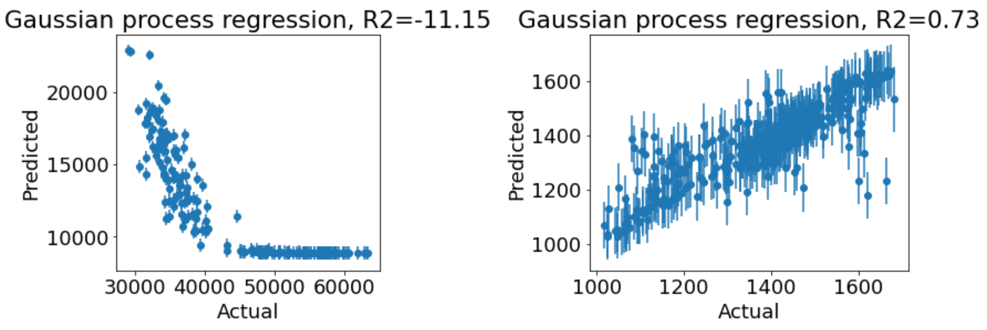

Figure 36(a) (left), 36(b) (right) showing the error plot of the bitcoin price and electricity consumption data set respectively

The flat prediction in figure 34 for a range in the data set is due of the combining effect of the RBF kernel, that has a tremendously smoothening capability, and the constant kernel, which is basically a uniform prior. The error plot precisely captures the behavioural difference in the prediction of the model on both the data sets. In the second data set (figure 35), because the data has a considerable spread and is noisy to an extent, the smoothening capacity of the RBF kernel is counterbalanced by the white kernel. The negative correlation in figure 36(a) is due to the opposite prediction trend of the model on the test data to the actual value as can be seen in figure 34. On the other hand, because the prediction of the model on the electricity consumption data has high accuracy, the R2 value in this case close to 1 as can be seen in figure 36(b). However, since variation in the error bar of the model prediction in the second data set is higher than that in the first data set, the confidence in the prediction values in figure 36(b) is lower than that in figure 36 (a), though there is no doubt regarding the correlation of the predictions being positive with the actual values in 36(b). Thus, it can be said that the Gaussian Process Regression model with "ConstantKernel() + ConstantKernel()*RBF() + WhiteKernel ()" kernel will produce better results when there is some spread in the data as in the case of the Electricity Consumption data set.

<u>GPR Model with ConstantKernel() * RBF():</u> This kernel has been formed using 2 different kernels are Constantkernel() and RBF(). The GPR model with this kernel produces a MAPE of 89.81%, which is a very poor accuracy, while predicting on the Bitcoin price test data set and a MAPE of 9.89% while predicting on the electricity consumption data set. The prediction of model on the Bitcoin and electricity consumption, have been shown in figures 37 and 38 respectively. Figure 39 (a) and (b) capture the error in the prediction between the true value and the prediction for both the datasets. The main difference between the previous kernel and this kernel is the absence of the white kernel is this case. The effect of this absence can be clearly seen in the prediction in figure 37, where the prediction of the model on the test data is perfectly flat. Also, the prediction on the electricity data set is also constant around most of the test data points.

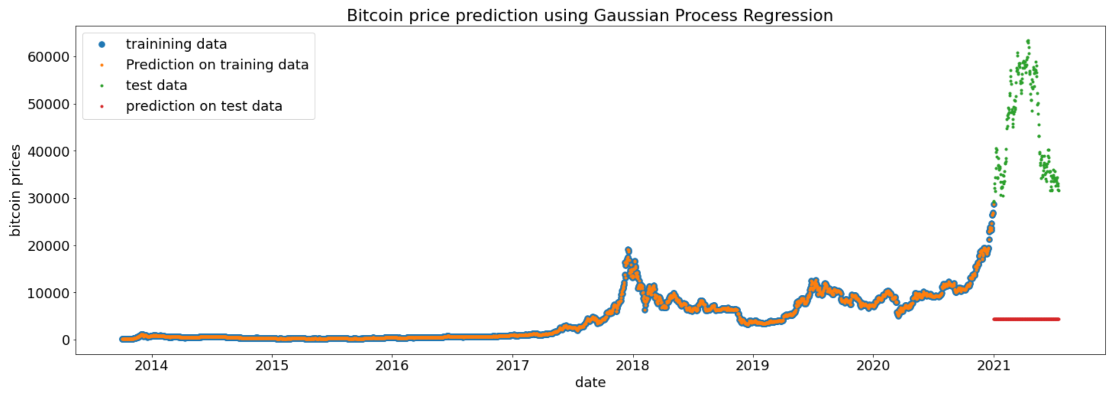

Figure 37 showing the prediction of the model on the Bitcoin price test data set

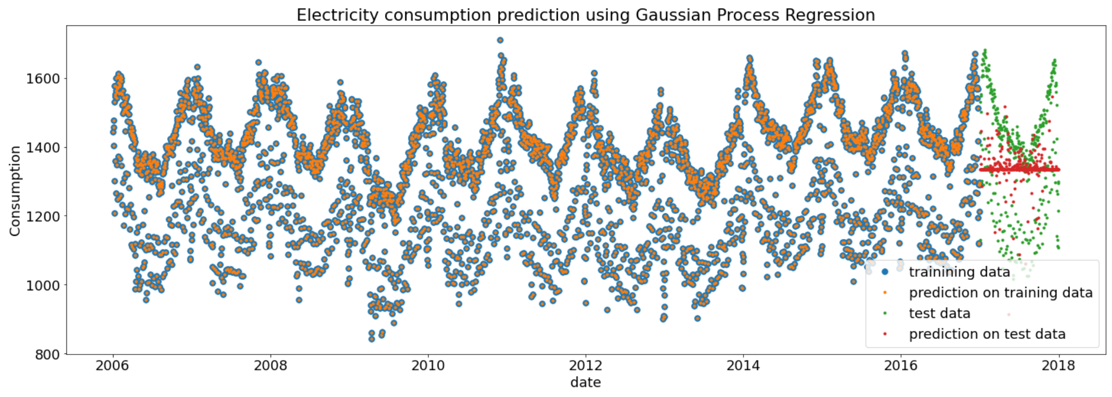

Figure 38 showing the prediction of the model on the electricity consumption test data set

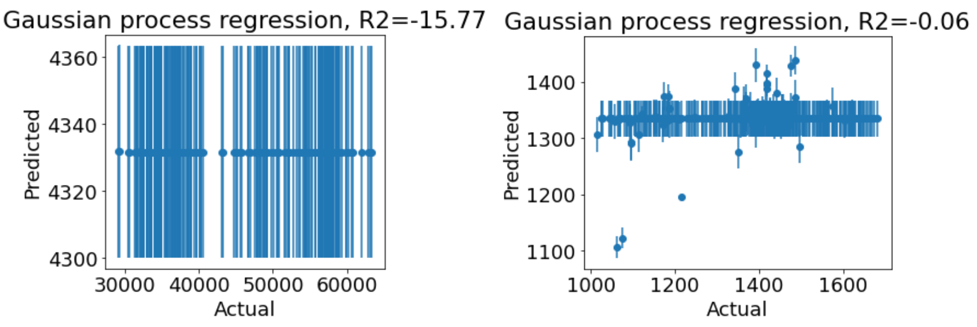

Figure 39(a) (left), 39(b) (right) showing the error plot of the bitcoin price and electricity consumption data set respectively

From figure 39(a), it can be seen that the error bar is flat, which is expected from the kind of prediction that the model makes in figure 37. Also, there is no confidence in the model predictions in case of the Bitcoin data set. The negative R2 value in the error bar of both the data sets show that there is no correlation between the predicted values and the actual test data. However, since the R2 value in the second data set is very close 0, it can be said that the prediction is almost equivalent to the mean of the test data points, something that is evident from the prediction of the model in figure 38. It can be clearly seen that the predictions made by the model on most of the test data point lie half-way through the span of the consumption axis (y-axis) of the data. As a result of which, even though the MAPE of the model on the electricity data set is not very bad, the quality of the prediction made is not good at all. Hence, it can be said that this kernel is not a good choice for making predictions on time series data.

GPR Model with DotProduct() + Whitekernel(): This kernel has been formed using 2 different kernels are DotProduct() and WhiteKernel(). The GPR model with this kernel produces a very low MAPE of 3.45%, which is a very high accuracy and same as that of the Kernel Ridge model, while predicting on the Bitcoin price test data set and a MAPE of 7.69% while predicting on the electricity consumption data set. The prediction of model on the Bitcoin and electricity consumption, have been shown in figures 40 and 41 respectively. Figure 42 (a) and (b) capture the error in the prediction between the true value and the prediction for both the datasets. This kernel proves to be a very powerful kernel since the accuracy of the model predictions, in both the data sets, is very high in spite of such limited amount of information being fed into the model for training.

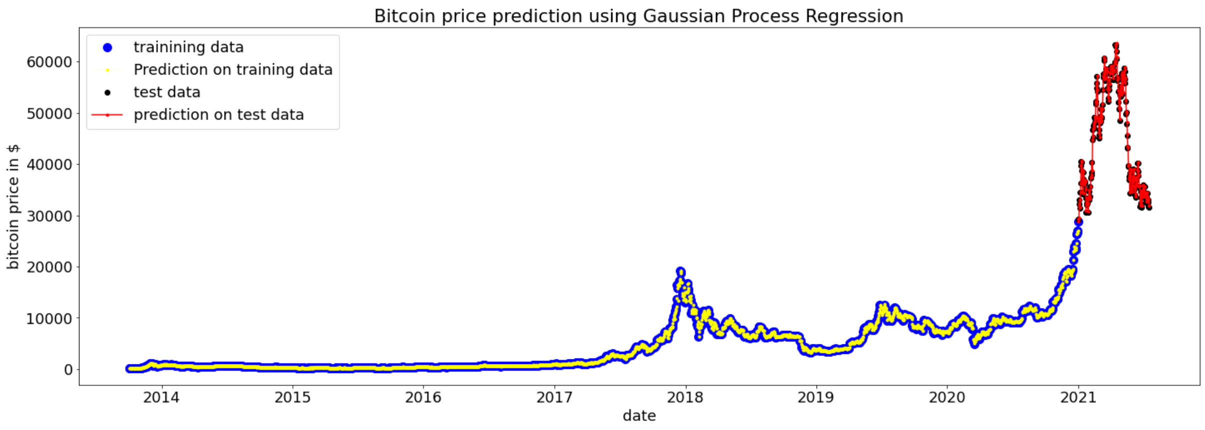

Figure 40 showing the prediction of the model on the Bitcoin price test data set

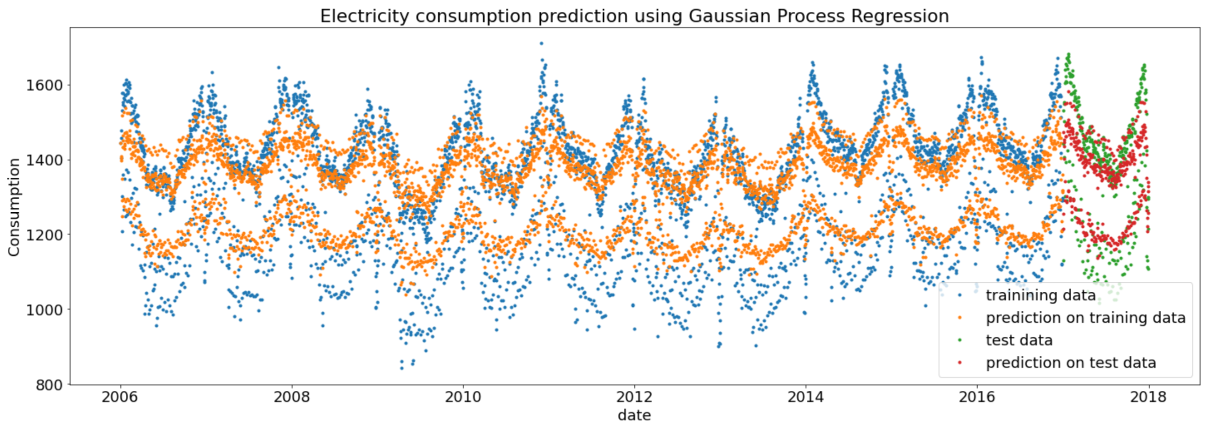

Figure 41 showing the prediction of the model on the electricity consumption test data set

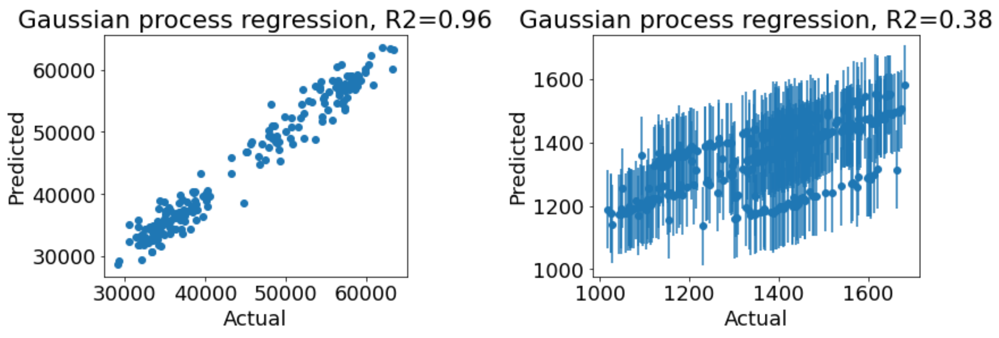

Figure 42(a) (left), 42(b) (right) showing the error plot of the bitcoin price and electricity consumption data set respectively

From figure 40 and 42(a), it can be seen that the prediction of the model using the DotProduct() + Whitekernel() kernel is quite perfect. The R2 value of the error plot of the Bitcoin price data set is close to 1 and this shows that the prediction on the individual training data points is quite high. On the other hand, the R2 value of the error bar plot of the electricity consumption data set is quite low. This tells us that the even though the overall prediction quality of the model in this particular data set is not bad (MAPE < 8%), the accuracy of the individual predictions is quite low as can be seen figure 41. Also, it can be seen from the error bar plot in figure 42(b) that the confidence of the predictions is quite low.

Chapter 5

Conclusion and Future work

5.1 Conclusion

From the study carried out, it can be concluded that not all predictive regression models have the capability to accurately predict Bitcoin price based solely on history data and that too when then data in itself is neither periodic nor does it have any significant spread. And the ones that do make precise predictions can be considered really powerful models. It cannot be conclusively said that the models that do not have a good prediction accuracy would not be able to make Bitcoin price predictions. This is because the amount of information fed into the models regarding the data is very less. It is quite possible that if more information, in the form of features, are fed into the models, the prediction accuracy of these models would increase drastically. However, it can definitely be concluded that when it comes to making predictions solely based on history pricing data, some models will simply not work. This point can also be used when working on stocks. Table 5 gives a gist of the model performances from Chapter 4.

A very interesting observation here is that the models that had the best predictions on the Bitcoin data set are Kernel Ridge (with linear kernel) and GPR (with DotProduct() + WhiteKernel()). The similarity between these two kernels is that both the linear kernel and the DotProduct kernel are more or less $x^T x'$, that is a '**linear combination**' of the D-dimension elements of the data points. Hence, it can be concluded that the introduction of priors with linear characteristics has played a pivotal role in achieving such high prediction accuracy of the models.

Model	**Model Accuracy on Bitcoin Data Set (MAPE)**	**Model Accuracy on Electricity Consumption (auxiliary) Data Set (MAPE)**	**Comments**
Random Forest Regressor	69.62%	5.84%	Not suitable to make predictions on time series data set due to flat prediction tendency. Can result in high mean accuracy but that would be misleading since most of the individual accuracy would be way off
K Nearest Neighbour	36.37%	4.02%	Not suitable to make predictions on this particular Bitcoin data set. Can perform better when more Bitcoin related information is fed in
Kernel Ridge	3.45%	7.67%	Near perfect prediction on very limited information. Can be categorised as a powerful model. Able to predict the bubble
Natural Cubic Spline	31.90%	11.92%	Not suitable for the given Bitcoin data set. Makes a very general prediction on the Electricity consumption data set as well. Even though mean prediction accuracy is not bad on the electricity consumption data set, the predictions on the individual level are way off.

Gaussian Process Regression	3.45% - 89.81%	3.97% - 9.89%	Prediction accuracy heavily depends on the choice of kernel. Different kernels have different characteristics and behave differently with different kinds of data set. With the right kind of kernel, this algorithm can be used to model any kind of time-series data.

Table 5: gist of the model performances

5.2 Future Work and Prospects

As mentioned in section 1.2, this study provides the first ground work to a series of research that I plan to carry out in the near future on crypto-currency and then extend the findings on cryptocurrency predictions to stocks and debentures. The scope of this dissertation was limited to studying whether Bitcoin price prediction can be made on the basis of historical price and also analysing the performance of different predictive regression models in predicting future price. The scope has been sufficiently covered in this dissertation. Now that the prediction behaviour of these predictive regression models on future Bitcoin price based solely on historical data is quite clear, the next immediate research would be to study the impact of several other endogenous predictor variables such as Bitcoin volume traded, Bitcoin value traded, number of Bitcoins mined, number of unique Bitcoin transactions and many more. From here on, I wish to extend this research further on to study the exogenous factors that affect Bitcoin prices. Of course, a lot of work has already been done in this sphere, but a holistic study of endogenous and exogenous factors has not been done yet on Bitcoin price prediction. At the same time, research could be done to identify more endogenous and exogenous features that have not yet been considered in the researches and that truly affect prices of cryptocurrencies. Such kind of study will introduce a tremendous amount of predictability on Bitcoin price and make a huge contribution towards making Bitcoin a global currency.

Also, after studying the endogenous factors affecting Bitcoin price, a pivot will be made by using the findings of the study towards researching the endogenous factors that affect stock prices. Most shares and stocks related studies focus on the effects of exogenous features on stock price predictions. However, similar to the case of Bitcoin, a holistic study to develop a model that takes into account both endogenous and exogenous factors that affect stock prices will serve a better and a wider purpose in the world of financial assets.

In the technological domain, this work will be extended to study the performance of some more predictive models (not necessarily regression models but non-parametric) such as Support Vector Regression and Neural Networks. Neural Network based models such as LSTM are perform very well on time-series data.

Bibliography

[1] S. Nakamoto, “Bitcoin: A peer-to-peer electronic cash system", 2008.

[2] Chohan, Usman W., The Double Spending Problem and Cryptocurrencies (January 6, 2021).

[3] Z. Zheng, S. Xie, H. Dai, X. Chen and H. Wang, "An Overview of Blockchain Technology: Architecture, Consensus, and Future Trends," 2017 IEEE International Congress on Big Data (BigData Congress), 2017, pp. 557-564, doi: 10.1109/BigDataCongress.2017.85.

[4] C. Gupta and A. Mahajan, "Evaluation of Proof-of-Work Consensus Algorithm for Blockchain Networks," 2020 11th International Conference on Computing, Communication and Networking Technologies (ICCCNT), 2020, pp. 1-7, doi: 10.1109/ICCCNT49239.2020.9225676.

[5] J. Liang, L. Li, W. Chen and D. Zeng, "Towards an Understanding of Cryptocurrency: A Comparative Analysis of Cryptocurrency, Foreign Exchange, and Stock," 2019 IEEE International Conference on Intelligence and Security Informatics (ISI), 2019, pp. 137-139, doi: 10.1109/ISI.2019.8823373.

[6] Lewis, A. (2018). The Basics of Bitcoins and Blockchains: An Introduction to Cryptocurrencies and the Technology that Powers Them. [online] Google Books. Mango Media Inc. Available at: https://books.google.co.in/books?hl=en&lr=&id=5pUREAAAQBAJ&oi=fnd&pg=PT11&dq=papers+on+cryptocurrency+basics&ots=FsFIv8Ol8s&sig=-SZypHOqx5xrOZfv2VXIgK-CBcU#v=onepage&q=papers%20on%20cryptocurrency%20basics&f=false [Accessed 10 Aug. 2021].

[7] P. K. Kaushal, A. Bagga and R. Sobti, "Evolution of bitcoin and security risk in bitcoin wallets," 2017 International Conference on Computer, Communications and Electronics (Comptelix), 2017, pp. 172-177, doi: 10.1109/COMPTELIX.2017.8003959.

[8] Ciaian, P., Rajcaniova, M. and Kancs (2016). The digital agenda of virtual currencies: Can BitCoin become a global currency? Information Systems and e-Business Management, [online] 14(4), pp.883–919. Available at: https://link.springer.com/content/pdf/10.1007%2Fs10257-016-0304-0.pdf.

[9] J. Bonneau, A. Miller, J. Clark, A. Narayanan, J. A. Kroll and E. W. Felten, "SoK: Research Perspectives and Challenges for Bitcoin and Cryptocurrencies," 2015 IEEE Symposium on Security and Privacy, 2015, pp. 104-121, doi: 10.1109/SP.2015.14.

[10] J. Liang, L. Li, W. Chen and D. Zeng, "Towards an Understanding of Cryptocurrency: A Comparative Analysis of Cryptocurrency, Foreign Exchange, and Stock," 2019 IEEE International

Conference on Intelligence and Security Informatics (ISI), 2019, pp. 137-139, doi: 10.1109/ISI.2019.8823373.

[11] S. Velankar, S. Valecha and S. Maji, "Bitcoin price prediction using machine learning," 2018 20th International Conference on Advanced Communication Technology (ICACT), 2018, pp. 1-1, doi: 10.23919/ICACT.2018.8323675.

[12] K. Rathan, S. V. Sai and T. S. Manikanta, "Crypto-Currency price prediction using Decision Tree and Regression techniques," 2019 3rd International Conference on Trends in Electronics and Informatics (ICOEI), 2019, pp. 190-194, doi: 10.1109/ICOEI.2019.8862585.

[13] G. L. Joshila, A. P, D. U. Nandini and G. Kalaiarasi, "Price Prediction of Bitcoin," 2021 5th International Conference on Trends in Electronics and Informatics (ICOEI), 2021, pp. 113-116, doi: 10.1109/ICOEI51242.2021.9452976.

[14] P. V. Rane and S. N. Dhage, "Systematic Erudition of Bitcoin Price Prediction using Machine Learning Techniques," 2019 5th International Conference on Advanced Computing & Communication Systems (ICACCS), 2019, pp. 594-598, doi: 10.1109/ICACCS.2019.8728424.

[15] T. Phaladisailoed and T. Numnonda, "Machine Learning Models Comparison for Bitcoin Price Prediction," 2018 10th International Conference on Information Technology and Electrical Engineering (ICITEE), 2018, pp. 506-511, doi: 10.1109/ICITEED.2018.8534911.

[16] F. Ferdiansyah, S. H. Othman, R. Zahilah Raja Md Radzi, D. Stiawan, Y. Sazaki and U. Ependi, "A LSTM-Method for Bitcoin Price Prediction: A Case Study Yahoo Finance Stock Market," 2019 International Conference on Electrical Engineering and Computer Science (ICECOS), 2019, pp. 206-210, doi: 10.1109/ICECOS47637.2019.8984499.

[17] S. Tandon, S. Tripathi, P. Saraswat and C. Dabas, "Bitcoin Price Forecasting using LSTM and 10-Fold Cross validation," 2019 International Conference on Signal Processing and Communication (ICSC), 2019, pp. 323-328, doi: 10.1109/ICSC45622.2019.8938251.

[18] S. Roy, S. Nanjiba and A. Chakrabarty, "Bitcoin Price Forecasting Using Time Series Analysis," 2018 21st International Conference of Computer and Information Technology (ICCIT), 2018, pp. 1-5, doi: 10.1109/ICCITECHN.2018.8631923.

[19] N. P, R. J. Tom, P. Gupta, A. Shanthini, V. M. John and V. Sharma, "Prediction of Bitcoin Price Using Bi-LSTM Network," 2021 International Conference on Computer Communication and Informatics (ICCCI), 2021, pp. 1-5, doi: 10.1109/ICCCI50826.2021.9402427.

[20] I. A. Hashish, F. Forni, G. Andreotti, T. Facchinetti and S. Darjani, "A Hybrid Model for Bitcoin Prices Prediction using Hidden Markov Models and Optimized LSTM Networks," 2019 24th IEEE

International Conference on Emerging Technologies and Factory Automation (ETFA), 2019, pp. 721-728, doi: 10.1109/ETFA.2019.8869094.

[21] C. Wu, C. Lu, Y. Ma and R. Lu, "A New Forecasting Framework for Bitcoin Price with LSTM," 2018 IEEE International Conference on Data Mining Workshops (ICDMW), 2018, pp. 168-175, doi: 10.1109/ICDMW.2018.00032.

[22] T. I. Adegboruwa, S. A. Adeshina and M. M. Boukar, "Time Series Analysis and prediction of bitcoin using Long Short Term Memory Neural Network," 2019 15th International Conference on Electronics, Computer and Computation (ICECCO), 2019, pp. 1-5, doi: 10.1109/ICECCO48375.2019.9043229.

[23] B. M. Pavlyshenko, "Bitcoin Price Predictive Modeling Using Expert Correction," 2019 XIth International Scientific and Practical Conference on Electronics and Information Technologies (ELIT), 2019, pp. 163-167, doi: 10.1109/ELIT.2019.8892303.

[24] W. Yao, K. Xu and Q. Li, "Exploring the Influence of News Articles on Bitcoin Price with Machine Learning," 2019 IEEE Symposium on Computers and Communications (ISCC), 2019, pp. 1-6, doi: 10.1109/ISCC47284.2019.8969596.

[25]O. Sattarov, H. S. Jeon, R. Oh and J. D. Lee, "Forecasting Bitcoin Price Fluctuation by Twitter Sentiment Analysis," 2020 International Conference on Information Science and Communications Technologies (ICISCT), 2020, pp. 1-4, doi: 10.1109/ICISCT50599.2020.9351527.

[26] D. R. Pant, P. Neupane, A. Poudel, A. K. Pokhrel and B. K. Lama, "Recurrent Neural Network Based Bitcoin Price Prediction by Twitter Sentiment Analysis," 2018 IEEE 3rd International Conference on Computing, Communication and Security (ICCCS), 2018, pp. 128-132, doi: 10.1109/CCCS.2018.8586824.

[27] G. Serafini et al., "Sentiment-Driven Price Prediction of the Bitcoin based on Statistical and Deep Learning Approaches," 2020 International Joint Conference on Neural Networks (IJCNN), 2020, pp. 1-8, doi: 10.1109/IJCNN48605.2020.9206704.

[28] A. Aggarwal, I. Gupta, N. Garg and A. Goel, "Deep Learning Approach to Determine the Impact of Socio Economic Factors on Bitcoin Price Prediction," 2019 Twelfth International Conference on Contemporary Computing (IC3), 2019, pp. 1-5, doi: 10.1109/IC3.2019.8844928.

[29] J. Luo, "Bitcoin price prediction in the time of COVID-19," 2020 Management Science Informatization and Economic Innovation Development Conference (MSIEID), 2020, pp. 243-247, doi: 10.1109/MSIEID52046.2020.00050.

[30] A. Radityo, Q. Munajat and I. Budi, "Prediction of Bitcoin exchange rate to American dollar using artificial neural network methods," 2017 International Conference on Advanced Computer Science and Information Systems (ICACSIS), 2017, pp. 433-438, doi: 10.1109/ICACSIS.2017.8355070.

[31] R. Albariqi and E. Winarko, "Prediction of Bitcoin Price Change using Neural Networks," 2020 International Conference on Smart Technology and Applications (ICoSTA), 2020, pp. 1-4, doi: 10.1109/ICoSTA48221.2020.1570610936.

[32] Anupriya and S. Garg, "Autoregressive Integrated Moving Average Model based Prediction of Bitcoin Close Price," 2018 International Conference on Smart Systems and Inventive Technology (ICSSIT), 2018, pp. 473-478, doi: 10.1109/ICSSIT.2018.8748423.

[33] Mahmoud, H.F.F. (2021). Parametric Versus Semi and Nonparametric Regression Models. International Journal of Statistics and Probability, 10(2), p.90.

[34] C. Seiler, X. Pennec and M. Reyes, "Parametric regression of 3D medical images through the exploration of non-parametric regression models," 2010 IEEE International Symposium on Biomedical Imaging: From Nano to Macro, 2010, pp. 452-455, doi: 10.1109/ISBI.2010.5490313.

[35] D. C. Dai and K. C. Wiese, "Performance prediction for RNA design using parametric and non-parametric regression models," 2009 IEEE Symposium on Computational Intelligence in Bioinformatics and Computational Biology, 2009, pp. 16-23, doi: 10.1109/CIBCB.2009.4925702.

[36] Yuhu Cheng, Xuesong Wang and Xilan Tian, "Reinforcement learning method based on semi-parametric regression model," 2010 Chinese Control and Decision Conference, 2010, pp. 11-15, doi: 10.1109/CCDC.2010.5499145.

[37] H. Weiduo and X. Chang, "Semi-parametric Regression in Dam Observation Data Processing," 2010 Second International Conference on Computer Modeling and Simulation, 2010, pp. 501-504, doi: 10.1109/ICCMS.2010.82.

[38] Non-Parametric and Semi-Parametric Methods for Longitudinal Data. (2008). [online] . Available at: https://content.sph.harvard.edu/fitzmaur/lda/C6587_C008.pdf [Accessed 23 Aug. 2021].

[39] V. Cherkassky, Yunqian Ma and Jun Tang, "Model selection for k-nearest neighbors regression using VC bounds," Proceedings of the International Joint Conference on Neural Networks, 2003., 2003, pp. 1143-1148 vol.2, doi: 10.1109/IJCNN.2003.1223852.

[40] S. Sun and R. Huang, "An adaptive k-nearest neighbor algorithm," 2010 Seventh International Conference on Fuzzy Systems and Knowledge Discovery, 2010, pp. 91-94, doi: 10.1109/FSKD.2010.5569740.

[41] M. Rizwan and D. V. Anderson, "Using k-Nearest Neighbor and Speaker Ranking for Phoneme Prediction," 2014 13th International Conference on Machine Learning and Applications, 2014, pp. 383-387, doi: 10.1109/ICMLA.2014.68.

[42] P. Dong, H. Peng, X. Cheng, Y. Xing, X. Zhou and D. Huang, "A Random Forest Regression Model for Predicting Residual Stresses and Cutting Forces Introduced by Turning IN718 Alloy," 2019 IEEE International Conference on Computation, Communication and Engineering (ICCCE), 2019, pp. 5-8, doi: 10.1109/ICCCE48422.2019.9010767.

[43] Q. Zhou, P. Zhu, Z. Huang and Q. Zhao, "Pest Bird Density Forecast of Transmission Lines by Random Forest Regression Model and Line Transect Method," 2020 7th International Conference on Information, Cybernetics, and Computational Social Systems (ICCSS), 2020, pp. 527-530, doi: 10.1109/ICCSS52145.2020.9336898.

[44] Teixeira-Pinto, J.H. & A. (n.d.). 2 Piecewise Regression and Splines | Machine Learning for Biostatistics. [online] bookdown.org. Available at: https://bookdown.org/tpinto_home/Beyond-Linearity/piecewise-regression-and-splines.html#PWR1 [Accessed 23 Aug. 2021].

[45] Li-Yuan Xu, Min Zhang, Wei Zhu and Yu-Lin He, "Comparison of geometric and arithmetic means for bandwidth selection in Nadaraya-Watson kernel regression estimator," 2013 International Conference on Machine Learning and Cybernetics, 2013, pp. 999-1004, doi: 10.1109/ICMLC.2013.6890742.

[46] M. I. Shapiai, Z. Ibrahim, M. Khalid, L. W. Jau and V. Pavlovich, "A Non-linear Function Approximation from Small Samples Based on Nadaraya-Watson Kernel Regression," 2010 2nd International Conference on Computational Intelligence, Communication Systems and Networks, 2010, pp. 28-32, doi: 10.1109/CICSyN.2010.10.

[47] Wang, J. (2021). An Intuitive Tutorial to Gaussian Processes Regression. [online] . Available at: https://arxiv.org/pdf/2009.10862.pdf [Accessed 23 Aug. 2021].

[48] brilliant.org. (n.d.). Multivariate Normal Distribution | Brilliant Math & Science Wiki. [online] Available at: https://brilliant.org/wiki/multivariate-normal-distribution/ [Accessed 24 Aug. 2021].

[49] Carl Edward Rasmussen and Williams, C.K.I. (2008). Gaussian processes for machine learning. Cambridge, Mass. Mit Press.

[50] Scikit-learn.org. (2018). 3.2.4.3.2. sklearn.ensemble.RandomForestRegressor — scikit-learn 0.20.3 documentation. [online] Available at: https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestRegressor.html [Accessed 28 Aug. 2021].

[51] Scikit-learn.org. (2019). sklearn.neighbors.KNeighborsClassifier — scikit-learn 0.22.1 documentation. [online] Available at: https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html [Accessed 28 Aug. 2021].

[52] scikit-learn.org. (n.d.). sklearn.kernel_ridge.KernelRidge — scikit-learn 0.24.2 documentation. [online] Available at: https://scikit-learn.org/stable/modules/generated/sklearn.kernel_ridge.KernelRidge.html [Accessed 28 Aug. 2021]

[53] Y. Bai, P. Tang and C. Hu, "Radiometric Normalization of Multi-Temporal Images Using Kernel Canonical Correlation Analysis with Linear, Polynomial and Gaussian Kernels," IGARSS 2018 - 2018 IEEE International Geoscience and Remote Sensing Symposium, 2018, pp. 5009-5012, doi: 10.1109/IGARSS.2018.8519069.

[54] R. Sahak, W. Mansor, Y. K. Lee, A. Zabidi and A. I. M. Yassin, "Orthogonal least square and optimized support vector machine with polynomial kernel for classifying asphyxiated infant cry," 2011 IEEE International Conference on Signal and Image Processing Applications (ICSIPA), 2011, pp. 104-108, doi: 10.1109/ICSIPA.2011.6144159.

[55] scikit-learn.org. (n.d.). sklearn.gaussian_process.kernels.ConstantKernel — scikit-learn 0.24.2 documentation. [online] Available at: https://scikit-learn.org/stable/modules/generated/sklearn.gaussian_process.kernels.ConstantKernel.html#:~:text=Constant%20kernel. [Accessed 30 Aug. 2021].

[56] scikit-learn.org. (n.d.). sklearn.gaussian_process.kernels.RBF — scikit-learn 0.24.2 documentation. [online] Available at: https://scikit-learn.org/stable/modules/generated/sklearn.gaussian_process.kernels.RBF.html?highlight=rbf%20kernel#sklearn.gaussian_process.kernels.RBF [Accessed 30 Aug. 2021].

[57] scikit-learn.org. (n.d.). sklearn.gaussian_process.kernels.WhiteKernel — scikit-learn 0.24.2 documentation. [online] Available at: https://scikit-learn.org/stable/modules/generated/sklearn.gaussian_process.kernels.WhiteKernel.html?highlight=white%20kernel#sklearn.gaussian_process.kernels.WhiteKernel.