Big Data Mining for Smart Cities: House Price Prediction using Time-Series Analysis

Syuqran Naim bin Shamsuddin

Main academic Supervisor: Prof. Christos Tjortjis, International Hellenic University

A Master Thesis submitted for the Erasmus Mundus Joint Master's Degree on Smart Cities and Communities (SMACCs)

August 2021

University of Mons, Heriot Watt University, International Hellenic University, University of the Basque Country

Acknowledgements

First and foremost, I am extremely grateful to my supervisor, Prof. Christos for his invaluable advice, continuous support, and patience during my master's study. His immense knowledge and plentiful experience have encouraged me in all the time of my academic research and daily life. I would like to thank all my peers, professors and colleagues at SMACCs for support throughout the master's program especially during this trying time of COVID-19 pandemic. Finally, I would like to express my gratitude to my parents Musalmah Abu Bakar and Shamsuddin Mustapha for without their tremendous understanding and encouragement in the past few years, it would be impossible for me to complete my study.

Abstract

This dissertation was written as a part of the MSc in Smart Cities and Communities (SMACCs) under the European Union’s Erasmus + Program. It intends to analyse the application of data mining in smart cities especially in the real estate sector. Multiple machine learning and data mining algorithms approaches are compared for evaluation against the dataset selected. The methodology was implemented using the data mining Knowledge Discovery (KDD) framework on the housing market in London, *dubbed the smartest city in the world* where 33 boroughs with distinct market prices are tested against each other. Extensive data from 1995 until 2021 are used to build an optimal time series forecast model to predict the house prices. Univariate time series analysis was shown to be better in forecasting house prices than multivariate time series.

Keywords: House Price Forecast, Multivariate Time Series, Autoregressive Moving Integrated Average (ARIMA) Models, Stationary Time Series, Vector Autoregressive (VAR) Models

Syuqran Naim Shamsuddin

August 2021

Table of Contents

List of Figures

List of Tables

1 Introduction

The world´s appetite for seamless and smart services has ever been increasing since the technology boom in the end of 20th century. Our daily experiences in cities are increasingly tactile and efficient within the smart intelligent systems: smart devices alert everyday joe of subway delays but assure them that the day's air quality is good, so he/she can make an informed decision to cycle to work, staying fit in the process. An entrepreneur applies for a business license and is pleased to find not only a simple digital form with fast approval but ample city data that helps her identify a good location for her new business. A middle-aged woman worried about her aging father living alone is reassured to learn that local healthcare providers can monitor his diabetes and video chat with him in his home. Smart devices are now the new compass to the city.

Wealth of information at our fingertips navigate us to a more efficient use of our time and resources in our day-to-day activities. Infinite information is continuously fed by layers of sensors and trackers embedded throughout our urban environment: data flows in real time, mined into analytics ecosystems to run hyper-complex city operations and infrastructure, often with minimum human interaction. As million more urban dwellers use them to make better decisions, the data accumulate back to the information flow, making the city more responsive and productive. This means less waiting time at transits and queue and more economical solutions to everyday needs. Energy, resources, space, and investment are utilized more efficiently. Ultimately, creating the ecosystem that defines a smart city.

Our growth in technology is also supported by the ever-growing urban population. However, rampant physical development exacerbated by the digital ease in doing business and social mobility is seen to have an adverse effect on how we use our limited resources in cities: land. Real estate boom in big cities has been key in the consumption of physical space, where consolidation and densifying city core has driven the migration of middle class to the city periphery: the suburbs, adding more footprint and destruction of the environment. The rise of middle class also has significantly affected the market where the newly affluent middle class begins to spend more capital into housing both from investment and accommodation.

Growing housing market also established a challenge in ensuring affordability, especially in world capital cities like London, New York, and Singapore. Analysis on current and future house prices is a critical endeavour to policymakers, investors, and individual buyers and sellers in these cities for many reasons. Firstly, housing takes a substantial aggregate portion of households wealth and expenditure. This meant that house prices possess the potential to widely affect consumption through the ability of financing-related debts. As population grows, the housing demand will only grow. Disproportionate demand growth will only exacerbate the price increase and threaten affordability. Heavy price fluctuations shocks in real estate market possess the potential to widely affect and shake the local economy if it's not properly addressed. In the wake of 2008 economic crisis, related businesses have recognized the importance of predicting the performance metric of the market: pricing, supply, demand, permits and starts. Technology in data mining and machine learning presents an interesting opportunity to address this need. Rampant adaptation of machine learning and data mining in other sectors has only motivated the real estate industry to adapt. There are various machine learning algorithms that can use big data mining potentials in addressing the real estate market. Among the most notable are deep learning, neural network, regression, classification, and time series.

In the present study, we will apply univariate time series and multivariate time series analysis to develop dynamic structures of real house prices in 33 boroughs in London over the period of January1995 to July 2018, using UKHPI monthly average prices. Both univariate (ARIMA) and multivariate (VAR) are then used for one-step ahead forecasting for housing prices for August 2018 to March 2021. The accuracy of these forecasts is examined by contrasting the predicted values with the actual house prices with root square mean error (RMSE). . Univariate time series analysis was shown to be better in forecasting house prices than multivariate time series.

All the experiments on this thesis were executed in Python 3.6 and assisted by PowerBI. In addition, the implementations of all algorithms come from Scikit-Learn and tsa.model which are machine-learning packages, on a predefined form.

This thesis is organized as follow: Chapter 2 will investigate fundamental concepts of smart cities, big data mining, current practices and how they intersect. Chapter 3 presents the smart city application of data mining in real estate. The scope and methodology of the study is then discussed in chapter 4 following a brief literature review on valuation, forecast standard practice of the real estate market.

2 Background

Smart city is not a new concept, it has been the topic of discussion since the dot-com boom, and foreground to many science-fiction literature and movies. Data Mining is a key component of smart city where big data, considered as a significant part of the ecosystem feeds and gain information in a perpetual informational machine.

2.1 Smart City

An indicative smart city definition from ISO/IEC [1] recognizes that smart and sustainable city as “an innovative city that uses information technology and other means to improve quality of life, efficiency of urban operation and services, and competitiveness, while ensuring that it meets the needs of present and future generations with respect to economic, social, and environmental aspects.” A common view for most technical urban technologist and professionals alike [2].

However, the term smart city has been interpreted differently in different context and field of study. Usage of the term “smart” as a branding tool also has been the subject to many debates. For example, despite being a world ranked smart city, with extensive technology application form wide use of surveillance systems, traffic management and highly robust open data platforms, London is still riddled with disproportionate growth, empty skyscrapers, homelessness and urban poor [3] .Where as some cities like “sponge cities” in China solve their urban water management issues with efficient but low-tech, old school, engineering and planning strategies [4]. It can be said that a conscious balance of these low-tech and high-tech approaches can be a sustainable way forward. This gap additionally, highlights the potential “smartness” aspect of city growth that can be explored and further understood.

A comprehensive report by Dameri [5] indicated that there are compulsory elements in a comprehensive a smart city definition: component, boundaries, scope and terminology. They are summarized as “a well-defined geographical area, in which high technologies such as ICT, logistic, energy production, and so on, cooperate to create benefits for citizens in terms of wellbeing, inclusion and participation, environmental quality, intelligent development; it is governed by a well-defined pool of subjects, able to state the rules and policy for the city government and development". Technology *(city management ecosystem and infrastructure)*, people *(socio-economic and socio-culture)* and institution *(policy, education, and governance)* then make up as the key stakeholders. We can safely deduce that the technologies in smart city are the key enablers, supported by a robust policy but focuses on outcomes that are beneficial for all its stakeholders.

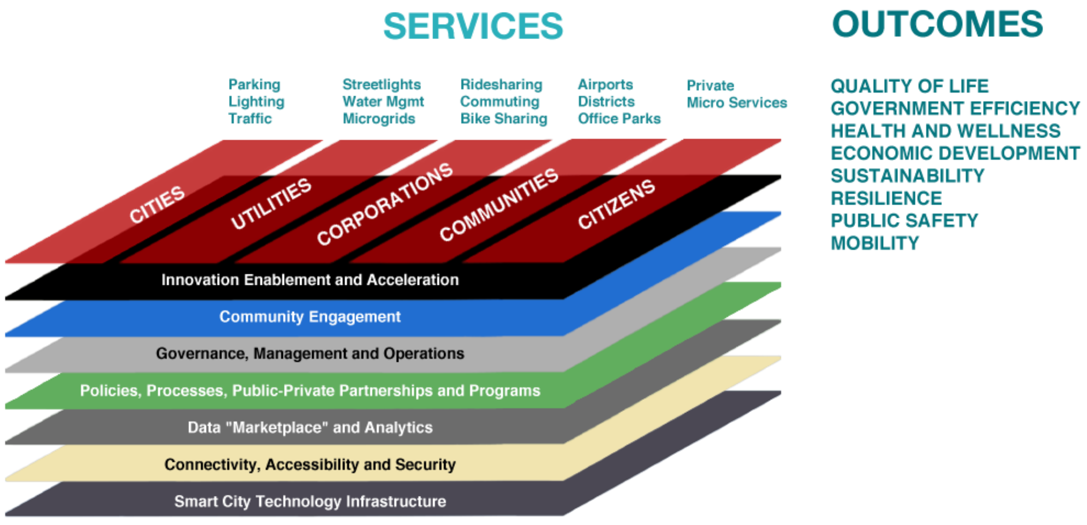


Figure 1 : Smart City Ecosystem Framework

With that in mind, we can agree that the dimensions and layers form an ecosystem that is comprised of the elements mentioned by Dameri [5]. Figure 1 shows an ecosystem of smart cities that includes all the stakeholders or as depicted in [6] as value creators: cities, utilities, corporations, communities and citizens. They are supported by "capability layers" that are: innovation layer, community engagement layer, governance and operational layer, policies and financing layer, data information and marketplace layer, connectivity, and security layer and finally the enabling technologies layer. This in the end, cumulatively work in the ecosystem to deliver the desired "goal" of improved health, quality of life, seamless, efficient government services, sustainability, and resiliency in our cities.

2.2 Key challenges in Smart Cities

Smart cities are one of the world's answers to rapid urbanization. As major cities and government embarks on the digital transformation bandwagon, several fundamental challenges hinder the adoption of the smart city ecosystem namely:

Infrastructure: There is the cost dimension to technologies like sensors and IoT in smart cities namely to install, maintain and operate them.
Reliability of Ecosystem: As reliance to the technologies increase, how sure are the system's reliability in unprecedented times? What would happen if a sub-sector of smart dashboard collapses and what are the strategies to mitigate this?
Privacy and Security: A key important factor in adoption of smart cities is the security aspect, both a real and perceptive concern. This includes issues of personal data privacy, encryption, inconsistent public interface, and national security (external threats/hacking etc.). Regulatory and policy steps area able to help address this concern. Example: General Data Protection Regulation (EU) 2016/679 (GDPR) [7].
Citizen Engagement: Without citizen engagement, the potential of these technologies and adoption of the initiative can be difficult and slow. Public participation and engagement also could reduce the negative perception of technological involvement in their daily life [8].

When these challenges are address in tandem with effective implementation plan, the smart city terminology will be a thing of the past, as growing cities inadvertently will become smart sans the "smart city" characteristics.

2.3 Smart City Infrastructure and Applications

Today, most initiatives or projects associated with smart city are used to improve and ease their daily operations. However, most of them work in silo and are not interconnected [9]. Most smart city initiative focuses only on transport and traffic managements [10] where efficiency of movement and mobility is managed without any consideration to other key sectors like healthcare managements, and resource distribution etc., resulting in loss of opportunity cost.

Future of a smart city is big data and information system that informs daily operations, predict, and mitigate urban growth and maintain the city ecosystem: through analysis of real time and historical data. This platform will provide the necessary interface for the public and the government. Smart City Geospatial Dashboard is among the popular project in smart city governance, where real-time situation and operational data are mined, visualized and analyzed for policymakers to make informed decision [11]. Another notable example, Digital Twin Cities project is a simulation platform that generates real-time virtual model of a city that track sensors and historical data, integrating these information into multi-discipline, multi-scale, and multi-probability scenarios [12]. A Hollywood portrayal of digital twin city is not uncommon, in the movie Elysium, a simulated 3D holographic cityscape of the planet, complete with real time infographics of operations was the main centerpiece of the command center.

A seamless smart city infrastructure would require minimum human intervention, but it will affect maximum human quality of life in the city. They are key sectors that the *smartness* can be applied, depicted in Table 1.

Smart city ecosystem will only mature in time, with rigorous exploration into technologies to be applied in every aspect of life. City Brain for example is an initiative lead by Ali Baba Cloud Technologies, that enable city managers and government in cognizing, transforming, and operating cities [13]. City brain enables utilization of massive data in real time, identity trends and pattern through machine learning and formulate strategic solution based on global resources and dynamics that surpass local-level awareness.

Key Sectors	Application	Projects
Transport and Traffic Management	Vehicular Congestion Frequency of usage Accidents Commuting Optimization Features Traffic data processing Signal management	City Brain [13] Digital Twin Cities [12] Smart Light Electric Vehicle Network
Population Distribution Mapping	Land Use and Urban Planning Congestion mapping Demographic distribution Population mapping Gender distribution	Super City Planning [14] Smart Real Estate [15]
Utilities Distribution Network	Resource availability Resource reliability Utilities infrastructure Utilities usage	Smart Grid [16] Smart Building [16] Smart Meter
Health Care Management	Frequency of visit Type of treatments Type of incidents Emergency response time	City Brain [13] Smart Health Seat Pleasant [17]
Disaster Mitigation and Security	Historical data Simulation study Security assessment CCTV Surveillance data	Environmental Monitoring

Table 1 : Key Sectors in Smart City and Potential Application [9]

2.4 Smart Cities in Context

Adoption around the world in pursuit of the smartest cities tittle has resulted in some impressive initiative and project in magnitude of scale and size. Investment into technology and infrastructure has seen some cities dominating the rankings year on year. These rankings are considered an effective instruments for most cities to attract investments and competitive edge to their cities [18]. This ranking is also beneficial for cities to assess their own strength and weakness acting as an external key performance indicator (KPIs) to their smart city initiatives. There are many rankings available publicly, however, its reliability as an assessment tool has also been in question. Academics and professionals has been critical of city rankings, where increased competition between cities can threaten long term development possibly due to *cutting corners* (deregulation, structural and spatial neglect and risk delicate socio-economic balance)[18]. Ranking can be a *double-edged sword*, but a holistic, context-sensitive (local vs global), well-documented and methodically advanced rankings conducted by universities or economic research institutes can be a good reference. With that, cities can position themselves in the region and focus on improving sectors that are lacking in their profile.

Cities in Motion Index (CIMI) is widely considered an exhaustive ranking of smart cities that include 101 indicators across nine (9) key dimensions: human capital, social cohesion, economy, governance, environment, mobility and transportation, urban planning, international projection, and technology; reflecting both objective and subjective data [19]. Large and populous cities like London and New York have been dominating the ranks for the 6 consecutive years, with notable presence of some smaller city-state like Singapore and Hong Kong in their top 10. Interestingly, there is a line drawn with each city’s approach where cities like Singapore, London and New York have opted for a high-tech and infrastructure intensive approach and other cities like Copenhagen, Amsterdam, and Vienna took a community-centric approach.

2.5 Big Data in Smart Cities

Big data is an essential component of a smart city. Based on Figure 1, data mining would be characterized in the Data “Marketplace and Analytics layer, where big data platforms gathered, stored, shared and communicated from interconnected Internet of Things (IoT). This layer aim to improve efficiencies across the smart city infrastructure.

2.5.1 General Concepts

Figure 2: Visualization of Keywords in Big Data-related Papers [23]

Knowledge discovery of a vast amount of data can be overwhelming for an average human to comprehend. To make sense of all the data coming from variety of sources: (geospatial data, traffic data, vehicular traffic data, crime statistics) we need big data analytics. Through analytics, city stakeholders can draw connections across distinct sources to reveal useful information. Figure 2 shows a popular keywords relating to big data obtained from the research done in [20]. This shows the multidisciplinary use of big data in the world today where word concomitants are the gauge to the various interests that enrich scientific field [21]. Its impact has been enormous in propelling efficient and robust solution to most industries, and it will continue to grow.

Big data has evolved vastly and rapidly, the nature of the discipline has allowed various definition of big data in specific industry and no universally accepted definition has existed. However, authors in [20], [22]–[24] has articulated big data in the followings 5Vs :

- ***Variety:*** This refers to complexity of big data and its ability to integrate structured and unstructured data (whether it is text, images, voice, videos, streaming data, signals, or other types of data) from multiple sources into a comprehensive resource or database.
- ***Volume:*** This relates to the sheer nature of big data that will challenge existing hardware, software in terms of storage, processing, analysis, and visualization capabilities. This also inevitable mean the exponential growth will affect the capability of big data to generate knowledge and insights.
- ***Velocity:*** This refers to the speed with which the data is generated, analyzed, and reprocessed into the platforms. Today this is mostly possible within a fraction of a second, known as real time. Velocity also raises a new concern on data ageing, in question of the data validity [25]. For example, time sensitive data like traffic and crime surveillance.
- ***Veracity (Validity):*** This refers to data quality and authenticity that ensure the credibility of the data. A bad data can be a liability and a burden to any decision based on the data analytics output.
- ***Value:*** This refers to the potential added value to the user of big data (business, government, planners etc.)

Interestingly, some researchers argue that the term “big” in big data will fade over time and “data” will be self-describing thus naturally include all the big data characteristic mentioned above [25]. The impact of big data in smart city is unmeasurable, where the success of a smart city, depends on how robust the use of big data in its infrastructure.

2.5.2 Big Data Value Chain Operators

Big data roles in the new technology age are undisputable, where its vital businesses, government and people are incorporating data architecture into their organization to stay competitive. In smart city, big data innate characteristic of 5Vs poses a challenge to the smart city framework. Therefore, an approach by author in [26] is the Knowledge Discovery in Database (KDD) model is used to streamline the big data analytics. In the model (**Error! Reference source not found.**), three operators are responsible: input, data analysis and output. The value chain operators are made up of subsystems and the information flow in KDD is described as a string of actions needed to generate value and insight from big data. The operators include these key processes:

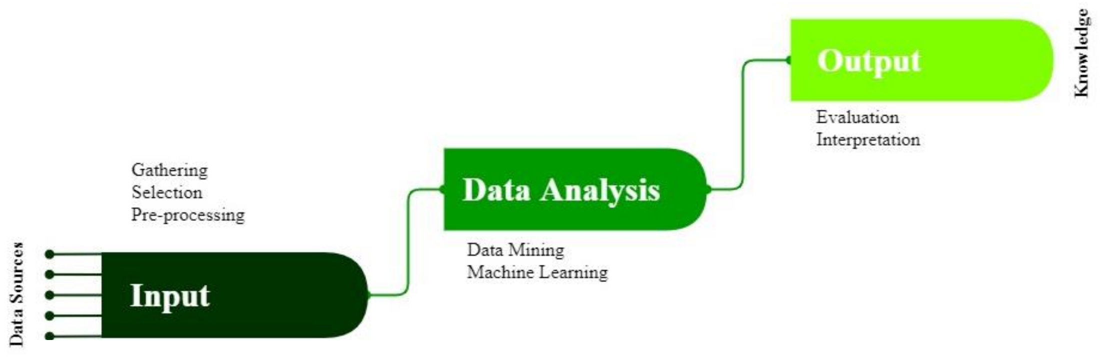


Figure 3: Knowledge Discovery in Databases model [28]

- ***Input:*** This is a critical phase where data are extracted (gathered), selected, pre-processed, and transformed to useable data that are cleaned and aligned with the 5Vs attribute. The selection operators are tasked to integrate and select relevant information from the gathered database. The preprocessing operator then, would filter, clean, and detect the inconsistent, missing, or incomplete data. Then the data would be transformed into the variety of data formats into data-mining-capable format. Data Input phase effectively reduce the complexity and scale the data that is appropriate for data analysis.
- ***Data Analysis:*** This phase is responsible for discovering the hidden patterns or rules from the datasets, most investigators in data mining field use the term to describe how they hone the “ground” (i.e. raw data) into “gold nugget” (i.e. information or knowledge)[26]. Data mining techniques and tool will be discussed further in the next sub-chapter.
- ***Output:*** This covers the data-driven business activities that need access to data, its analysis, and the tools needed to integrate the data analysis within the business activity. The mined data is evaluated for its accuracy and veracity. Finally, the data usage goes through interpretation where organization use the insight to make informed decision on strategies or measure existing performance criteria etc.

2.5.3 Platforms

To address the challenging nature of big data, platform scalability is key in ensuring a successful workflow. A big data platform can scale vertically and horizontally which is also known as scale up or out [25]. Vertical scaling implies additional computing power like RAM and CPUs while working on a single Operating System (OS). In the opposite, horizontal scaling implies a division of datasets over multiple parallel servers or *shards*.

So, more machines are added as much as needed to improve the platform’s performance. Both has pros and cons, which ultimately depends on the situation at hand. A vertical scaling is great in handling due to a single operating system, but it can be costly in comparison. While horizontal scaling provides you the ability to process your data in smaller chunks and possibly with less time but, managing multiple instances of operating system can be complex. Sample of vertical scaling are Graphics Processing Unit (GPU), High-Performance Computing Clusters (HPC), and Multicore processors, and Field Programmable Gate Arrays (FPGA) and some of the popular horizontal scaling platforms are Apache Hadoop and Map Reduce.

However, most research or projects in smart cities favors horizontal scaling due to the multi domain nature of smart city that is ever expanding thus its only sensible to rely on a horizontal scale out approach, as the smart city services or infrastructure grow.

2.6 Data Mining

Big data is a conceptual term that describe the large amounts of data whereas data mining refers to a technique to analyze data. In Figure 3, Data mining belongs to the data analysis phase of the knowledge discovery or KDD. Techniques in data mining is not as seldom used in urban analytics as Geographic Information System (GIS) but it presents an opportunity for analyst or researcher to combine these two techniques to get detailed differentiation in the urban forms.

2.6.1 Data Mining Functionalities

In a world of endless streams of data, there is a great need to transform them into something of value or knowledge. It is established that data mining is an essential step in knowledge discovery process (Figure 3). Data mining is a process of knowledge and pattern discovery from large amount of data (big data). Data can be sourced from various database, data warehouse, the web or real time data that are streamed directly into the system.

Data mining can be used to run a predictive or descriptive task aptly named to their functionalities. Predictive mining task performs induction on existing data to make prediction and descriptive mining task characterize present properties of the data in the datasets [27]. There are several data mining tasks that are used to specify the type of patterns to be retrieved from data mining tasks. The data mining tasks is as follows:

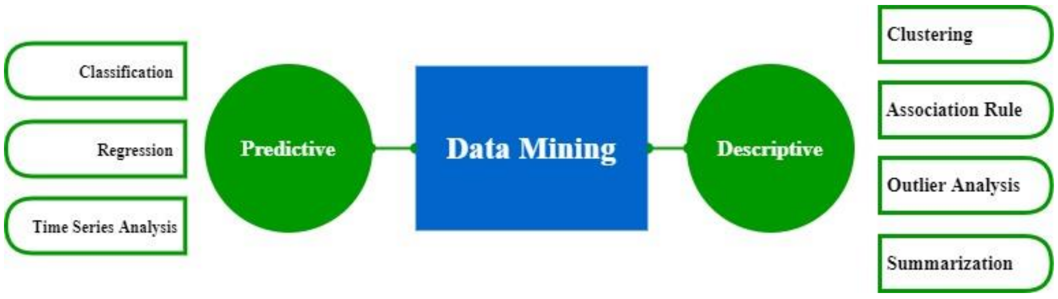


Figure 4: Types of Data Mining

- ***Classification:*** It consist of a process of acquiring a model (function) that describes or differentiates the data classes or concepts. Classification model obtained from the analysis into a *training data* (i.e., data objects for which the class labels are known). The model is then used to predict the class label of objects when it is unknown. Example of use: Categorizing applicant for a credit card in to "good", "bad" credit rating by analyzing the attributes through select techniques.
- ***Regression:*** Also used in prediction analysis, regression is used to predict a numeric or continuous value. Regression might be used to predict the cost of a product or service, given other variables. example, regression would be used to predict a home's value based on its location, square feet, price when last sold, the price of similar homes, and other factors.
- ***Cluster Analysis:*** It is a process which objects are clustered or grouped based on the principle of maximizing the intraclass similarity and minimizing the interclass similarity [27]. That is, clusters are formed so that objects within a cluster have high similarity in comparison to one another but are rather dissimilar to objects in other clusters based on distance.
- ***Association Analysis:*** It entails the discovery of association rule or frequent patterns in data. Mining frequent patterns precedes to the discovery of fascinating associations and correlations within data [27]. However, the rule can be discarded or deemed uninteresting if it does not satisfy a minimum *support* threshold and a minimum *confidence* threshold. *Support* is an indication of frequency of the itemset appearing in the dataset, where *confidence* means is how frequently is the rule is proven to be true. Example of use: Mining frequent items bought together in grocery purchases, then making

decision to shelf them next to each other.

- ***Outlier Analysis:*** It is also known as anomaly detection where datasets that may not conform to the normal or general behavior of the data. Outliers might be spotted using statistical tests that presume a distribution or probability model for the data or using distance calculations where objects that are remote from any other cluster are considered *outliers*. Example of use: It can be used to detect fraudulent usage of credit card or any other illegal activities.
- ***Summarization:*** Summarization is a data mining concept which entails methods for finding a compact description of a dataset. Simple summarization techniques such as tabulating the mean and standard deviations are often applied for exploratory data analysis (EDA***)***, data visualization and automated report generation.
- ***Time Series Analysis:*** Time series is a sequence of events where the next event is determined by one or more of the preceding events. Time series reflects the process being measured and there are certain components that affect the behavior of a process. Time series analysis includes methods to analyze time-series data to extract useful patterns, trends, rules, and statistics. Stock market prediction is an important application of time-series analysis.

A data mining system has the potential to discover millions of patterns and rules based on large amount of data that is available this day and age. With that also, raises questions about their interestingness or relevance of said patterns to the organization's benefits etc. There are measures to hone these discovered patterns into ranks, filtering the uninteresting ones, for example through the *objective measures of pattern interestingness or subjective interestingness measure*. Measures of pattern interestingness are vital for the efficient discovery of patterns by target users by pruning the patterned discovered that do not satisfy a prespecified interestingness parameters.

2.6.2 Challenges in Data Mining

Data mining as a field is still at its adolescent stage where it will be ever-growing, and ever-expanding. To continue in this growth uptrend, a few key challenges need to be addressed to further improve and enrich the field. They are articulated as follows:

- ***Security and Social Challenges:*** Despite the immense benefit of knowledge discovery in data mining, there is a real concern in breach of personal privacy and security and increasingly negative public perception to the intrusive nature of the technology [28]. The undesired discovery of patterns and access to possible sensitive data can further hinder the adoption of data mining in smart cities. However, a sub-field of study privacy preserving data mining (PPDM) is gaining traction in its development recently, as to ensure safeguards of sensitive information will still maintaining potential utility of information. This is a welcomed initiative to improve real privacy and safety concern while ensuring confidence to the public of their personal privacy and security.

- ***User Interface:*** The knowledge discovery process is only beneficial if it is noteworthy and comprehensible by the user. There is important consideration for data mining platforms allows flexible, creative and interactive interface to allow dynamic exploration of the focus of the task such as possible by-products of the initial query (i.e. unexpected pattern discoveries). The interface should also consider specific user background knowledge or previous discovered patterns.
- ***Data quality and management:*** The interoperability nature of smart city data structure means the data needs to adhere to a certain level of quality. Quality of Information (QoI) of data from multi sources of various smart city infrastructure Internet of Things (IoT) such as weather, traffic, disaster and pollution can be different and complex [28]. Of course, the data typology and quality are dependent on the functional requirement as well, thus a holistic understanding of the application and use of these data within the smart city framework can help clarify the minimum level of data standardization and quality, thus enabling data integration and aggregation for a high-level knowledge discovery (data is filtered for relevance for only high level understanding). Data often contains noise, missing value and uncertainty, thus proper preprocessing and cleaning measures is important to ensure data veracity.
- ***Complex Data***: Data can come from variety of sources and some organization use different data structures. Diverse sources generate different type of data like structured and unstructured can prove very challenging to streamline and prepped for the system. The creation of effective and efficient data mining tools for varied applications remains a difficult and dynamic area of research.
- ***Performance:*** the data mining system performance depends on the efficiency and scalability of the algorithm. A faulty or inappropriate algorithm to the specific task can affect the overall performance of the system.

3 Real Estate & Smart City

It is forecasted that smart city projects will reinforce mature cities to new heights, while emerging cities to be competitive as technology become more accessible and cheaper. Real Estate is a direct beneficiary of smart city investments, where we see these projects can quickly increase demand and prices of real estate assets tied to the investments.

3.1 Impact of Smart City in Real Estate

Most cities embarking on a smart city initiative are banking on the notion of improved efficiency, productivity, and overall quality of life for its citizens. The trickle-down ideals of job creation thorough expansion of the physical, information and communication infrastructure are the reason smart city initiative is viewed as the *next big thing*, in line with industrial revolution 4.0. With allocation of special economic zones and policies that supports smart cities, a dynamic synergy between public and private entities will pull capital and investment towards the initiative and create a robust environment for smart city implementations [29]. The development of smart city will of course benefit the real estate industry and increase demand for residential and other asset class such as hotels, retails, and offices. They are intrinsically linked. These interest and attention to the real estate aspect of smart city has given birth to a new concept of smart real estate [15]. Below are a few impacts of smart city implementation to real estate:

- Encourage public-private partnership in smart real estate projects
- Better quality of real estate product
- Ability to predict real estate trends and volatility
- Ensure a stable supply and demand of housing ensuring affordability
- Create a healthy demand for smart management and other smart features
- Inform policy maker on impact to the land use and real estate policy in cities

In mature cities most development in smart city will be mostly refurbishment and retrofit, but in new and emerging cities, smart city development are often build from scratch as an answer to rising population [30]. In this case, the impact to real estate would vary where mature cities would often have more complex variables that affect the real estate supply, price, and overall trend. In contrast, newer smart cities, housing affordability can be a controlled and planned to ensure a well distributed land use while it will be difficult to do so in mature cities. However, in mature cities, government can identify opportunity places and site for housing, amenities, public spaces opportunities using sophisticated algorithm that make use of big data. Mature cities can adopt a retrofit approach to smart real estate.

3.2 Data mining in Real Estate

Real estate growth in major cities is exacerbated mostly by a perceived demand and supply. Traditional market analysis uses a comparative market analysis (CMA) tool that is low in granularity that only include a small number of metrices. With availability of big data, a bigger metrics can be included in the process of analysing the real estate market. An efficient way to do this analysis of big data is through data mining.

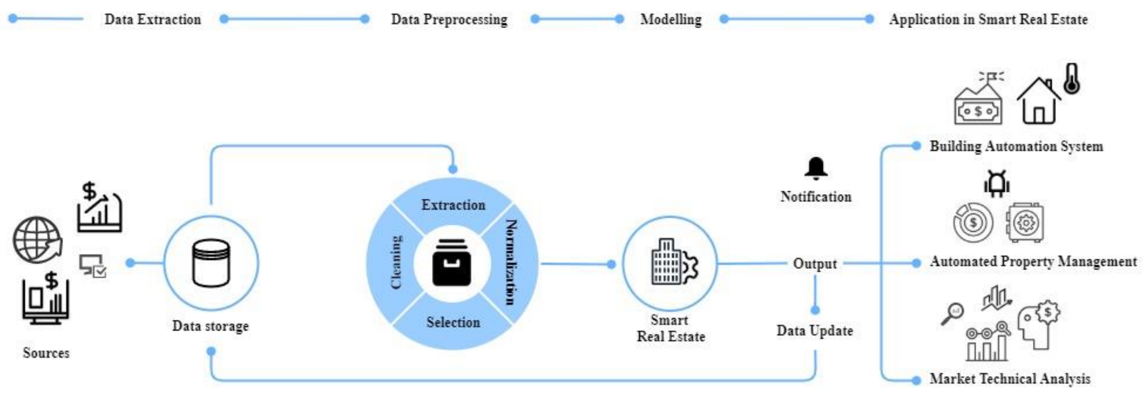


Figure 5: Data Flow in Smart Real Estate *(own source)*

There is a wide range of application of data mining or machine learning within the real estate field. Both government and private sector can benefit from the knowledge gain from data mining. For example, authorities can track building development progress, expedite permit planning process, and ensure housing policy interests are protected while encouraging a dynamic and competitive commercial real estate while private developers can identify opportunities and attractive projects to work on. These technology can be used to forecast supply, market trends, valuation and client management in commercial real estate [31]. Even other private sectors like banking can use forecast data to strategize capital requirement, possible interest revenue and losses.

There are various level of data mining and machine learning application in smart real estate. Some are used at proxy level (customer service and consumer apps) while more sophisticated application of the technology can further into analytics in building automation system (smart buildings and homes), automation of property management and analysis of the real estate market [32]. Table 2 shows the potential application of data mining and machine learning in real estate and some promising companies that are currently pursuing them.

Application Area	Description	Companies
Building Automation Systems	Analytical framework that includes Internet of Things (IoT) (i.e., sensors, camera and etc) for electrical, lighting, security, and transportation system in the building. Data collected also can support the smart city database and allow predictive analytics for other smart city services [33].	Pointgrab [34] BuildingIQ [35]
Property Management	Automation of property management job function like managing and maintaining assets in big real estate portfolio [36]. It can predict maintenance requirement of building services, rental payment, lease extensions and other concierge services like tenant reports etc.	VTS [37] AppFolio [38] Zen Place [39]
Market Analysis	Valuation process and understanding the real market analysis (macro and micro) is achieved by machine learning. It can help investors understand the market better and make informed decision with a bigger variable and with highly specific needs.	Zillow [40]

Table 2: Application of data mining and machine learning application in real estate

This paper will focus on the application of data mining and Market Analysis, where analysis of performance metrics of the industry is critical.

3.3 Real Estate Market Analysis

Real estate market analysis is usually the first step a real estate investor should make before financing a property or investment. It is also used by government to assess the health of the real estate market to ensure a stable supply and demand for housing, commercial areas, land, and government institutional services. Real estate market analysis explores the current condition of the local market – buyer's market or seller's market. It is a big part of the financial market.

To analyse the real estate market there two (2) key approaches that is fundamental analysis and technical analysis. Fundamental analysis is used for long-term forecast of values of future phenomena, based on historical data and other factors likely to affect the value of properties [41]. Technical Analysis on the other hand is used more for short-term decisions that uses data from short periods of time to develop the patterns used to predict securities or market movement, while fundamental analysis relies on information that spans years. Technical analysis on the real estate market can be used for authorities to make appropriate policy regarding real estate, while for private sectors use this analysis (price, sales volume, and social indicator of neighborhood) to discern promising or maturing neighborhoods and make business decision based on the macro factors [42].

3.3.1 Real Estate Valuation Methods

Before addressing literatures that discuss technical analysis usage in forecasting, it is essential to look at current valuation models used in the real estate industry. They are used at various level of analysis. As outlined by Jefferies in [43], the valuation model can be categorized as traditional, statistical (hedonic) and automated valuation model.

Traditional Model

Historically, valuation of real estate is done through *manual appraisal (traditional)* which utilizes the capitalization of the net income of an asset using a yield (or cap rate) that has been extrapolated from neighbouring transactions, or *comps (comparable transactions)*[44]. The *comps* will then be discretized or normalized for it to be comparable using common trend analysis, matched-pairs analysis, or simple surveys of the market. Nevertheless, these *comps* are not truly analogous as its valuation is highly skewed where comparison is only done with previously done transaction price: thus, the appraisal does not really represent a real market price. For example: in a comparative transaction analysis, only a few properties of the same criteria (location, size, and state of the local real estate market at the time of evaluation). The traditional valuation model in the past, meant that a macro level analysis of residential market is very much concentrated locally and difficult to configure accurately at state or national level.

Statistical (Hedonic) Model

The hedonic model use framework that allow individual analysis that doesn't have an observable market price [45]. The price is then determined by property attributes such as location, context, construction year, size, room count, capital improvements and other observable values. The hedonic pricing model is dependent on market prices, requiring comprehensive, available, and reliable data sets.

The model also employs *comps* (comparable transaction of properties) within the immediate real estate market over a predetermined period. The comps will then be discretized or normalized for it to be comparable using common trend analysis, matched-pairs analysis, cluster analysis or simple surveys of the market. This model is common because they apply simple regression model that are easy to understand and execute.

The hedonic model has many benefits, including appraising values on tangible selections, mainly when utilized to markets with robust and accurate data. Simultaneously, the method is flexible enough to be adapted to relationships among other external factors.

However, hedonic models are often not as accurate because they are global models, where the algorithm is constant across all attributes. Variables have different weightage and importance depending on the location and context; thus, a single model might not predict a property price in different vicinities (different neighbourhood) accurately. Newer method of property index series was introduced such as Real Capital Analytics

(RCA) where transactions are indexed so a comprehensive comparative analysis can be done, it still fails to address the problem of imprecise valuation of individual property that has varied attributes that can affect their valuation.

The statistical method is well used even today in analysing macroeconomic trends in real estate. Most of commercial real estate indexes today utilizes the statistical method for valuation etc.

Automated Valuation Model (AVM)

There are modern pricing mechanisms in real estate which includes various machine learning model. They can be classified as artificial neural networks (ANN) [46] decision trees [44], random forests, gradient boosting and support vector machines (SVM). According to [47], machine learning AVMs are used by specialist valuers for immediate and palpable advantages, including the automation of the valuation process and existence of control for the results achieved. The main disadvantage is the complexity of applying the suggested solutions, requiring a team of specialists from different fields – programmers, statisticians, mathematicians, valuers, market analysts – to develop and operate systems of this nature and scale.

However, there are certain limitation of AVM, which can be improved with a human input [48]: a hybrid method of an automated valuation model with a human edge (manual appraisal), where physical faults that cannot be spotted by a machine, such as quality of construction material, interior quality, faults and wear of the property can be quantified into the valuation. Because of that, the AVM is highly attractive as a mass appraisal method, where it can act as a first layer of valuation for real estate companies, authorities, and banks, further supported by a simplified traditional valuation model (manual appraisal) to get a greater accuracy on the price estimates. AVM can be used for both macro and micro level analysis of the property valuation model, where at macro level the valuation focuses on intra-location (district, neighbourhood, or locality) level and micro analysis would be property-specific price estimates.

3.4 Time Series Analysis for Housing Market

In this section, literature review on technical analysis using time series analysis in housing market is presented. It covers various application and studies of both multivariate and univariate time series analysis in forecasting key housing market metrics: sales, rental, demand and more. Housing market is one of the key elements in the real estate industry and a key indicator of economic health of any city. It is the keystone of the industry and of high importance to ensure a stable socio-economic and socio-spatial health. Natural disaster, political turmoil, and other unexpected events can cause shock to the system, thus, a forecast that takes this consideration is important. For example, the housing market in London is experiencing plateau in prices following Brexit in 2016 [49].

Technical analysis both in predictive and descriptive capacity plays an important role and are widely used to study the capital market. In the traditional sense, this analysis is used to determine the probability of changes in current rates based on their past changes, accounting for factors which had, have, or may have an influence on shaping the supply and demand of a given asset. In context of the study, technical analysis is a set of techniques used for assessing the value of an asset based on the analysis of the asset's trajectories as well as statistical tools [50]. The basic tools of technical analysis are groups of methods connected with 1) trends, 2) formations, read based on graphs and 3) indicators. Currently, technical analysis in real estate market is driven by automation of its processes through data mining. The potential capacity of big data mining has enabled analyst to make sense reams of massive data at a bigger scale. There are various data mining and machine learning techniques used in technical analysis in real estate, among them are regression, classification, artificial neural network, and time series.

Due to similarity in nature of stock market and housing market that is time-dependent, time series analysis focusing on univariate and multivariate model are studied for its application in the housing market. Both type of analysis are accepted as forms of forecasting model in currency markets and forecasting foreign exchange rates, behavioral study of economic variables and in meteorology [51]. However, there are a limited studies that uses time series analysis for house market pricing where it is more common for companies like Zillow and Propertyguru to do short term forecast with deep learning and regression using *comps:* data from previous house transaction [52].

In relation to that, Liow in [53] examined the relationship between property prices and properties stock prices behaviors in short and long terms. His study suggest that residential (house) prices impact the stock market in short terms while commercial properties prices in the latter. This highlight a possible similarity in behavior of prices thus same proven forecasting method for stock prices deems to be suitable for house prices.

A technical analysis with univariate time series is characterized by a single variable. It does not deal with causes or relationships. The term univariate time series refers to a singular observation(s) recorded sequentially over equal time increments. Unlike other areas of statistics, univariate time series model contains lag values of itself as independent variables. These lag variables can play the role of independent variables. An example of the univariate time series is the Autoregressive Integrated Moving Average (ARIMA) models. Zainun, Mohamed Ghazali and Salehuddin in [54] uses ARIMA to study the low-cost housing demand in Malaysia. The study aims to address the supply-demand mismatch where supply for low-cost housing in areas that has low demand are built exceedingly more than areas or state that needs them more. ARIMA was chosen due to its accuracy for short term forecasting. However, this paper fails to address the appropriate model selection technique where only three (3) parameter (p, d, q) combination were tested. Key takeaway from this literature is that the parameter tuning, or selection could be done better with other method like grid search or *auto.arima ()* in R where iteration of best combination is tested.

In a comparable study [55], McGough and Tsolacos applied short-term forecasting techniques ARIMA to predict retail rents. An ARIMA (1,2,0) model provided the best results suggesting that retail rents can be partially predicted in the short run based on movements in their past values. Their findings provide a greater comprehension of the short-term dynamics of commercial rental values and the forecasting of turning points.

Multivariate data analysis is usually employed to consider several variables in a model but will highlight the impact of the extraneous variables (stochastic terms). This model is an extension of the univariate case and involves two or more input variables. It does not limit itself to its past information but also incorporate the past of other variables. In a simpler sense, several correlated time series are observed simultaneously over time, instead of observing a single series as in univariate case. A popular multivariate time series model is Vector Autoregressive (VAR) model. In [56], author Yelmez and Kestel used technical analysis to predict house prices and employed VAR time series analysis and other non-parametric model to find appropriate fits to forecast Turkey's house price index (HPI). Their study revealed that their lags 1. The variables like GOLD, USD, EURO, INF which interact with also their first order lags (l1) to HPI. However, it is found that the correlation and explanatory power of the other variables is relatively small, and they concluded that VAR (1) has the potential to be used as an identifier for house price drivers, but they were not the best model for forecasting purposes. Brown, Song and McGillivray in [57] also used VAR as a comparative model with Time Varying Coefficient (TVC). Several VAR and time series model specifications are investigated. The VAR model using four V (4) lags and the AR (8) are found to be the best in terms of the Akaike Information Criterion (AIC), the Schwarz Criterion (SC) and the forecasts obtained. The VAR model suggest that it is found to forecast the short term with lower error, but TVC seems to forecast better in longer term.

Both univariate and multivariate time series models are designed for forecasting purposes. However, its efficiency needs to be tested for applicability as each datasets has might have a different story. This will be investigated further in the next section.

4 Methodology

4.1 Problem Statement

House prices is one of the most important elements in the housing market analysis. The price of houses can be affected by many factors and differs in market scale. London was selected as a study area based on the following reasons: London is the world capital city and have been consistently ranked as a top smart city [17]. Greater London Authority (GLA) has a robust blueprint and smart city plan [55] and London has a robust open data platform in London Data Store [54]. London consists of 33 boroughs with distinct socio-economic profile and house prices. Figure 7 depicts the current boundary of the boroughs. It is well documented that several boroughs in London is notorious for its housing prices. Most common factors in driving house prices are:

- Supply and Demand
- Macroeconomic Driver
- Location, Size and Typology
- Population Growth
- Socioeconomic Profile

This work focuses on the *supply and demand* and the *socioeconomic profile*: where data on sales volume of houses and crime rates in London is widely available. The aim of this study is to utilize data mining technology in understanding the real estate market. Our main strategy is to develop a data mining / machine learning techniques to forecast price of housing in London. This forecast model can be used to identify key opportunities areas for real estate investment, regeneration policy and housing strategies. This would help policymakers make long term strategies, private citizens making financial plans for personal real estate investment and private sector to use the model to make decision on capital investment and business strategies.

Data was extracted from the open data platform, parsed, and transformed from various sources into a readable format using several algorithm and techniques for data pre-processing, mining and then data visualization. For this project, three main tasks are aligned to achieve the desired output: 1) **Data Gathering and Selection, 2) Data Pre-processing and Transformation and 3) Model Selection for Prediction.**

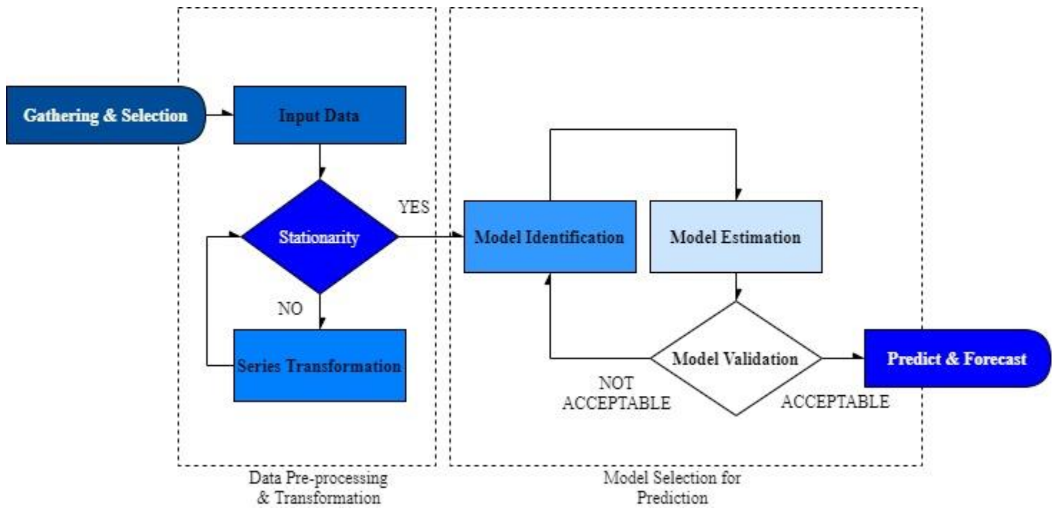


Figure 6: General Methodology

4.2 Data Gathering and Selection

4.2.1 Dataset Description

There were multiple sources of data that was extracted. The datasets have been extracted from London Data Store [58] originating from various United Kingdom government agencies. It is released under UK Open Government License v2 and v3.

London Borough Geographic Boundaries: This is a geopackage data of the Official Ordnance Survey polygons from https://data.london.gov.uk/dataset/london_boroughs showing Borough boundaries and reference code to link to national statistics in 2018. This data will be used for visualization of London's borough in the data analysis.

House Price Index: This dataset was extracted from https://data.london.gov.uk/dataset/uk-house-price-index. This was a webpage in London Datastore that houses the monthly updates on United Kingdom House Price Index (UK

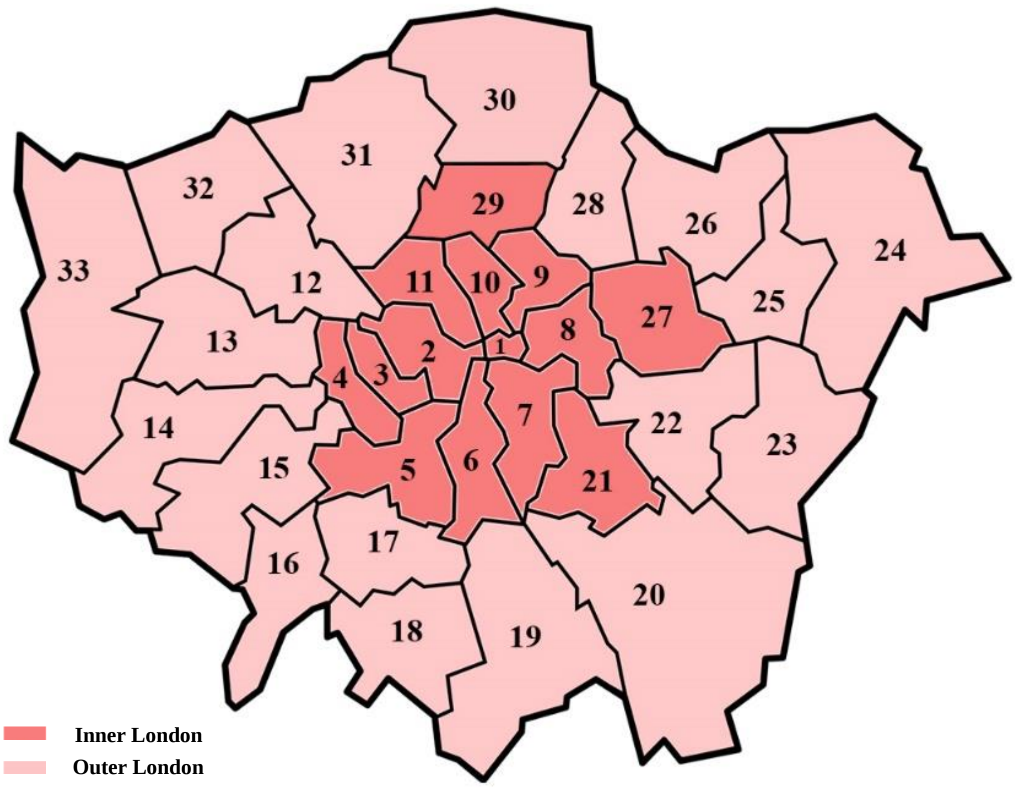
HPI). The four (4) different datasets in the xlsx file which is 1) monthly average prices by type (detached, semidetached, flat, and terraced), 2) monthly average prices, 3) monthly index prices and 4) monthly sales volume of houses specified in 33 boroughs in Greater London Area from the year 1995 until 2021 (recent month of march). There is also a summarized regional data for London (inner-outer London, east-west-north-south London and overall number for England and other region). The UK HPI applies a hedonic regression model that utilizes the various sources of data on property price (for example the Price Paid [59] dataset) and attributes to produce up-to-date estimates of the change in house prices. It is important to note that the UK HPI uses geometric means which reduces the weighting (influence) of high-value residential properties when compared to the arithmetic mean and hence is almost always lower, except when all prices are equal. The geometric mean is usually closer to the median than the arithmetic mean [60].

Crime Data: This data was also extracted from London Datastore originated from the London Metropolitan Police database https://data.london.gov.uk/dataset/recorded_crime_summary. This data counts the number of crimes at three different geographic levels of London (borough and ward) per month, according to crime type. From the year of 2000 to 2021.

For the study, we transformed and combined the data into a single csv file. The data was parsed using python 3.6 with Jupyter Notebook. This dataset is a monthly variable of each housing market in all 33 boroughs from the year January 1995 to January 2020. The dataset initially contained 13549 instances and 7 variables as shown in Table 3.

Variable	Type	Description
date	Timestamp	Time period of the record
area	Varchar	Name of the borough
average_price	float	Mean house price
code	Number	Area code according to Authority
houses_sold	Float	Number of houses sold
no_of_crime	Float	Number of crimes committed
borough_flag	Number	Indication of borough in London

Table 3: Dataset Detail Description



1.	City of London	12.	Brent	17.	Bexley
2.	Westminster	13.	Ealing	18.	Havering
3.	Kensington and Chelsea	14.	Hounslow	19.	Barking and Dagenham
4.	Hammersmith and Fulham	15.	Richmond upon Thames	20.	Redbridge
5.	Wandsworth	16.	Kingston upon Thames	21.	Newham
6.	Lambeth	17.	Merton	22.	Waltham Forest
7.	Southwark	18.	Sutton	23.	Haringey
8.	Tower Hamlets	19.	Croydon	24.	Enfield
9.	Hackney	20.	Bromley	25.	Barnet
10.	Islington	21.	Lewisham	26.	Harrow
11.	Camden	22.	Greenwich	27.	Hillingdon

Figure 7 : London Borough Boundary 2020

Additional data instances were added halfway through the project which includes house prices, sales, and crime numbers from February 2020 until March 2021. The datasets now contain 14175 instances and 7 attributes. This affectively means the data for housing market during the pandemic will be reflected in this study. More on the data structure and description, a public access to the dataset is available in GitHub: https://github.com/syuqranPSYQ/SmartCitiesHousingForecast/data .

4.3 Data Preprocessing and Transformation

The data pre-processing phase entails the process of cleaning and transforming the data that is relevant to the output of the mining process. A few attributes like dates and year needs to be separated and data from yearly datasets need to be combined into the monthly datasets. Initially it is observed in Figure 8,that there are some values for *'houses_sold'* in the February and March 2021 in every borough, a significant percentage of missing values for the *'no_of_crimes'* variable, however, since the dataset instances also include 27% summarization of other geography such as (*inner London, outer London, England, and other data from outside of London*) the *'no_of_crimes'* attribute is still valuable for observation.

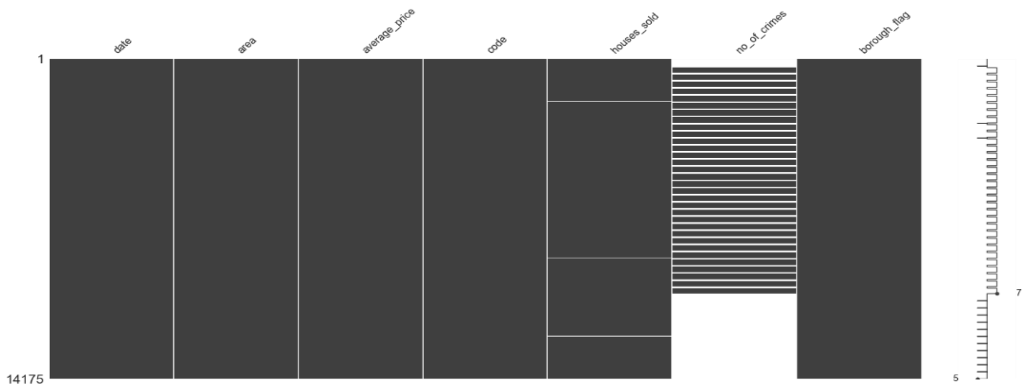


Figure 8 : MSNO Matrix to visualize missing value

Since the dataset includes other values than the relevant values for house prices in London, data frame sections were made to proceed with data exploration into 2 segments as shown in Table 4.

Data Frame	Description	Remarks
london_mean_price	Dataset that belongs to the 33 London Borough	Using borough_flag ('1') as a separator
england_mean_price	Dataset of England average house prices	Using 'area' as an identifier

Table 4: Data frame section cuts for processing

4.2.1 Exploratory Data Analysis (EDA)

An initial Exploratory Data Analysis (EDA) was done to get a look into the valuable information of the datasets and number of missing values or if there are any unimportant attributes. The variables are also tested on how they are distributed and interact. A combination of classical data exploration in Python using Pandas and Plotly with dashboard tool like PowerBI is used in this EDA.

Visualization tools such as *Key Influencers* and *Top Segments* in PowerBI were used to highlight attributes that drives metrics and identify interesting patterns for further investigation. *Key Influencers* analyses the dataset, ranks the factors, contrasts the relative importance of these factors, and displays them as key influencers and top segments for both categorical and numeric metrics using machine learning algorithms provided by ML.NET [61]. ML.NET algorithm for *Key Influencers* run linear regression, using the same data transformations as the categorical factors, and using the SDCA regression algorithm. *Top Segments* uses ML.NET to run a decision tree, using *Fast tree* algorithms (categorical and numerical), to find interesting subgroups. The objective is to end up with a subgroup of data points that is relatively high in the metric of interest. A sample dashboard is included in Appendix for further reference.

Over the last three decades, house prices have increased regularly due to the global capital flow to London and United Kingdom in General. Figure 9 illustrates London’s vs England average house price evolution for all types of housing for the period from January 1995 to March 2021. In this period, overall average house prices have increased by 5-fold in London.

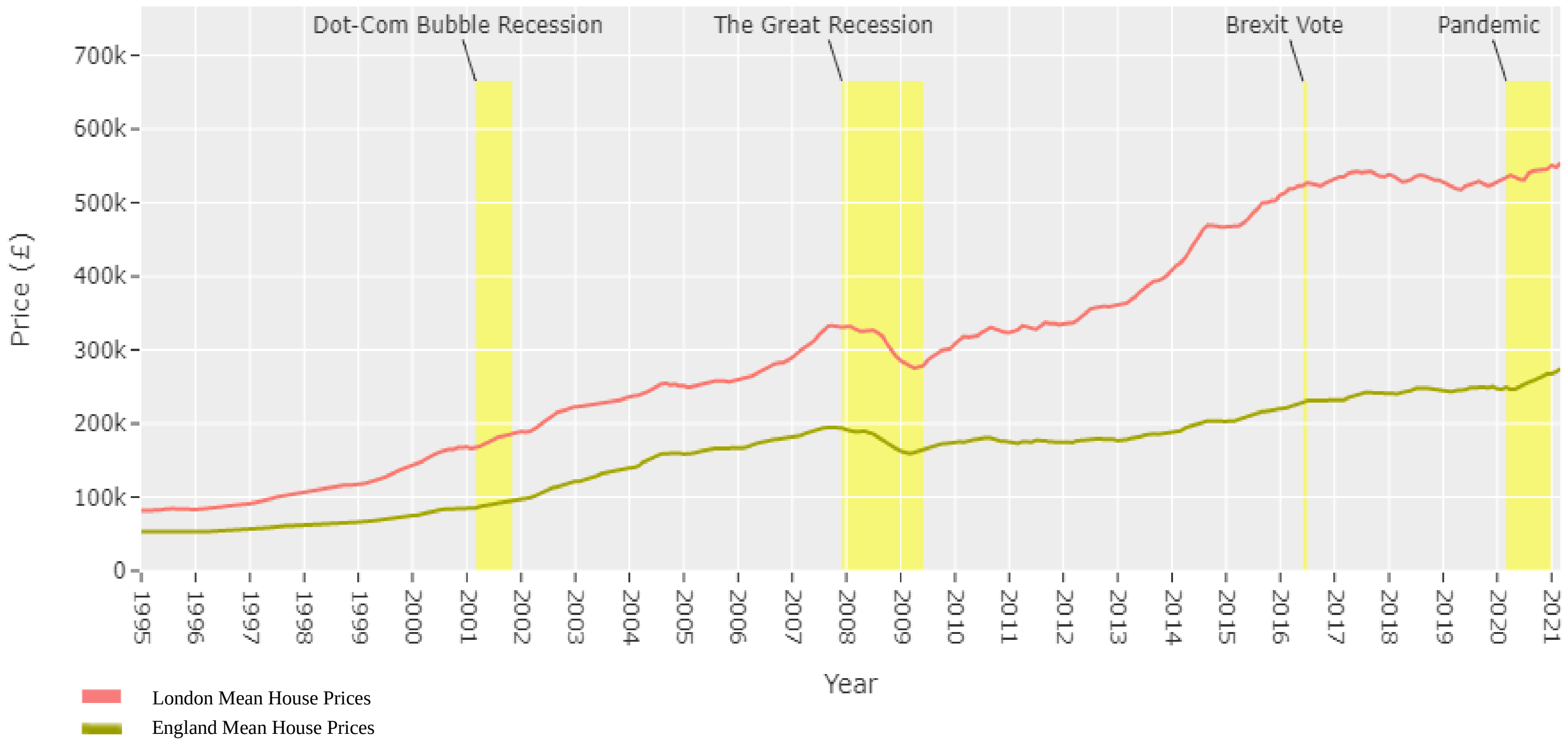


Figure 9: London vs England Average House Prices (1995 -2021)

As shown in Figure 9 and Figure 10, the recession that followed the dot-com bubble in 2000 had no material impact on house prices. This is because this recession period was mostly impacted the United State of America (USA). This upward trend, however, was disrupted by the Great Recession (2007-2009) which is caused by the US housing market collapse. London seems to have experienced a heavier price drop than England's house prices. However, London appeared to recover quickly to pre-recession level in 2011, and the average price increased steadily until 2016. England pursued a similar behaviour but with a moderate rise in the same period. Arguably, the Brexit referendum had an impact in London housing since the average price hit a plateau after 2016. It seems the rest of England was not affected as much from the landmark vote.

Upon closer inspection of average prices in London area by borough in Figure 10: Average Prices of Houses in London denoted per Borough (1995-2021)Figure 10Figure 10, expensive borough like Kensington experiences a larger drop in prices in the 2008 recession which is expected because high net purchases are deem riskier thus a slower demand for housing in expensive areas can impact prices and valuation. The slow downward price drop from Q1 to Q2 in 2018 (after shock) was because most agent and owner was in denial over the actual situation and they were holding on to pre-shock valuations[62]. Interestingly, pre pandemic prices of houses in Kensington experienced a sharp increase up until February 2020 following a 5 year 2.5% decline [63]. As expected, house prices in cheaper boroughs have a lower volatility across time with a smooth upward trend with and less disruption from external shocks as also seen in Figure 11.

The pandemic also (not surprisingly) have resulted in a slight price decline of houses in expensive boroughs (Table 5) , this can be deduced due to migration from city center to the countryside or quieter suburbs in the cheaper borough: decreasing demand. This migration also has slightly influenced the prices for the traditionally cheaper boroughs (Table 6) like Bexley and Newham to a slight increase in price, almost certainly due to increasing demand. However, the pandemic price shock didn't last long as prices rebounded back after Q3 2020 and a report from Savills confirms that the total value of property in the capital increased during the pandemic to £1.8 trillion [64]. This has prompted concern for the government as affordability for rent and buy post pandemic will be affected as emigration back to the capital is underway.

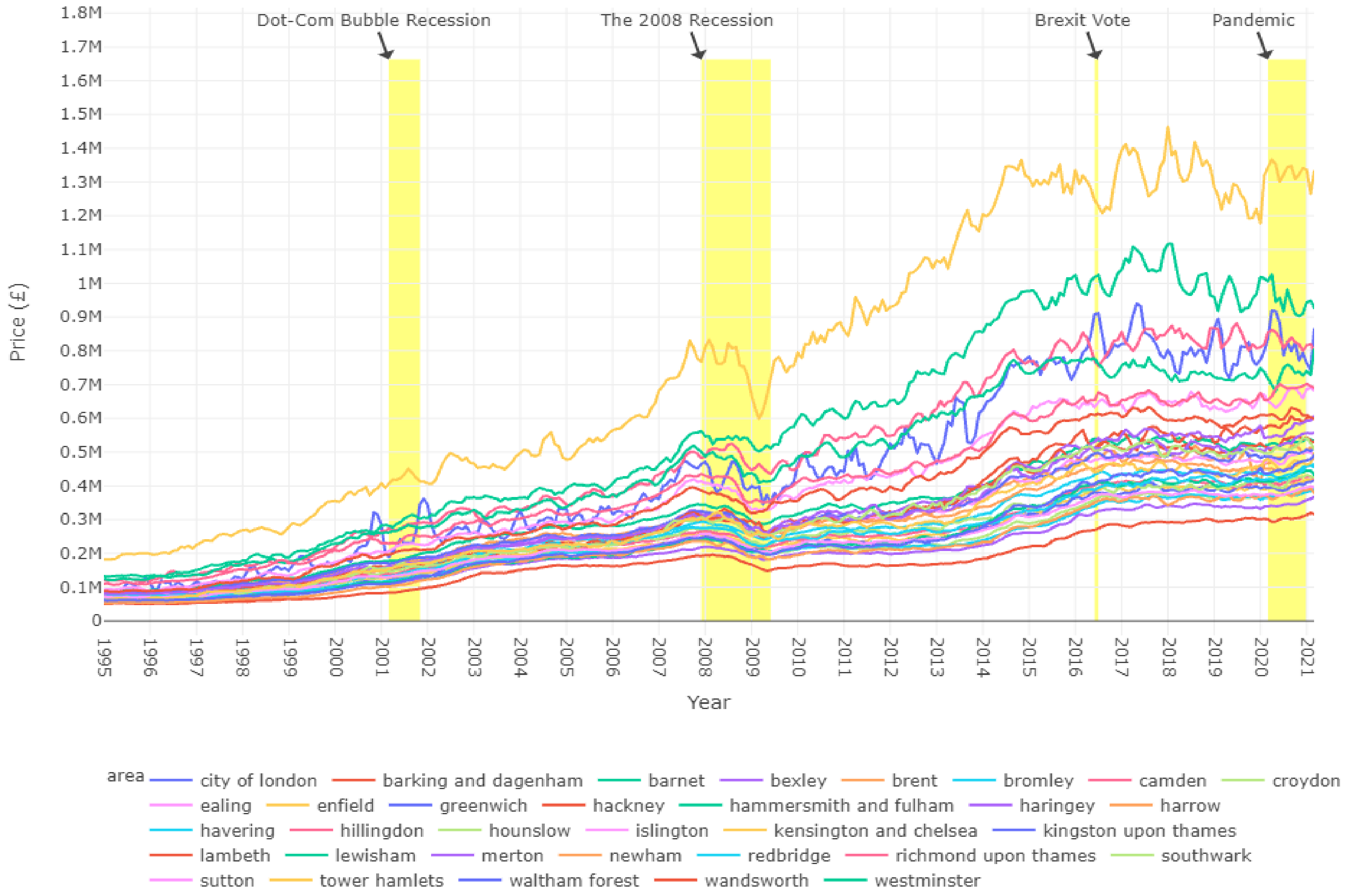


Figure 10: Average Prices of Houses in London denoted per Borough (1995-2021)

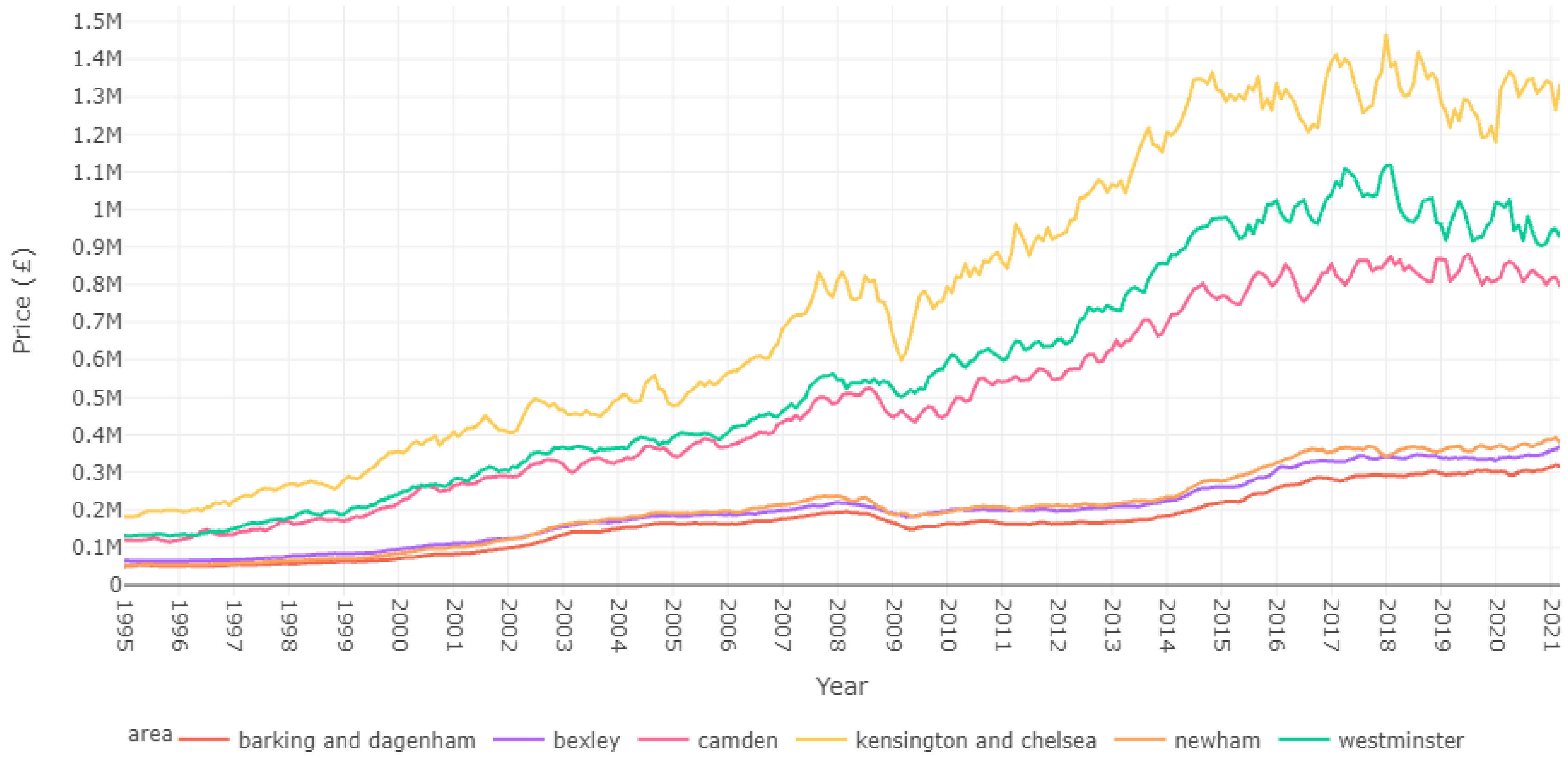


Figure 11: Average Prices of Most Expensive and Cheapest Borough (1995-2021)

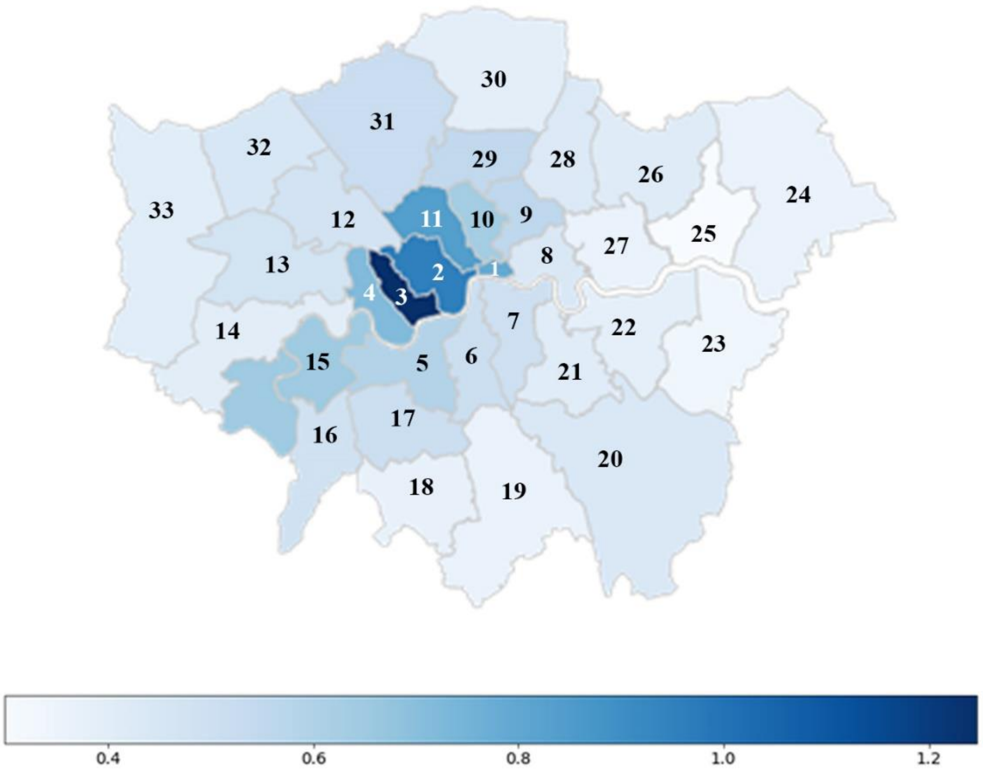


Figure 12: Visualization of Average House Prices per Borough

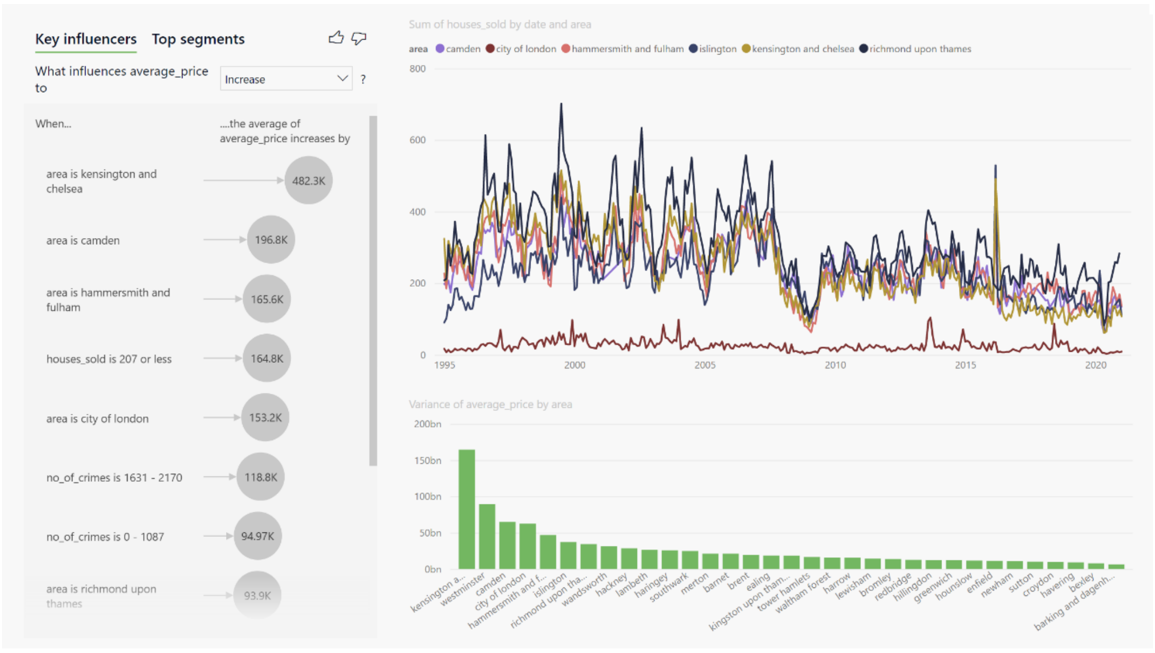
Legend	Area	Average Price	Location	
3	**Kensington and Chelsea**	768,141.96	Inner London	West NR
2	**Westminster**	562,021.43	Inner London	West NR
11	**Camden**	483,485.24	Inner London	North NR
4	**Hammersmith and Fulham**	456,724.04	Inner London	West NR
1	**City of London**	442,109.37	Inner London	Central NR
15	**Richmond upon Thames**	396,102.32	Outer London	Southwest SR
10	**Islington**	378,634.24	Inner London	North NR
5	**Wandsworth**	354,225.67	Inner London	West SR
31	**Barnet**	314,933.73	Outer London	Northwest NR
29	**Haringey**	305,212.74	Inner London	North NR

Table 5: Most Expensive Borough prices (1995-2021)

Legend	Area	Average Price	Location	
25	**Barking and Dagenham**	166,317.17	Outer London	East NR
23	**Bexley**	196,603.29	Outer London	Southeast SR
27	**Newham**	203,619.29	Inner London	East NR
24	**Havering**	212,556.68	Outer London	East NR
19	**Croydon**	216,839.75	Outer London	South SR
22	**Greenwich**	221,333.43	Outer London	Southeast SR
18	**Sutton**	224,633.68	Outer London	Southwest SR
21	**Lewisham**	225,983.46	Inner London	Southeast SR
28	**Waltham Forest**	230,929.23	Outer London	Northeast NR
30	**Enfield**	231,046.37	Outer London	North NR

Table 6: Cheapest Borough prices (1995-2021)

Key Influencer tool in PowerBI also revealed a relationship with some of the key variable in the dataset, where boroughs that were considered expensive like Kensington, Chelsea and Camden have the demand and it can be considered as ‘hot’ in the residential market as number of houses sold seems to follow the general trend of the whole of London despite the high *average_price* range than other boroughs. Other notable variables that are highlighted is *houses_sold* and *crime rate* where the increase in said variable at a specific threshold has the likeability to increase the *average_price* of said borough.



What influences average_price to: Increase

When...	....the average of average_price increases by
area is kensington and chelsea	482.3K
area is camden	196.8K
area is hammersmith and fulham	165.6K
houses_sold is 207 or less	164.8K
area is city of london	153.2K
no_of_crimes is 1631 - 2170	118.8K
no_of_crimes is 0 - 1087	94.97K
area is richmond upon thames	93.9K

Figure 13: Key Influencer Dashboard in PowerBI

For this reason, Kensington & Chelsea (moat expensive borough) is highlighted as a sample for its dynamic price history and sales for the forecasting experiment. Kensington & Chelsea has been historically an inner London wealthy and expensive area with the total dwelling number that accounts to 2.5% of London total dwellings which is 87,705 in 2019 [65]. It includes affluent areas such as Notting Hill, Kensington, South Kensington, Chelsea, and Knightsbridge. The cheapest borough of Barking & Dagenham is also sampled as a contrast sample.

Looking at the line graph in Figure 14 there seems to be a corelation as to house sales and prices as a lag can be seen from the graph, where an increase in price has a decreasing effect on the house sales year on year.

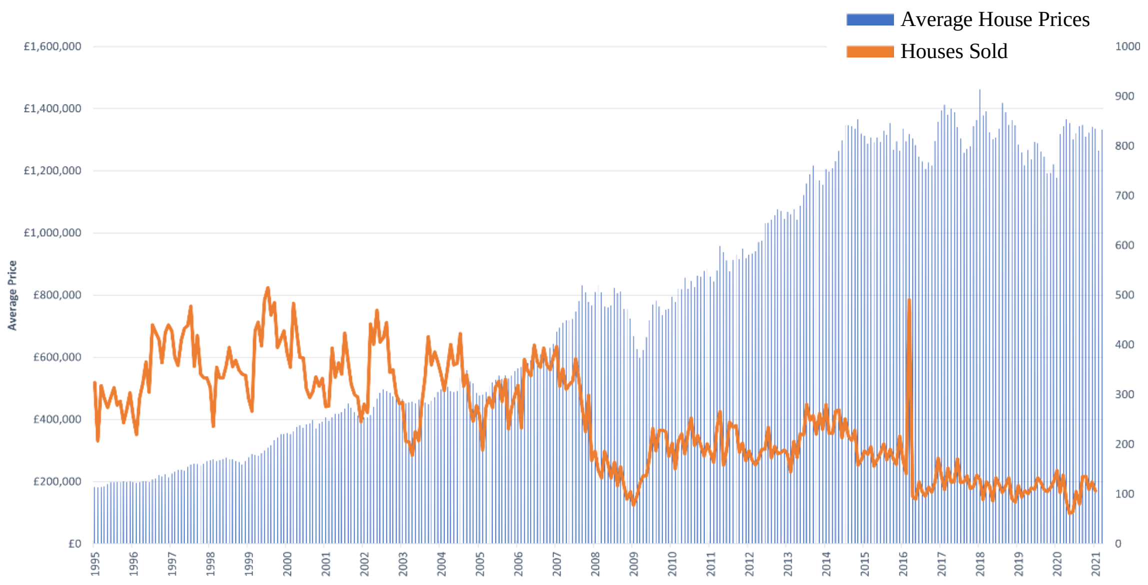


Figure 14: House Prices, Sales and Crime Counts in Kensington and Chelsea

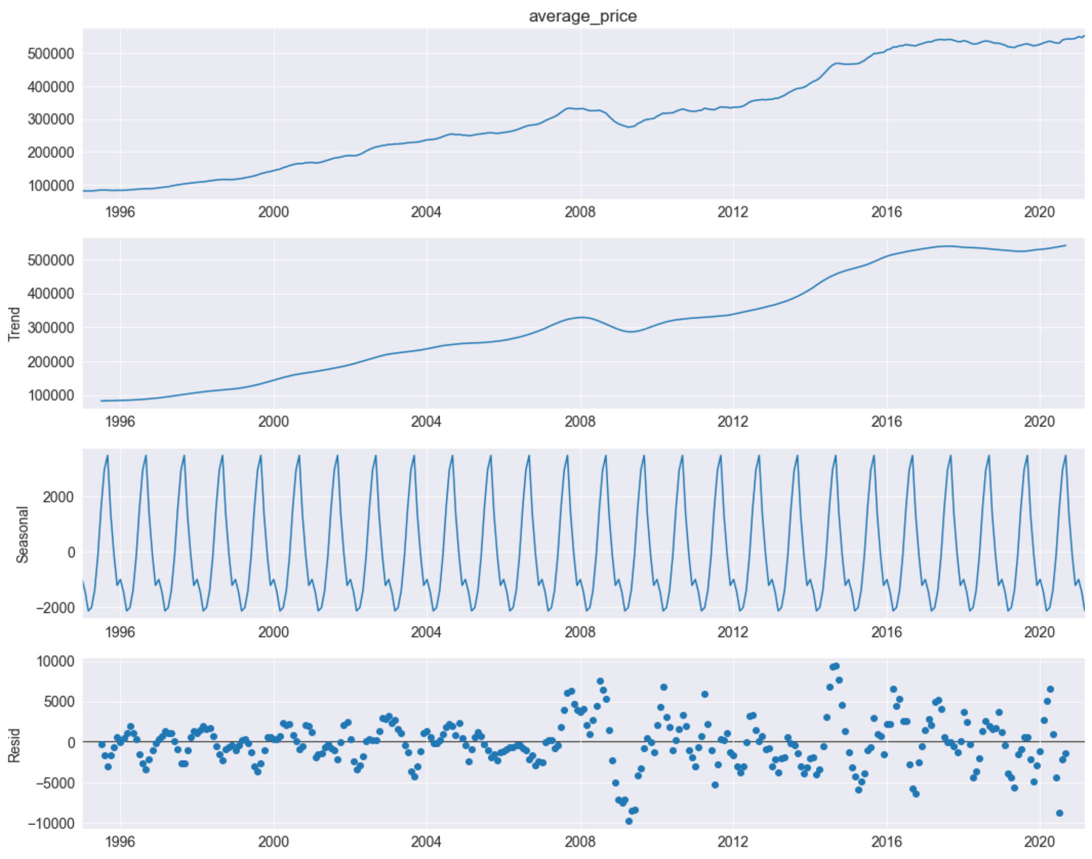


Figure 15 Time Series Decomposition (average price, trend, seasonal, noise(residual))

Before diving into the model selection and design, time series data are decomposed using the *tsa.model* package as seen in Figure 15. This is done to explore the variety of patters that the data shows. A time series can be thought of as being made up of 4 components: a) seasonal component, b) trend component, c) cyclical component, and d) noise component (residual).

We can see that the *trend* in the line time series (Figure 16) indicate a general upward motion from 1995 to 2007 with one visible drop in 2008-2009 and rebounded to upward motion until 2017 with a plateau. The seasonality information extracted from the series does seem reasonable. The residuals are also interesting, showing periods of high variability in the early and more scattered in the later years of the series.

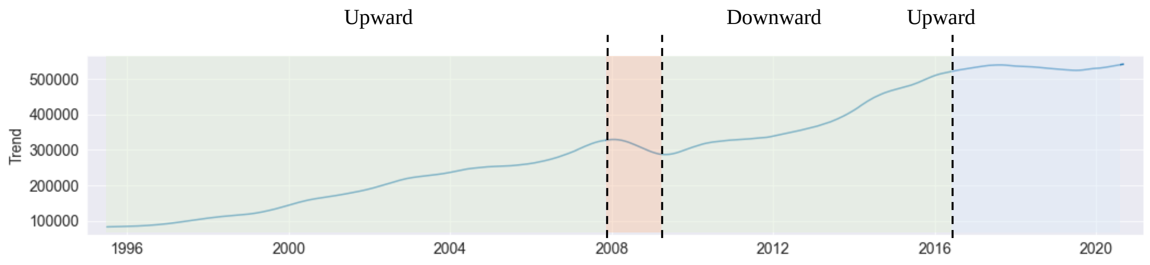


Figure 16: Trend Analysis on Average Price of Houses

4.4 Model Selection

In the early stages of the project, the author has identified the most common algorithm that work with time series analysis. A model for univariate and multivariate time series analysis is selected:

1) Autoregressive Integrated Moving Average (ARIMA)
2) Vector Autoregression (VAR)

The selected models are the extension of Autoregressive (AR) and Moving Average (MA) model where AR models anticipate series dependence on its own past value and MA models anticipate series dependence on past forecast error. The success use of the univariate ARMA (also known as Box-Jenkin approach) model for forecasting has motivated the exploration in expanding the model class to the seasonal, additional exogenous variable and multivariate use. These models are used to understand data like macroeconomic trends, weather, and stock prices behaviour. It is noted also that the selected dataset is continuous in type. thus, appropriate for time series analysis.

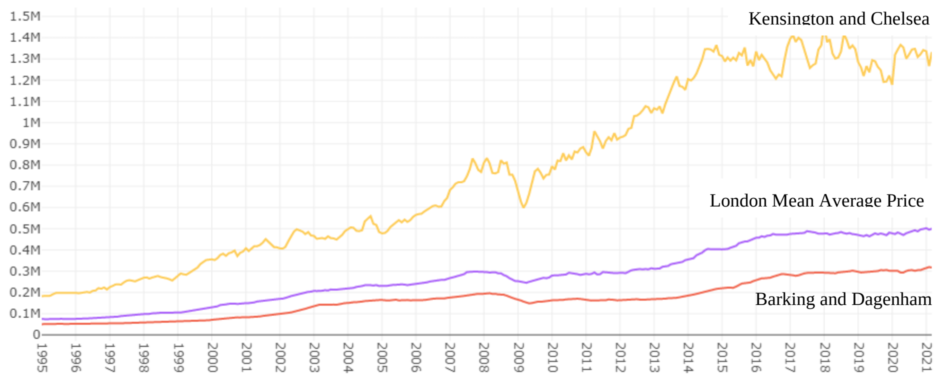


Figure 17: Average House Prices of Most Expensive and Cheapest Borough in London

For this study, two (2) borough that is the most expensive (Kensington & Chelsea) and the cheapest (Barking & Dagenham) compared to the London average houses dataset is selected for the experiment as it has the most dynamic pricing with vulnerability to market shock in the housing market. This can be a representation of the worst-case scenario and best-case scenario. Time series forecasting can generally be executed in an optimal condition in the following steps:

Step 1 : Test Harness
Step 2 : Persistence
Step 3 : Model Selection
Step 4 : Model Validation

In Step 1: The ***test harness*** is developed to investigate data and evaluate potential models. This involves two stages of a) defining a validation dataset that is separate from train-test dataset and will be used in Step 4. In this case, from 315 instances of are separated into as seen in

Figure 18 below:

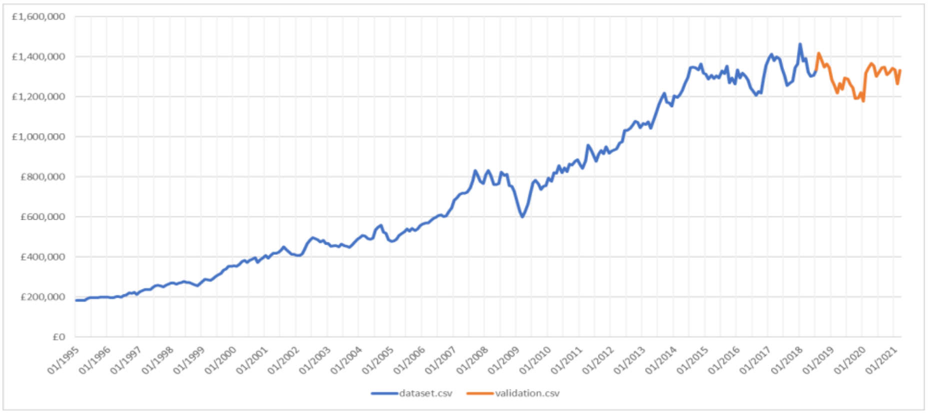


Figure 18: Sample Test Harness for Kensington & Chelsea

Dataset.csv : observations from January 1995 – July 2018 (283 instances) and

Validation.csv : observations from August 2018 – March 2021 (32 instances).

Stage b) in step 1 entails splitting data into training and testing set. In this case, 70% of the data was split for training purposes (df_train) and the remaining 30% will be used to test the forecast result (df_test). The test set is used to calculate the error rate of the final prediction method [66]. The cross-validation method also was tested with no significant difference from the other method. The test harness also includes model evaluation strategy using Root Mean Square Error (RMSE) using the helper function from the scikit-learn library mean_squared_error() that calculates the mean squared error between a list of expected values (the test set) and the list of predictions. We can then take the square root of this value to give us an RMSE score.

Step 2 **Persistence** entails a baseline model to be used for comparison where the baseline prediction for time series forecasting is called the naive forecast, or persistence. After running the initial naïve forecast the baseline RMSE is 42762.262. Step 3 is building the model. But before that, for both multivariate and univariate time series modelling, data needs to be stationary — meaning if there is a trend, it needs to be removed. To check for *stationarity*, Augmented Dickey-Fuller (ADF) test is used.

London	Column: average_price Significance Level 0.05 Test Statistic -1.0828 No. Lags Chosen 15 Critical value 1% -3.467 Critical value 5% -2.877 Critical value 10% -2.575 P-Value 0.7219 Weak evidence to reject the Null Hypothesis. Series is **Non-Stationary**.	Column: houses_sold Significance Level 0.05 Test Statistic -2.4368 No. Lags Chosen 12 Critical value 1% -3.466 Critical value 5% -2.877 Critical value 10% -2.575 P-Value 0.1316 Weak evidence to reject the Null Hypothesis. Series is **Non-Stationary**	Column: no_of_crimes Significance Level 0.05 Test Statistic -1.4805 No. Lags Chosen 13 Critical value 1% -3.466 Critical value 5% -2.877 Critical value 10% -2.575 P-Value 0.5431 Weak evidence to reject the Null Hypothesis. Series is **Non-Stationary**
Kensington and Chelsea	Column: average_price Significance Level 0.05 Test Statistic 0.2124 No. Lags Chosen 12 Critical value 1% -3.466 Critical value 5% -2.877 Critical value 10% -2.575 P-Value 0.973 Weak evidence to reject the Null Hypothesis. Series is **Non-Stationary**.	Column: houses_sold Significance Level 0.05 Test Statistic -2.0587 No. Lags Chosen 12 Critical value 1% -3.466 Critical value 5% -2.877 Critical value 10% -2.575 P-Value 0.2615 Weak evidence to reject the Null Hypothesis. Series is **Non-Stationary**	Column: no_of_crimes Significance Level 0.05 Test Statistic -1.5775 No. Lags Chosen 12 Critical value 1% -3.466 Critical value 5% -2.877 Critical value 10% -2.575 P-Value 0.4949 Weak evidence to reject the Null Hypothesis. Series is **Non-Stationary**
Barking and Dagenham	Column: average_price Significance Level 0.05 Test Statistic -1.293 No. Lags Chosen 12 Critical value 1% -3.466 Critical value 5% -2.877 Critical value 10% -2.575 P-Value 0.6324 Weak evidence to reject the Null Hypothesis. Series is **Non-Stationary**.	Column: houses_sold Significance Level 0.05 Test Statistic -2.3812 No. Lags Chosen 13 Critical value 1% -3.466 Critical value 5% -2.877 Critical value 10% -2.575 P-Value 0.1471 Weak evidence to reject the Null Hypothesis. Series is **Non-Stationary**	Column: no_of_crimes Significance Level 0.05 Test -1.4445 No. Lags Chosen 12 Critical value 1% -3.466 Critical value 5% -2.877 Critical value 10% -2.575 P-Value 0.5608 Weak evidence to reject the Null Hypothesis. Series is **Non-Stationary**

Table 7 : Augmented Dickey-Fuller (ADF) Test Result

As the data are non-stationary as shown in Table 7 the data needs to be transformed into stationary data using a simple method of *differencing*. This method will make a non-stationary time series stationary —by computing the differences between consecutive observations. Differencing can help stabilise the variance and mean of a time series by removing changes in the level of a time series, and therefore eliminating (or reducing) trend and seasonality. After *differencing* method, the data is tested for stationarity again with ADF.

London	Column: average_price Significance Level 0.05 Test Statistic -5.3773 No. Lags Chosen 15 Critical value 1% -3.467 Critical value 5% -2.878 Critical value 10% -2.575 P-Value 0.0 Rejecting Null Hypothesis.. Series is **Stationary**	Column: houses_sold Significance Level 0.05 Test Statistic -4.1159 No. Lags Chosen 15 Critical value 1% -3.467 Critical value 5% -2.878 Critical value 10% -2.575 P-Value 0.0009 Rejecting Null Hypothesis.. Series is **Stationary**	Column: no_of_crimes Significance Level 0.05 Test Statistic -3.759 No. Lags Chosen 12 Critical value 1% -3.466 Critical value 5% -2.877 Critical value 10% -2.575 P-Value 0.0034 Rejecting Null Hypothesis.. Series is **Stationary**
Kensington and Chelse	Column: average_price Significance Level 0.05 Test Statistic -4.1939 No. Lags Chosen 11 Critical value 1% -3.466 Critical value 5% -2.877 Critical value 10% -2.575 P-Value 0.0007 Rejecting Null Hypothesis.. Series is **Stationary**.	Column: houses_sold Significance Level 0.05 Test Statistic -4.2828 No. Lags Chosen 11 Critical value 1% -3.466 Critical value 5% -2.877 Critical value 10% -2.575 P-Value 0.0005 Rejecting Null Hypothesis.. Series is **Stationary**	Column: no_of_crimes Significance Level 0.05 Test Statistic -3.8108 No. Lags Chosen 11 Critical value 1% -3.466 Critical value 5% -2.877 Critical value 10% -2.575 P-Value 0.0028 Rejecting Null Hypothesis.. Series is **Stationary**
Barking & Dagenham	Column: average_price Significance Level 0.05 Test Statistic -3.4651 No. Lags Chosen 11 Critical value 1% -3.466 Critical value 5% -2.877 Critical value 10% -2.575 P-Value 0.0089 Rejecting Null Hypothesis.. Series is **Stationary**	Column: houses_sold Significance Level 0.05 Test Statistic -3.1029 No. Lags Chosen 12 Critical value 1% -3.466 Critical value 5% -2.877 Critical value 10% -2.575 P-Value 0.0263 Rejecting Null Hypothesis.. Series is **Stationary**	Column: no_of_crimes Significance Level 0.05 Test Statistic -3.8919 No. Lags Chosen 11 Critical value 1% -3.466 Critical value 5% -2.877 Critical value 10% -2.575 P-Value 0.0021 Rejecting Null Hypothesis.. Series is **Stationary**

Table 8 : ADF Result after First Differencing

London	Column: average_price Significance Level 0.05 Test Statistic -5.3773 No. Lags Chosen 15 Critical value 1% -3.467 Critical value 5% -2.878 Critical value 10% -2.575 P-Value 0.0 Rejecting Null Hypothesis.. Series is **Stationary**	Column: houses_sold Significance Level 0.05 Test Statistic -4.1159 No. Lags Chosen 15 Critical value 1% -3.467 Critical value 5% -2.878 Critical value 10% -2.575 P-Value 0.0009 Rejecting Null Hypothesis.. Series is **Stationary**	Column: no_of_crimes Significance Level 0.05 Test Statistic -3.759 No. Lags Chosen 12 Critical value 1% -3.466 Critical value 5% -2.877 Critical value 10% -2.575 P-Value 0.0034 Rejecting Null Hypothesis.. Series is **Stationary**
Kensington and Chelse	Column: average_price Significance Level 0.05 Test Statistic -4.1939 No. Lags Chosen 11 Critical value 1% -3.466 Critical value 5% -2.877 Critical value 10% -2.575 P-Value 0.0007 Rejecting Null Hypothesis.. Series is **Stationary**.	Column: houses_sold Significance Level 0.05 Test Statistic -4.2828 No. Lags Chosen 11 Critical value 1% -3.466 Critical value 5% -2.877 Critical value 10% -2.575 P-Value 0.0005 Rejecting Null Hypothesis.. Series is **Stationary**	Column: no_of_crimes Significance Level 0.05 Test Statistic -3.8108 No. Lags Chosen 11 Critical value 1% -3.466 Critical value 5% -2.877 Critical value 10% -2.575 P-Value 0.0028 Rejecting Null Hypothesis.. Series is **Stationary**

Barking & Dagenham	Column: average_price Significance Level 0.05 Test Statistic -3.4651 No. Lags Chosen 11 Critical value 1% -3.466 Critical value 5% -2.877 Critical value 10% -2.575 P-Value 0.0089 Rejecting Null Hypothesis.. Series is **Stationary**	Column: houses_sold Significance Level 0.05 Test Statistic -3.1029 No. Lags Chosen 12 Critical value 1% -3.466 Critical value 5% -2.877 Critical value 10% -2.575 P-Value 0.0263 Rejecting Null Hypothesis.. Series is **Stationary**	Column: no_of_crimes Significance Level 0.05 Test Statistic -3.8919 No. Lags Chosen 11 Critical value 1% -3.466 Critical value 5% -2.877 Critical value 10% -2.575 P-Value 0.0021 Rejecting Null Hypothesis.. Series is **Stationary**

Table 8 shows that the test statistic value -5.37, -4.19 and -3.46 for London, Kensington & Chelsea, and Barking & Dagenham are smaller than the critical value at 5% of -2.88. This suggests that we can reject the null hypothesis with a significance level of less than 5% (i.e., a low probability that the result is a statistical fluke). Rejecting the null hypothesis means that the process has no unit root, and in turn that the time series is stationary or does not have time-dependent structure. This indicates that only one level of differencing is required.

4.4.1 Auto-regressive Integrated Moving Average (ARIMA) *p,d,q*

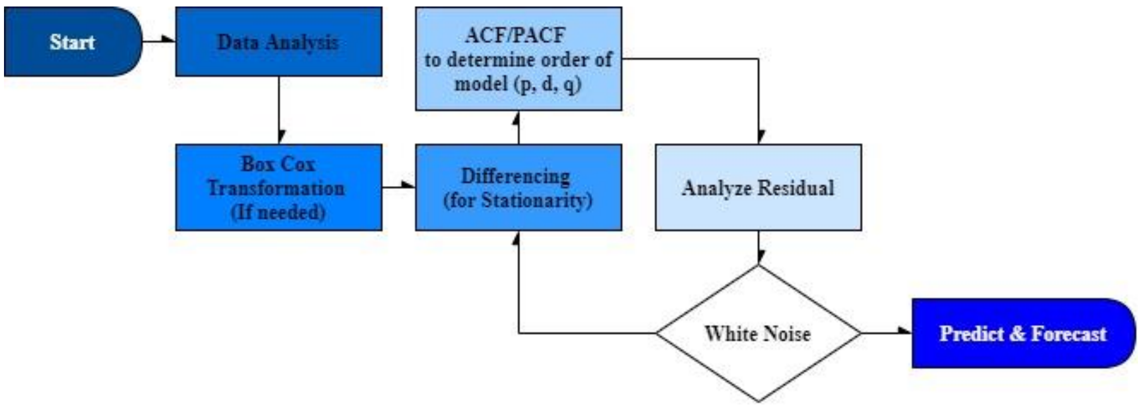

Start
Data Analysis
ACF/PACF to determine order of model (p, d, q)
Box Cox Transformation (If needed)
Differencing (for Stationarity)
Analyze Residual
White Noise
Predict & Forecast

Figure 19: ARIMA flow chart

Auto-regressive Integrated Moving Average (ARIMA) use only the past values (lag and lagged forecast error) of the time series to predict its future values. It is a univariate time series forecasting. Since the study aims to understand price trends, we shall focus only on average_price for forecasting purposes. The ARIMA model can be broken down into three different components each one with a parameter *(p, d, q)* representing the characteristics of the time series:

$$\underbrace{y'_t}_{\text{differenced time series}} = \overset{\text{intercept}}{c} + \underbrace{\varphi_1 y'_{t-1} + \dots + \varphi_p y'_{t-p}}_{\textbf{AR (p)}} + \underbrace{\theta_1 \varepsilon_{t-1} + \dots + \theta_q \varepsilon_{t-q} + \varepsilon_t}_{\textbf{MA (q)}}$$

Equation 1: Formula for ARIMA (p, d, q)

Auto-regressive AR(p): Auto-regressive models explain random processes as linear combinations, such that the output variable depends linearly on its previous values and a random variable. In our case, the AR model will make that forecast based on the previous house prices. The parameter *p* indicates the number of autoregressive terms, as in, the number of terms in your linear combination. The model can be described as:

$$y'_t = c + \varphi_1 y'_{t-1} + \dots + \varphi_p y'_{t-p} + \varepsilon_t$$

Equation 2: Formula for AR (p)

Integrated I(d): The value of *d* is determined by how many levels of differencing (step 3) is required, which in this case after running the ADF test and one level of differencing, the *d* parameter in our ARIMA model should at least be a value of 1.

Moving Average MA (q): Like auto-regressive models, in moving-average models the output variable is explained linearly, but this time is an average of the past forecast errors.

$$y'_t = c + \varepsilon_t + \theta_1 \varepsilon_{t-1} + \dots + \theta_q \varepsilon_{t-q}$$

Equation 3: Formula for MA(q)

The next step is to select the lag values for the Autoregression (AR) and Moving Average (MA) parameters, p and q respectively by reviewing Autocorrelation Function (ACF) and Partial Autocorrelation Function (PACF) plots.

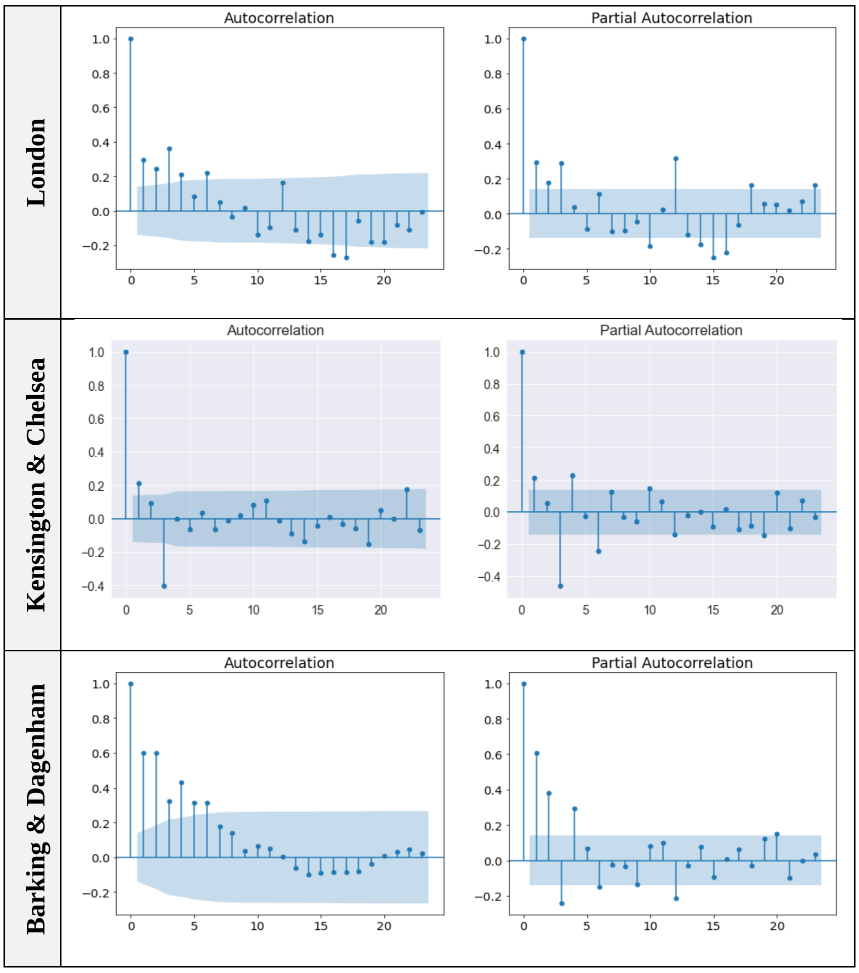


Figure 20 : ACF and PACF graph

It can be observed that the ACF for all three (3) shows a lag starting at 1 and for Kensington & Chelsea significant lags at 3,14 and 23 for while the PACF shows a significant lag at 1, 3, 4, 6, 7 and decreasing drop off. A good starting point for the p and q values could be 1 or 2 onwards. Barking & Dagenham ACF shows significant lag at 1 to 7 with a decreasing drop off onwards and PACF at 1, 2, 3, 4, 6, 9, and 12. London ACF shows significant lag at 1 to 6 with recuring lag at 16 and 17 while PACF shows significant lag from 1 to 3 and spotty reoccurrences from 12 to 25.

Grid Search ARIMA Hyperparameters

In this section, we will search values of ***p, d,*** and ***q*** for combinations that result in least error, and find the combination that results in the best performance. Grid Search [67] is used to explore all combinations in a subset of integer values. Based on significant lag in ACF/PACF in Figure 20, combination of hyper parameter (***p*** : 0 to 13) , (***d*** : 0 to 3), (***q*** : 0 to 12) which means there will be 676 runs of test harness and it took 4 hours on average to execute.

London	.... ARIMA(4, 3, 4) RMSE=4180.290 ARIMA(4, 3, 5) RMSE=4280.375 ARIMA(4, 3, 6) RMSE=4022.147 **Best ARIMA(4, 3, 6) RMSE=4022.147**
Kensington & Chelsea	.... ARIMA(4, 3, 4) RMSE=33290.296 ARIMA(4, 3, 5) RMSE=29493.111 ARIMA(4, 3, 6) RMSE=31003.782 **Best ARIMA(4, 0, 6) RMSE=28973.799**
Barking & Dagenham	.... ARIMA(4, 3, 4) RMSE=2029.948 ARIMA(4, 3, 5) RMSE=2005.498 ARIMA(4, 3, 6) RMSE=2007.320 **Best ARIMA(4, 0, 6) RMSE=1665.131**

Figure 21: Grid-Search result for best ARIMA configuration

Result

Grid Search allowed the study to discover the best ARIMA hyperparameter configuration where ARIMA (4, 0, 6) had the lowest RMSE at 28,973.8 and 1,665.1 for both Kensington & Chelsea, and Barking & Dagenham while ARIMA (4, 3, 6) was discovered as the best model with lowest RMSE of 4,022.1. London and Barking & Dagenham has noticeably better results than Kensington & Chelsea. It can be deduced that the price behaviours as seen in Figure 17 is the driving factor. Kensington has a higher magnitude in price fluctuations (higher swings) while London and Barking & Dagenham has steadier swing and price fluctuations. This can be an early indication that ARIMA might be able to predict dataset with lower fluctuations better. The selected model then was executed again then plotted to see the accuracy on a plotted graph of test dataset and predicted dataset. It seems like the model can forecast the house prices that has a good accuracy with the lowest RMSE at 28,973.8.

Residual Analysis

Ideally, the distribution of residual errors should be a Gaussian with a zero mean. Residuals will be analysed with a histogram and density plots.

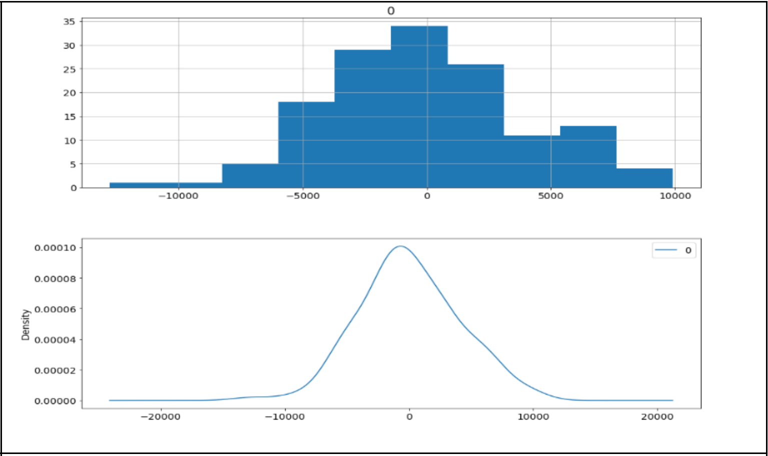
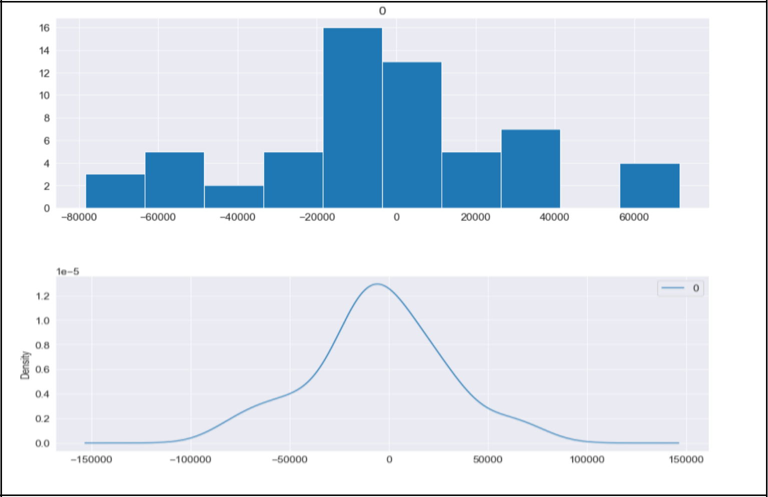
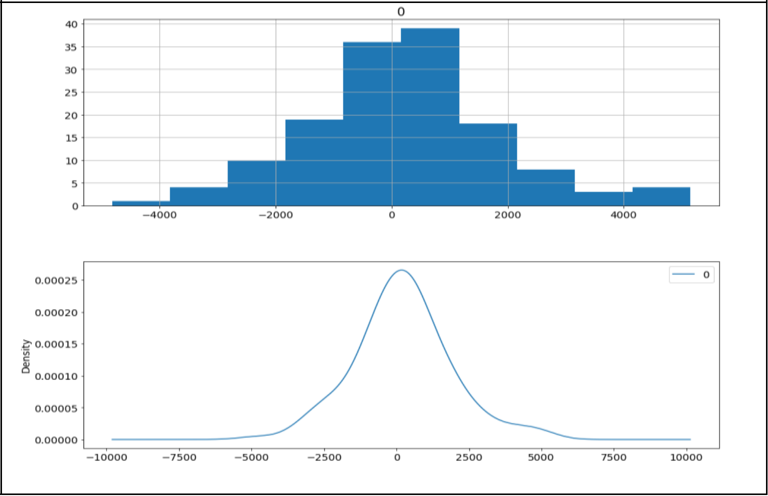
London	
Kensington & Chelsea	
Barking & Dagenham	

Figure 22: Histogram and Density Plot for Residual Analysis

The graphs in Figure 22 suggest a Gaussian-like distribution with an even tail. This is perhaps a sign that the predictions are not biased. ACF/PACF in Figure 23 also suggest what little autocorrelation is present in the time series has been captured by the model.

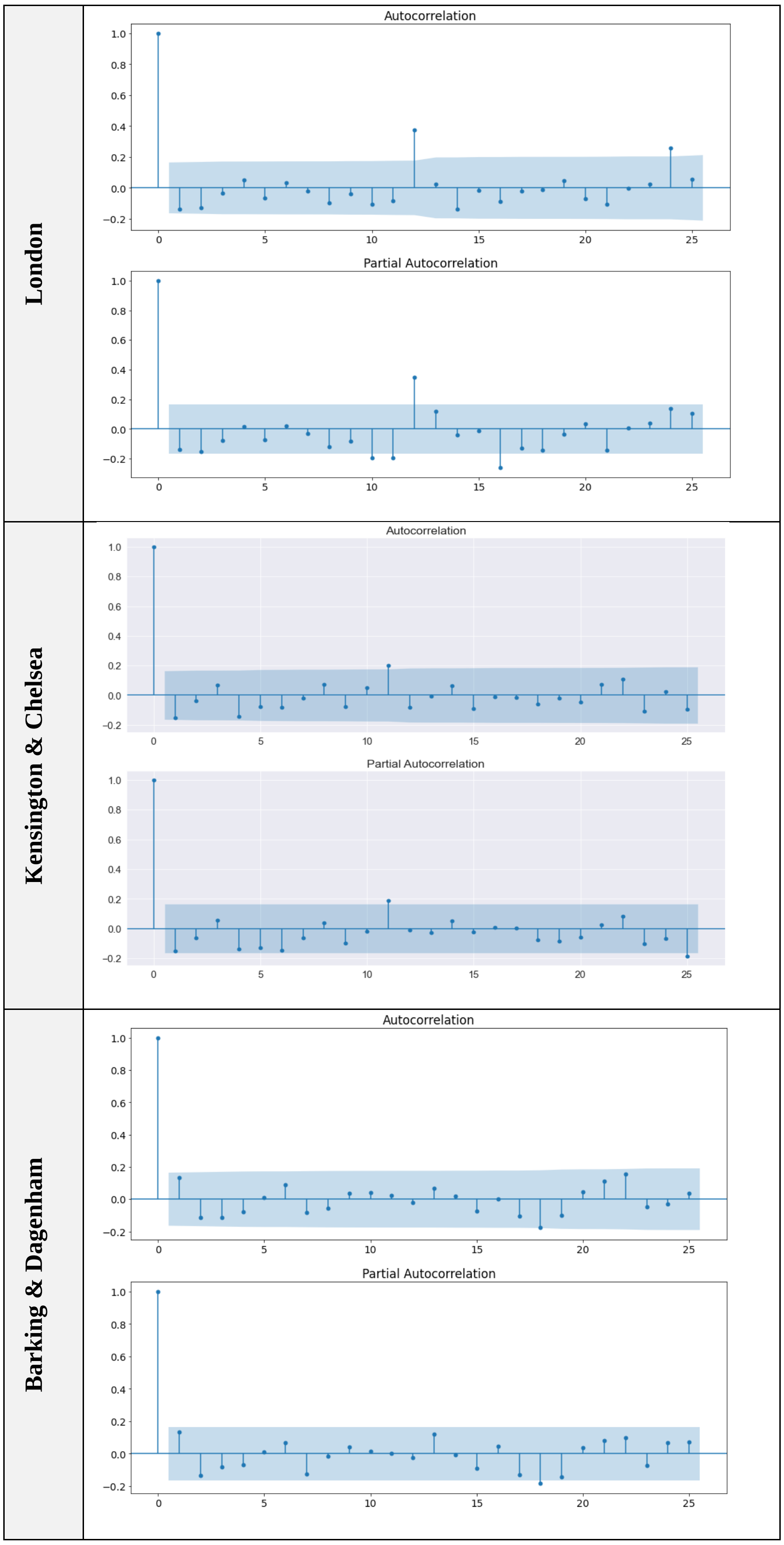


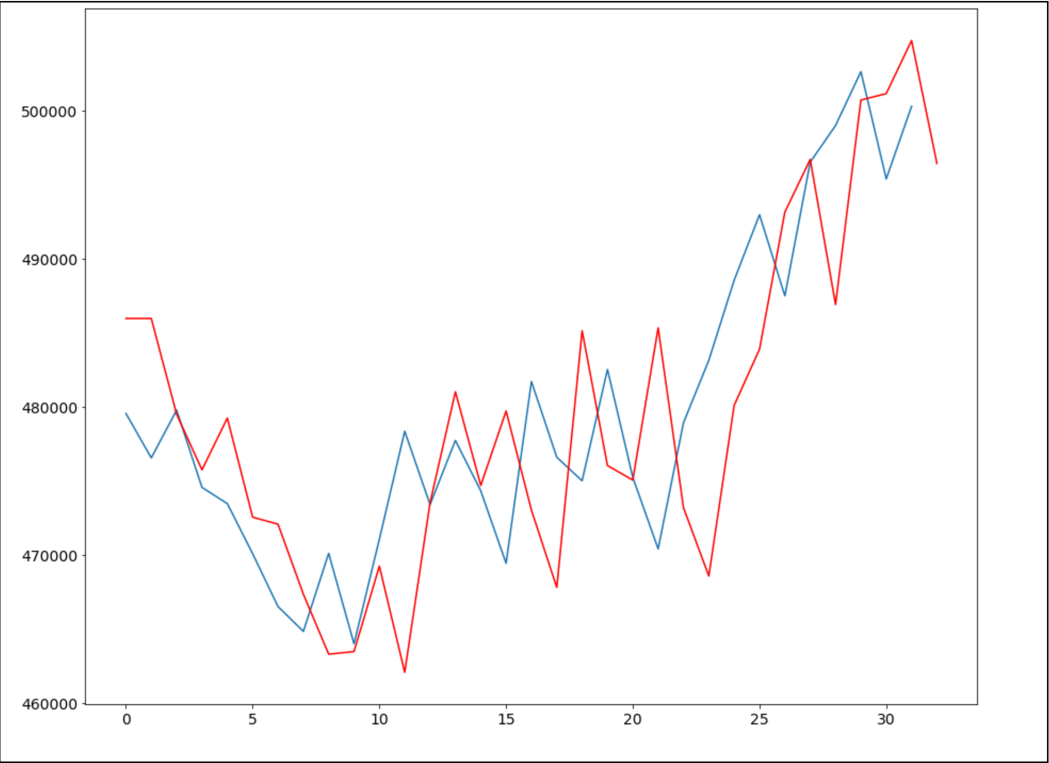
Figure 23: Residual Analysis ACF/PACF

Since the model is proven to be unbiased, no power transformation (if Box Cox function is needed) for the model to proceed to validation.

Model Validation

After models have been developed and a final model selected, it must be validated and finalized with the validation set from the test harness. This section includes the following steps:

1) **Finalize Model**: Finalizing involves fitting an ARIMA model on the entire dataset.
2) **Make Prediction**: Load the finalized model and make a prediction. In this case, the prediction result for the immediate next time step (August 2018) is £ 1,402,101.2 where the actual price is quite close at £ 1,418,032.
3) **Validate Model**: In the test harness section, we saved the final 33 months of the original dataset (August 2018 – March 2021). This model is then tested with *'unseen'* data. The result is depicted in Figure 24 where the prediction made on validation dataset which is unseen.

London	

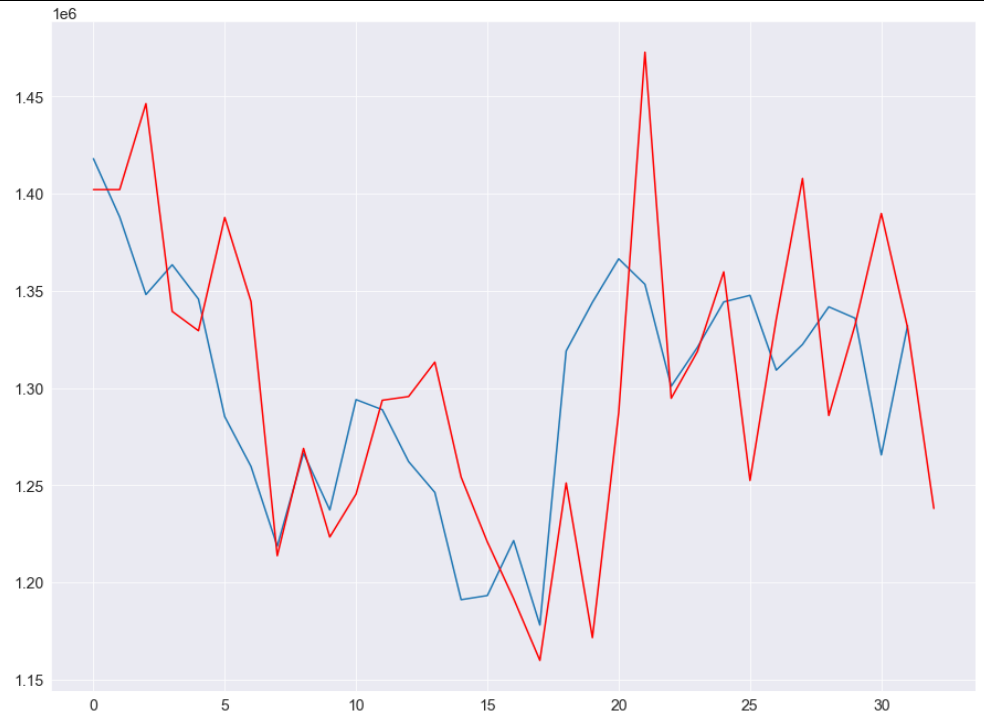
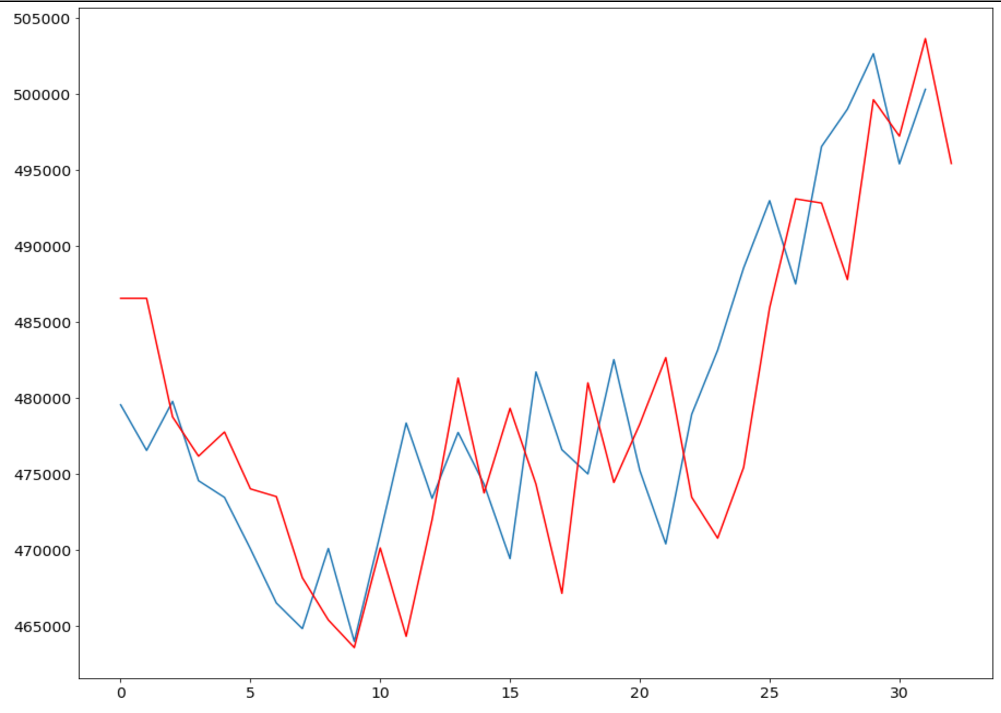
Kensington & Chelsea	
Barking & Dagenham	

Figure 24: Forecast vs Actual Data on Validation Dataset (August 2018 - March 2021)

The ARIMA model would appear to perform well up to the first one or two months will started to degrade in skill. It then manages to predict the seventh to ninth month and several following months. The overall result is acceptable and can be considered a success.

	Best ARIMA ( ) Model	**RMSE in dataset**	**RMSE in validation set**
London	4,3,6	4,022.147	6,222.666
Kensington & Chelsea	4,0,6	28,973.8	62,975.554
Barking & Dagenham	4,0,6	1,665.131	3,183.897

Table 9 : RMSE comparison for dataset vs validation set

As the table above, the RMSE for average prices of houses in London in general on the validation set is **6,222.6**, with a reasonable difference from the RMSE in the dataset at **4,002.2**. This is expected as the original data analysis has revealed that the residential market post Brexit (2016 onwards) which is within the validation set (August 2018 - March 2021) experienced a plateau (lesser fluctuation) and steady prices for the next few years. This will skew the support and forecast result.

The final RMSE for Kensington & Chelsea after this exercise on the validation set is **62,975.554,** quite far off from the result from the original dataset (January 1995 – July 2018) which is **28,973.8.** The RMSE of Barking & Dagenham almost doubled at **3,183.9** where the initial dataset achieved **1,665.1** RMSE. Both results are expected as the price behaviour are similar with the whole London price movement. Interestingly**,** the result of validation set in Kensington & Chelsea is not too different from the result in the simple persistence model (Step 2 in Model Selection).

4.4.2 Vector Auto-regressive (VAR) *p*

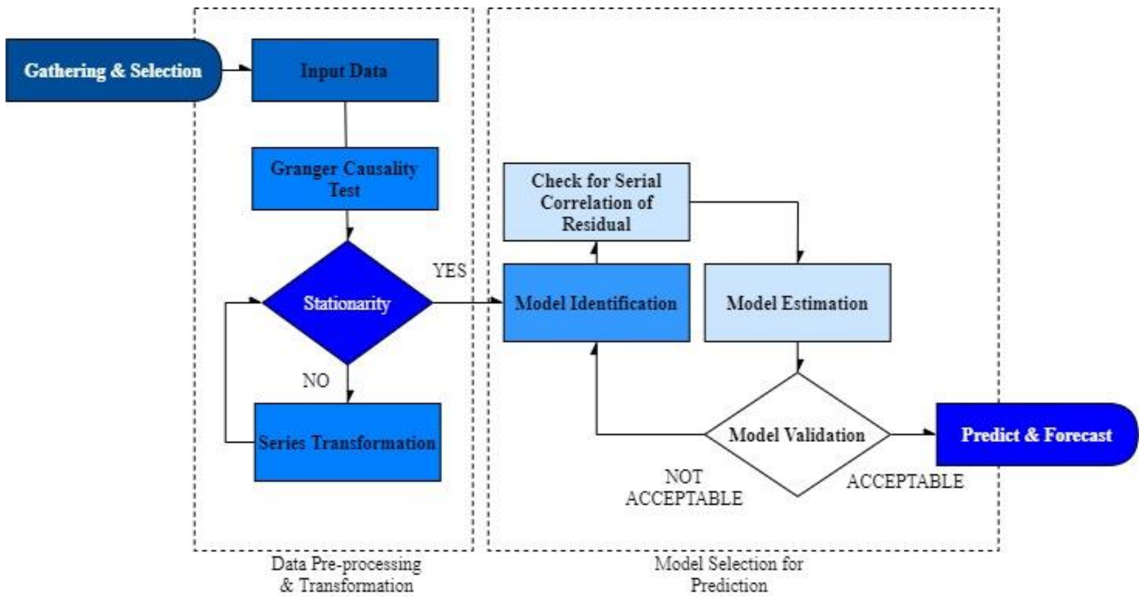


Figure 25: VAR implementation flow

Vector Autoregression (VAR) is a multivariate time series forecasting that is used when two or more time series influence each other. VAR is most suitable when the goal is to examine the relationship between the selected time series (variables) over time. Vector of variables is modelled as depending on their own lags and on the lags of every other variable in the vector. Three (3) variables indicating house prices, sales and crime counts that is extracted and expected to be explanatory variables for each other. For example, the structure of equations for a VAR (1) model with two time series (variables `**Y₁**` and `**Y₂**`) is as follows:

$$Y_{1,t} = \alpha_1 + \beta_{11,1} Y_{1,t-1} + \beta_{12,1} Y_{2,t-1} + \epsilon_{1,t}$$
$$Y_{2,t} = \alpha_2 + \beta_{21,1} Y_{1,t-1} + \beta_{22,1} Y_{2,t-1} + \epsilon_{2,t}$$

Equation 4: $\boldsymbol{Y}_{\{1,\, t-1\}}$ and $\boldsymbol{Y}_{\{2,\, t-1\}}$ are the first lag of time series $\mathbf{Y_1}$ and $\mathbf{Y_2}$ respectively.

The statsmodel algorithm for VAR () was derived follow in large part the methods and notation from the author in [68]. Each time series is visualized to analyse any pattern, trend or seasonality as seen in figure below.

Causation and Co integration test

		average_price_x	houses_sold_x	no_of_crimes_x
London	**average_price_y**	1.00	0.00	0.03
	houses_sold_y	0.00	1.00	0.00
	no_of_crimes_y	0.07	0.01	1.00
Kensington & Chelsea	**average_price_y**	1.00	0.02	0.07
	houses_sold_y	0.00	1.00	0.03
	no_of_crimes_y	0.10	0.10	1.00
Barking & Dagenham	**average_price_y**	1.00	0.02	0.07
	houses_sold_y	0.03	1.00	0.04
	no_of_crimes_y	0.15	0.04	1.00

Table 10 : Result of Granger Causality Test

As seen in Table 10 above, the relationship between this time series is also tested for causation using Granger Causality Test where variables that affect the house prices can be evaluated in *average_price_y* rows. For example, p-value of 0.0 (*average_price_y*) for *houses_sold* in London represents the Grangers Causality test for *houses_sold* causing *average_price*. The test revealed that the p-value is less that the significance level of 0.05. So, the null hypothesis can be rejected and conclude that *houses_sold* is causing *average_price*. This is similar for Kensington & Chelsea and Barking & Dagenham. However, *no_of_crimes* p-value in both borough of Kensington & Chelsea and Barking & Dagenham are slightly above the significant level thus confirming the null hypothesis. This meant that *no_of_crimes* does not cause changes to *average_price*. This is however different for *average_prices* in London, where no_of _crimes caused the *average price*.

This makes this system of multi time series a good candidate for using VAR models to forecast. Other interesting discoveries is that both *average_price* and *no_of_crime* is always causing the *houses_sold* but *average_price* never caused any changes to the *no_of_crimes.*

		test statistic	> C (95%)	significance
London	**average_price**	19.79	>24.2761	False
	houses_sold	4.5	>12.3212	False
	no_of_crimes	0.02	>4.1296	False
Kensington & Chelsea	**average_price**	18.3	>24.2761	False
	houses_sold	5.27	>12.3212	False
	no_of_crimes	0.01	>4.1296	False
Barking & Dagenham	**average_price**	16.45	>24.2761	False
	houses_sold	4.83	>12.3212	False
	no_of_crimes	0.01	>4.1296	False

Table 11: Result of Johannsen Cointegration test

Cointegration test also revealed that false significance for all the time series implying that there is no long run and statistically significant relationship. This means VAR (short run) model is a suitable model candidate and not for long term [69] On the contrary, if there is significant relationship, vector error correction model (VECM) analysis is needed. However, according to this result the unrestricted VAR model is sufficient.

Selection of VAR(p) order

The time series also use the same ADF test in Table 7. In selecting the VAR (p) order, iteration of the VAR model with parameter from 1 to 9 in increasing order to pick the model with least Akaike Information Criterion (AIC). Though the usual practice is to look at the AIC, best fit comparison estimates of BIC, FPE and HQIC can also be considered [54].

	Best VAR ( ) Model	Lowest AIC
London	4	47.442757
Kensington & Chelsea	4	37.936762
Barking & Dagenham	4	31.885573

Table 12 : Summary of Least AIC for all datasets

The results from the Table 19 in appendix suggest that the appropriate model for all the data is VAR(4) because all three datasets gave lag order 4 as minimum lag order. This is summarized in Table 12. Now that the model is set up with lag order 4, the forecasting ability is tested by inquiring the model to forecast n steps ahead.

Residual Analysis

After a suitable VAR model for the variables were estimated, this stage of the analysis deals with the diagnostic checking process. There are several methods to control the robustness of the model, graphical analysis tools and statistical tests for the residuals for the diagnostic checks were selected. Serial correlation of residuals is used to check if there is any leftover pattern in the residuals (errors). Checking for serial correlation using the Durbin Watson's Statistics is to ensure that the model is sufficiently able to explain the variances and patterns in the time series. The value of this statistic can vary between 0 and 4. The closer it is to the value 2, then there is no significant serial correlation. The closer to 0, there is a positive serial correlation, and the closer it is to 4 implies negative serial correlation.

		Watson's Statistics
London	**average_price**	1.94
	houses_sold	2.10
	no_of_crimes	2.01
Kensington & Chelsea	**average_price**	1.99
	houses_sold	1.99
	no_of_crimes	1.98
Barking & Dagenham	**average_price**	2.09
	houses_sold	2.00
	no_of_crimes	2.00

Table 13: Watson's Statistics Result

To forecast, the VAR model expects up to the lag order number of observations from the past data. This is because, the terms in the VAR model are essentially the lags of the various time series in the dataset, many of the previous values is provided as indicated by the lag order used by the model.

As seen as the differenced data forecast result (Table 26**Error! Reference source not found.Error! Reference source not found.**) in appendix, forecast values along with associated standard errors are in stationarity, thus the values need to be fit in the original form by *inverting* transformation: the final step of fitting the differenced data (*df_differenced*) into the original data frame. This is important to see the data in its original form with date as the index (Table 27). It is found that the result matches the initial hypothesis that its only able to predict short term based on the cointegration test result.

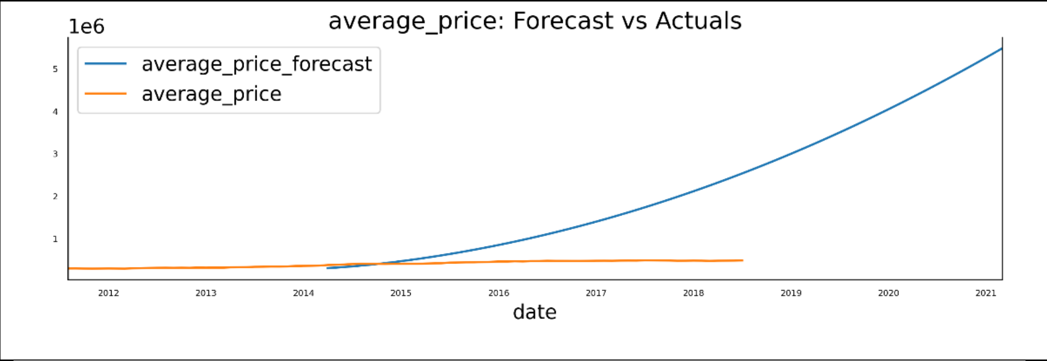
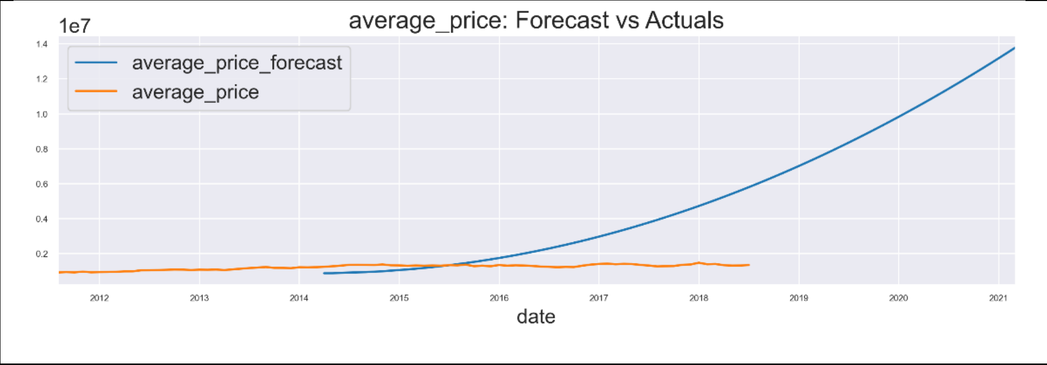
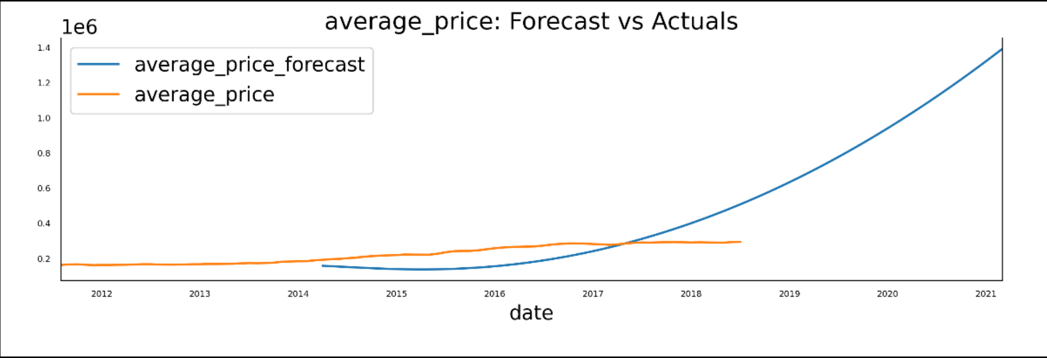
London	
Kensington & Chelsea	
Barking & Dagenham	

Figure 26: Visualization of the VAR average_price forecast result

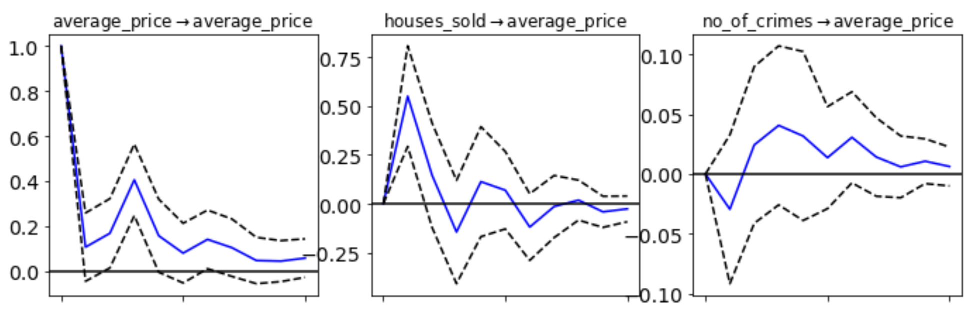
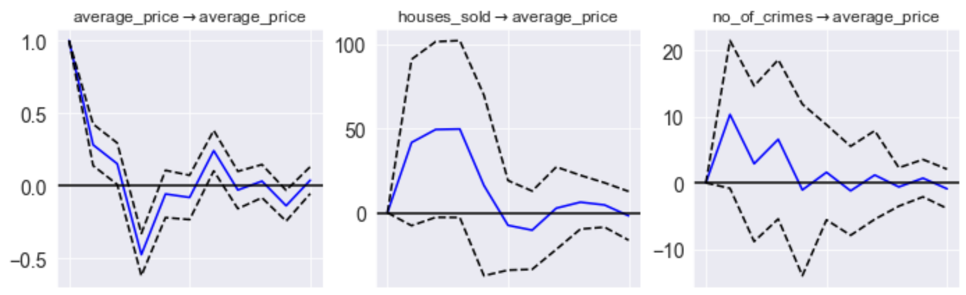
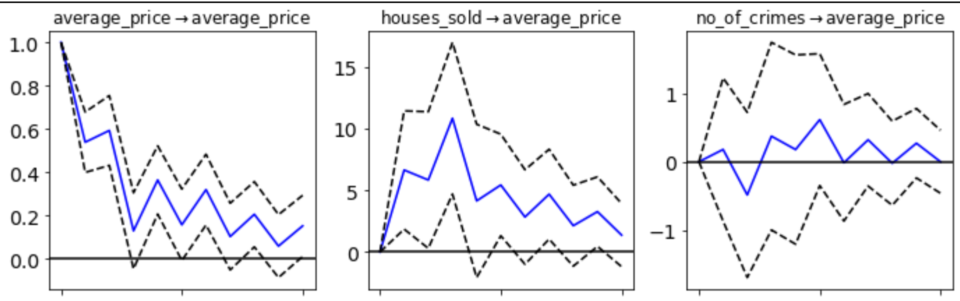
London	
Kensington & Chelsea	
Barking & Dagenham	

Table 14: Impulse Response of other variables to Average Price

Table 14 shows the impulse response of each variable to average prices of houses in London in general, Kensington & Chelsea and Barking & Dagenham. For house prices in London, initial positive shock to *houses_sold* increased the *average_price* sharply a fluctuate towards negative value but the effect starts to decay to 0 consistent with the behaviour of VAR model. Similarly, Kensington & Chelsea had a positive increase and stabilizes for a moment before decay to 0. *House_sold* positive shock in Barking & Dagenham, however, lead the prices up slower and begin decay at a slow pace.

Initial shock of *no_of_crimes* to seems to cause a minimal decrease in average prices for house prices in London but minimally increased the house prices for the boroughs (Kensington & Chelsea and Barking & Dagenham).

The RMSE for all house prices with VAR (4) model are significantly high at 2,35 mil, 5,52 mil and 2,35mil a huge gap from actual data (attached in appendix for comparison). However, looking at a shorter term of 6 month of the first validation set, the RMSE are better as seen in Table 16. proving the earlier hypothesis that this VAR(4) forecast better in short term.

London	Column: average_price mape : 4.0977 me : 1831187.1036 mae : 1831187.1036 mpe : 4.0977 corr : 0.9321 minmax : 0.6966 rmse : 2356034.5137	Column: houses_sold mape : 2.0168 me : -16580.1118 mae : 16775.3468 mpe : -1.9872 corr : 0.2031 minmax : 2.0135 rmse : 21573.3762	Column: no_of_crimes mape : 6.4413 me : 404639.3263 mae : 404639.3263 mpe : 6.4413 corr : 0.5205 minmax : 0.7112 rmse : 539794.6559
Kensington & Chelsea	Column: average_price mape : 3.0641 me : 4013759.9777 mae : 4018368.9863 mpe : 3.0591 corr : 0.7323 minmax : 0.5766 rmse : 5518427.4863	Column: houses_sold mape : 3.6221 me : 543.8342 mae : 543.8342 mpe : 3.6221 corr : -0.2576 minmax : 0.7181 rmse : 581.8347	Column: no_of_crimes mape : 7.7002 me : 13011.9737 mae : 13019.1368 mpe : 7.6966 corr : 0.2316 minmax : 0.7516 rmse : 16869.4743
Barking & Dagenham	Column: average_price mape : 1.0327 me : 270947.5579 mae : 282218.1538 mpe : 0.9652 rmse : 429115.0074 corr : 0.934 minmax : 0.3802	Column: houses_sold mape : 0.475 me : -82.767 mae : 82.767 mpe : -0.475 rmse : 92.9427 corr : 0.3824 minmax : 0.475	Column: no_of_crimes mape : 8.4199 me : 12457.6461 mae : 12457.9154 mpe : 8.4197 rmse : 16134.7611 corr : 0.4028 minmax : 0.7707

Table 15: Forecast Accuracy Metrics

	RMSE First 6 Month	**RMSE for Total Dataset**
London	51,970.14031	2,356,034.5137
Kensington & Chelsea	40,7297.2101	5,518,427.4863
Barking & Dagenham	51,951.55186	429,115.0074

Table 16: RMSE for Short Forecast vs Total Forecast

4.4.3 Univariate vs Multivariate Time Series

Results in both univariate and multivariate models are compared. ARIMA (4,3,6) and (4,0, 6) was chosen as the best model for forecasting average price for houses in London, its most expensive and cheapest borough (Kensington & Chelsea) and (Barking & Dagenham) respectively. The validation set is used to compare the performance of this ARIMA model with the multivariate VAR (4) model. Root mean squared error (RMSE) statistics are used for the performance evaluation tool.

	ARIMA ( ) Model	**RMSE**	**VAR ( ) Model**	**RMSE**
London	4,3,6	6,222.666	4	2,356,034.514
Kensington & Chelsea	4,0,6	62,975.554	4	5,518,427.486
Barking & Dagenham	4,0,6	3,183.897	4	429,115.0074

Table 17 : RMSE statistic of ARIMA vs VAR model

According to the RMSE statistic in the table above, univariate ARIMA gives better out-of-sample performance for average prices for all three datasets compared to VAR. It is important to also note that the ARIMA model also performs better for London overall average house prices and the cheapest borough (Barking & Dagenham). While VAR model achieved better RMSE results for 6-month forecast result (short term of the validation set) (see Table 16).

5 Conclusion and future work

This study aimed to understand the application of ‘smartness’ in the field of real estate and how smart city plays a key role in addressing the worlds rapid urbanization. Real estate in this regard takes a centre role in urbanization and proves to be a key to the implementation of a smart city. This use techniques which are commonly utilized in forecasting and attempts predictions of house prices in London. Housing market stability is important to ensure the citizen of any smart city to have access to sufficient and affordable housing because mortgages and rent takes a substantial portion of household wealth and expenditure. It is then, important for policy makers to track the housing prices and understand market behaviour for any policy to work. Tracking house prices can be a larger component to the smart city dashboard initiative where a suitable data mining or machine learning algorithm can track price changes and make accurate prediction of the direction of the housing market.

Literature review revealed some existing machine learning methods to track housing prices. The study framework concludes that a high granularity and size of data is vital in ensuring the accuracy of the forecast. Best practice steps such as test harnessing can ensure testing conditions are rigorous and detailed. Several machine learning algorithms were evaluated namely univariate and multivariate time series analysis. They were assessed for their ability to forecast house prices. They were tested in a test harness and walk-in validation to test the forecast result on unseen data. Univariate time series ARIMA (4, 0, 6) was used to forecast for August 2018. It was found to be more successful in predicting house prices in Kensington & Chelsea borough than multivariate time series VAR (4) by a considerable margin. This is consistent with the results found in the literature review [55], [56] where ARIMA outperformed VAR in performance. As these two algorithms has distinct methodology and system requirement the conclusions will be summarised with regards the following issues:

1. Data Extraction
2. Algorithm tuning
3. Extracted Knowledge
4. Future Research Direction

5.1 Data Extraction

Data extraction is a critical component of the whole process. To ensure that housing market data were comprehensive, various data sources were integrated when presenting limitations to normalize them into appropriate time interval and instances. The main limitation of our study is the relatively small number of observations because it uses only monthly data which were considered sufficient, but a weekly interval could improve the analysis of the training sets. The monthly frequency of the data used may be the reason for the intermediate results of VAR forecast error. There are many macroeconomic factors that could have been included in the studies as variables for the multivariate time series (VAR) that could prove more explanatory to house prices in London such as: monthly house supply, rent prices, building permits and interest rate and gross domestic product [70].

5.2 Machine learning algorithms

The different machine learning algorithms that were examined in this study exhibit different characteristics and strengths. Generally, univariate time series (ARIMA) was easier to build, but it takes a long time to optimize. The simple deduction is that the complexity of the model was medium thus marginal result from each machine learning was expected. All the results are very similar with minimal range of error. In [54], the authors used only ACF and PCF to determine the hyperparameter (p, d, q). This created considerable gaps with potential best (p, d, q) Learning form literature [54] limitation, grid search proves to be successful in getting the best hyperparameter. However, it takes a lot of time to execute (4 hours on average) thus time and cost might affect the efficiency of the model. In the real-world business environment, this could prove inefficient. Key takeaways would be to select very specific lag range for the parameter search using the ADF/PACF graph or skip the insignificant lags altogether. It will reduce processing time for hyperparameter grid-search.

The insignificant result from cointegration test also reveals that the dataset is not able to produce a statistically significant and long run relationship early on. Parameter tuning in VAR was easier, but there may be more rigorous way to select the lag order (p) that would be beneficial.

5.3 Extracted knowledge

The most important results of this work are summarized below:

1. Expensive boroughs in London have a higher magnitude of price fluctuations albeit the visually similar increasing price throughout the study period. Kensington & Chelsea borough has the highest price change year-on-year.
2. House prices in London over the decades generally have an upward trend with minor reduction in prices following shocks but it takes generally only two to three years to rebound to pre-shock prices.
3. The accuracy of the model in ARIMA (4, 0, 6) proves superior to VAR (4) in forecasting London monthly house prices using average price.
4. The result suggest that past values of the housing prices are sufficient to predict it future values.
5. The houses_sold is found to have causation to the average_price albeit a small significance proven by the co-integration test.
6. In retrospect, the prediction accuracy and result for ARIMA confirms the same result found in the literature review, that Univariate Time Series Analysis could provide better accuracy than Multivariate Time Series Analysis.
7. It is however, noted that the model accuracy for multivariate (VAR) can significantly be improved if more time series (variable) that have higher explanatory or cointegration power are included in the model. Causation and cointegration test can be a good determinant for selecting time series with higher explanatory power.

5.4 Future research directions

A quality work and research require time-consuming process and wide variety in approach. For that reason, within the time limitation, an idealistic approach was taken to achieve the study goal. However, there a key direction if the study is to be taken further.

The missing part of this studies is the iteration of the same forecast model to be tested in all the 33 boroughs in London. With the time limitation, only one (1) borough which is Kensington & Chelsea was selected due to the dynamic pricing history. It could be argued that other borough might have different pricing behaviours altogether, but the initial data analysis shows that movement of prices is almost similar in pattern albeit different in magnitude.

Although the dataset extracted was sufficient for the study, a higher granular data that covers weekly price changes could prove better at improving the final forecast model. This can also minimize the shocks to the price changes if there is any event or disaster that affect the overall house price supply and demand, ultimately affecting the house prices.

Another aspect that could be interesting to study is increasing the variables to be tested in the multivariate time series where data on house supply and demand might have higher correlation factor in comparison to the current tested variables.

On a larger scale, interesting direction for this study could be to create a live-feed machine learning model that can track housing prices and other real estate performance metric (rent, permits and new construction starts etc) in a smart city. This smart real estate can be a part of a city dashboard initiative optimizing the housing supply, demand and needs for its citizen. As population grows, the real estate market is only going to grow.

6 Bibliography

[1] "ISO/TS 37151:2015(en), Smart community infrastructures — Principles and requirements for performance metrics." https://www.iso.org/obp/ui/#iso:std:iso:ts:37151:ed-1:v1:en (accessed Mar. 09, 2021).

[2] S. Liapis, K. Christantonis, V. Chazan-Pantzalis, A. Manos, D. Elizabeth Filippidou, and C. Tjortjis, "A methodology using classification for traffic prediction: Featuring the impact of&nbsp;COVID-19," *Integr. Comput.-Aided Eng.*, vol. Preprint, no. Preprint, pp. 1–19, Jan. 2021, doi: 10.3233/ICA-210663.

[3] B. Sterling, "Stop Saying 'Smart Cities,'" *The Atlantic*, Feb. 12, 2018. https://www.theatlantic.com/technology/archive/2018/02/stupid-cities/553052/ (accessed Apr. 12, 2021).

[4] F. K. S. Chan *et al.*, "'Sponge City' in China—A breakthrough of planning and flood risk management in the urban context," *Land Use Policy*, vol. 76, pp. 772–778, Jul. 2018, doi: 10.1016/j.landusepol.2018.03.005.

[5] R. Dameri, "Searching for Smart City definition: a comprehensive proposal," *Int. J. Comput. Technol.*, vol. 11, p. 2544, Oct. 2013, doi: 10.24297/ijct.v11i5.1142.

[6] "A Smart City Ecosystem Framework for building sustainable smart cities," *Strategy of Things*, Jan. 24, 2018. https://strategyofthings.io/smart-city-ecosystem (accessed Apr. 17, 2021).

[7] "EUR-Lex - 32016R0679 - EN - EUR-Lex." https://eur-lex.europa.eu/eli/reg/2016/679/oj (accessed Apr. 17, 2021).

[8] E. Ismagilova, L. Hughes, N. P. Rana, and Y. K. Dwivedi, "Security, Privacy and Risks Within Smart Cities: Literature Review and Development of a Smart City Interaction Framework," *Inf. Syst. Front.*, Jul. 2020, doi: 10.1007/s10796-020-10044-1.

[9] K. Hein and F. B. A. Rosman, "Data-driven Approach for Smart Urban Planning Tool Development," p. 8.

[10] A. Mystakidis and C. Tjortjis, "Big Data Mining for Smart Cities: Predicting Traffic Congestion using Classification," *2020 11th Int. Conf. Inf. Intell. Syst. Appl. IISA*, 2020, doi: 10.1109/IISA50023.2020.9284399.

[11] C. Jing, M. Du, S. Li, and S. Liu, "Geospatial Dashboards for Monitoring Smart City Performance," *Sustainability*, vol. 11, no. 20, Art. no. 20, Jan. 2019, doi: 10.3390/su11205648.

[12] L. Deren, Y. Wenbo, and S. Zhenfeng, "Smart city based on digital twins," *Comput. Urban Sci.*, vol. 1, no. 1, p. 4, Mar. 2021, doi: 10.1007/s43762-021-00005-y.

[13] X. Hua *et al.*, “The City Brain: Practice of Large-Scale Artificial Intelligence in the Real World,” *IET Smart Cities*, vol. 1, May 2019, doi: 10.1049/iet-smc.2019.0034.

[14] M. M. Rathore, A. Paul, A. Ahmad, and G. Jeon, “From Smart City towards Next Generation Super City Planning,” *Int. J. Semantic Web Inf. Syst.*, vol. 13, no. 1, p. 20, 2017.

[15] E. Allameh, M. Heidari, de Vries, H. J. P. Timmermans, and J. Beetz, *Smart Home as a smart real estate, A state of the art review*. 2011.

[16] D. Al-Abri, A. Malik, S. Al-Saadi, M. Albadi, Y. Charabi, and N. Hosseinzadeh, “Smart Grids and Smart Buildings,” 2021. doi: 10.1007/978-1-4614-6431-0_78-2.

[17] mHealthIntelligence, “MD City Integrates Telehealth With Smart City Hub to Monitor Coronavirus,” *mHealthIntelligence*, Mar. 23, 2020. https://mhealthintelligence.com/news/md-city-integrates-telehealth-with-smart-city-hub-to-monitor-coronavirus (accessed Apr. 21, 2021).

[18] R. Giffinger, H. Gudrun, Gudrun, and G. Haindlmaier, “Smart cities ranking: An effective instrument for the positioning of the cities,” *ACE Archit. City Environ.*, vol. 4, Feb. 2010.

[19] “IESE Cities in Motion Index 2020 | Cities in Motion.” https://blog.iese.edu/cities-challenges-and-management/2020/10/27/iese-cities-in-motion-index-2020/ (accessed Apr. 17, 2021).

[20] A. De Mauro, M. Greco, and M. Grimaldi, “What is big data? A consensual definition and a review of key research topics,” *AIP Conf. Proc.*, vol. 1644, no. 1, pp. 97–104, Feb. 2015, doi: 10.1063/1.4907823.

[21] A. Kousis and C. Tjortjis, “Data Mining Algorithms for Smart Cities: A Bibliometric Analysis,” *Algorithms*, vol. 14, no. 8, Art. no. 8, Aug. 2021, doi: 10.3390/a14080242.

[22] D. Agrawal *et al.*, “Mind your Ps and Vs: A perspective on the challenges of big data management and privacy concerns,” in *2015 International Conference on Big Data and Smart Computing (BIGCOMP)*, Feb. 2015, pp. 1–6. doi: 10.1109/35021BIGCOMP.2015.7072814.

[23] Z. Allam and Z. A. Dhunny, “On big data, artificial intelligence and smart cities,” *Cities*, vol. 89, pp. 80–91, Jun. 2019, doi: 10.1016/j.cities.2019.01.032.

[24] “Big Data Impacts Data Management: The 5 Vs of Big Data.” https://davebeulke.com/big-data-impacts-data-management-the-five-vs-of-big-data/ (accessed Apr. 26, 2021).

[25] A. M. S. Osman, “A novel big data analytics framework for smart cities,” *Future Gener. Comput. Syst.*, vol. 91, pp. 620–633, Feb. 2019, doi: 10.1016/j.future.2018.06.046.

[26] C.-W. Tsai, C.-F. Lai, H.-C. Chao, and A. V. Vasilakos, “Big data analytics: a survey,” *J. Big Data*, vol. 2, no. 1, p. 21, Oct. 2015, doi: 10.1186/s40537-015-0030-3.

[27] J. Han and M. Kamber, *Data Mining: Concepts and Techniques*. 2000.

[28] L. Xu, C. Jiang, J. Wang, J. Yuan, and Y. Ren, “Information Security in Big Data: Privacy and Data Mining,” *IEEE Access*, vol. 2, pp. 1–28, Jan. 2014, doi: 10.1109/ACCESS.2014.2362522.

[29] P. Lecomte, “New boundaries Conceptual framework for the analysis of commercial real estate in smart cities,” *J. Prop. Invest. Finance*, vol. 37, pp. 118–135, Jan. 2019, doi: 10.1108/JPIF-10-2018-0083.

[30] “Are smart cities the answer to rising populations?,” *the Guardian*, Jan. 28, 2013. http://www.theguardian.com/sustainable-business/blog/smart-cities-answer-population-fewer-resources (accessed May 21, 2021).

[31] X. G. Li and Q. M. Li, “THE APPLICATION OF DATA MINING TECHNOLOGY IN REAL ESTATE MARKET PREDICTION,” p. 6, 2006.

[32] D. Faggella, “Machine Learning in Real Estate - Trends and Applications Presented,” *Emerj*. https://emerj.com/ai-sector-overviews/machine-learning-in-real-estate-trends-and-applications/ (accessed May 21, 2021).

[33] D. Djenouri, R. Laidi, Y. Djenouri, and I. Balasingham, “Machine Learning for Smart Building Applications: Review and Taxonomy,” *ACM Comput. Surv.*, vol. 52, Feb. 2019, doi: 10.1145/3311950.

[34] “Smart building sensors for workplace analytics,” *PointGrab*. https://www.pointgrab.com/ (accessed May 21, 2021).

[35] “BuildingIQ – Building Energy and Operational Intelligence.” https://buildingiq.com/ (accessed May 21, 2021).

[36] O. Barzilay, “Property Management May Be The Next Frontier For AI,” *Forbes*. https://www.forbes.com/sites/omribarzilay/2017/03/14/property-management-may-be-the-next-frontier-for-ai/ (accessed May 21, 2021).

[37] “Commercial Real Estate Platform,” *VTS*. https://www.vts.com/ (accessed May 21, 2021).

[38] “Property Management Software | AppFolio.” https://www.appfolio.com/ (accessed May 21, 2021).

[39] “Zenplace | Making Rentals Easy for Landlords and Property Owners| Online Software Platform for Rental Properties.” https://www.zenplace.com/ (accessed May 21, 2021).

[40] “Zillow: Real Estate, Apartments, Mortgages & Home Values,” *Zillow*. https://www.zillow.com/ (accessed May 21, 2021).

[41] M. Figurska and R. Wisniewski, “Fundamental Analysis – Possiblity of Application on the Real Estate Market,” *Real Estate Manag. Valuat.*, vol. 24, Dec. 2016, doi: 10.1515/remav-2016-0028.

[42] Buxton, “Micro Level Data: Are You Overlooking This Valuable Source of Real Estate Intelligence?” https://www.buxtonco.com/blog/micro-level-data-valuable-source-of-real-estate-business-intelligence (accessed Jun. 07, 2021).

[43] R. Jefferies, “History and development of real estate investment (property) valuation models,” Jan. 2017.

[44] N. Kok, E.-L. Koponen, and C. A. Martínez-Barbosa, "Big Data in Real Estate? *From Manual Appraisal to Automated Valuation*," *J. Portf. Manag.*, vol. 43, no. 6, pp. 202–211, Sep. 2017, doi: 10.3905/jpm.2017.43.6.202.

[45] O. Demirci, *Automated Valuation Models (AVMs): Machine Learning, namely Mass (Advanced) Valuation Methods and Algorithms*. 2021. doi: 10.13140/RG.2.2.12649.42080.

[46] V. Limsombunc, C. Gan, and M. Lee, "House Price Prediction: Hedonic Price Model vs. Artificial Neural Network," *Am. J. Appl. Sci.*, vol. 1, no. 3, pp. 193–201, Mar. 2004, doi: 10.3844/ajassp.2004.193.201.

[47] S. Zeicu, "STATISTICAL MODELING APPLIED IN REAL ESTATE VALUATION," p. 10.

[48] "Automated Valuation Model (AVM) – Examples of Uses of Big Data Software in Real Estate | ASPER BROTHERS," Nov. 14, 2019. https://asperbrothers.com/blog/automated-valuation-model/ (accessed Jun. 06, 2021).

[49] Karl E. Case, "The Central Role of Home Prices in the Current Financial Crisis: How Will the Market Clear?," *Brook. Pap. Econ. Act.*, vol. 2008, no. 2, pp. 161–193, 2009, doi: 10.1353/eca.0.0021.

[50] J. Brzezicka, "Assessing the Possibility of Implementing Tools of Technical Analysys for Real Estate Market Analysis," *Real Estate Manag. Valuat.*, vol. 24, Jul. 2016, doi: 10.1515/remav-2016-0016.

[51] S. (Bratu) Mihaela, "The Use of Varma Models in Forecasting Macroeconomic Indicators," *J. Econ. Sociol.*, vol. 6, Nov. 2013, doi: 10.14254/2071-789X.2013/6-2/9.

[52] "Zestimate Forecast Methodology - Zillow Research." https://www.zillow.com/research/zestimate-forecast-methodology/ (accessed Aug. 27, 2021).

[53] K. Liow, "Dynamic relationship between stock and property markets," *Appl. Financ. Econ.*, vol. 16, pp. 371–376, Feb. 2006, doi: 10.1080/09603100500390885.

[54] N. Y. Zainun, F. E. Mohamed Ghazali, and M. S. Mohd Sallehudin, "Prediction of Low Cost Housing Demand in Malaysia Using ARIMA Model," *MATEC Web Conf.*, vol. 47, p. 04008, 2016, doi: 10.1051/matecconf/20164704008.

[55] T. McGough and S. Tsolacos, "Forecasting commercial rental values using ARIMA models," *J. Prop. Valuat. Invest.*, vol. 13, no. 5, pp. 6–22, Jan. 1995, doi: 10.1108/14635789510147801.

[56] A. S. S. Kestel and B. Yilmaz, "Forecasting house prices in Turkey: GLM, VAR and time series approaches," *Pressacademia*, vol. 9, no. 4, pp. 274–291, Dec. 2020, doi: 10.17261/Pressacademia.2020.1310.

[57] J. P. Brown, H. Song, and A. McGillivray, "Forecasting UK house prices: A time varying coefficient approach," *Econ. Model.*, vol. 14, no. 4, pp. 529–548, Oct. 1997, doi: 10.1016/S0264-9993(97)00006-0.

[58] "London Datastore – Greater London Authority." https://data.london.gov.uk/ (accessed May 21, 2021).

[59] "Price Paid Data," *GOV.UK*. https://www.gov.uk/government/statistical-data-sets/price-paid-data-downloads (accessed Jun. 05, 2021).

[60] J. Brownlee, "Arithmetic, Geometric, and Harmonic Means for Machine Learning," *Machine Learning Mastery*, Dec. 16, 2019. https://machinelearningmastery.com/arithmetic-geometric-and-harmonic-means-for-machine-learning/ (accessed Jul. 22, 2021).

[61] "Power BI uses key influencers using ML.NET | .NET," *Microsoft*. https://dotnet.microsoft.com/apps/machinelearning-ai/ml-dotnet/customers/power-bi (accessed Jun. 10, 2021).

[62] "Where has the property crash hit hardest?," *the Guardian*, Nov. 22, 2008. http://www.theguardian.com/money/2008/nov/22/property-crash-uk-house-prices (accessed Aug. 30, 2021).

[63] "House prices most sluggish in prime central London and north-east England when Covid-19 struck." https://www.knightfrank.com/research/article/2020-04-29-house-prices-most-sluggish-in-prime-central-london-and-northeast-england-when-covid19-struck (accessed Aug. 30, 2021).

[64] "Savills UK | UK housing value hits record £7.56 trillion high." https://www.savills.co.uk/insight-and-opinion/savills-news/311889-0/uk-housing-value-hits-record-%C2%A37.56-trillion-high (accessed Aug. 30, 2021).

[65] "Number of dwellings in London 2019," *Statista*. https://www.statista.com/statistics/585272/number-of-dwellings-london-uk/ (accessed Jun. 21, 2021).

[66] "Data Mining: Practical Machine Learning Tools and Techniques." https://www.cs.waikato.ac.nz/ml/weka/book.html (accessed Aug. 28, 2021).

[67] S. B H and G. Dagnew, *Grid Search-Based Hyperparameter Tuning and Classification of Microarray Cancer Data*. 2019, p. 8. doi: 10.1109/ICACCP.2019.8882943.

[68] H. Lutkepohl, "Forecasting with VARMA Models," in *Handbook of Economic Forecasting*, vol. 1, Elsevier, 2006, pp. 287–325. Accessed: Jun. 07, 2021. [Online]. Available: https://ideas.repec.org/h/eee/ecofch/1-06.html

[69] B. N. Adeleye, "CrunchEconometrix: Time Series Analysis (Lecture 4 Part 1): Johansen Cointegration Test in EViews," *CrunchEconometrix*, Mar. 24, 2018. https://cruncheconometrix.blogspot.com/2018/03/time-series-analysis-lecture-4-part-1.html (accessed Aug. 28, 2021).

[70] P. Sivitanides, "Macroeconomic drivers of London house prices," *J. Prop. Invest. Finance*, vol. 36, Aug. 2018, doi: 10.1108/JPIF-02-2018-0012.

7 Appendix

Example of appendix (extra figures, source code, etc.). Pay attention that reviewers are not forced to read the appendixes.

7.1.1 ARIMA (p, d, q) Grid Search

London	**Kensington & Chelsea**	**Barking & Dagenham**
ARIMA(0, 0, 0) RMSE=188504.549	ARIMA(0, 0, 0) RMSE=34779.711	ARIMA(0, 0, 0) RMSE=101277.053
ARIMA(0, 0, 1) RMSE=96595.839	ARIMA(0, 0, 1) RMSE=34996.651	ARIMA(0, 0, 1) RMSE=51394.892
ARIMA(0, 0, 2) RMSE=55831.817	ARIMA(0, 0, 2) RMSE=33533.361	ARIMA(0, 0, 2) RMSE=29393.647
ARIMA(0, 0, 3) RMSE=36099.730	ARIMA(0, 0, 3) RMSE=28813.410	ARIMA(0, 0, 3) RMSE=17883.055
ARIMA(0, 0, 4) RMSE=27339.069	ARIMA(0, 0, 4) RMSE=28758.436	ARIMA(0, 0, 4) RMSE=16377.872
ARIMA(0, 0, 5) RMSE=22463.636	ARIMA(0, 0, 5) RMSE=28846.338	ARIMA(0, 0, 5) RMSE=11470.313
ARIMA(0, 0, 6) RMSE=20063.723	ARIMA(0, 0, 6) RMSE=28689.690	ARIMA(0, 0, 6) RMSE=12575.757
ARIMA(0, 1, 0) RMSE=4614.125	ARIMA(0, 0, 7) RMSE=28946.605	ARIMA(0, 1, 0) RMSE=2776.012
ARIMA(0, 1, 1) RMSE=4496.908	ARIMA(0, 0, 8) RMSE=29156.328	ARIMA(0, 1, 1) RMSE=2447.792
ARIMA(0, 1, 2) RMSE=4422.941	ARIMA(0, 0, 9) RMSE=29419.783	ARIMA(0, 1, 2) RMSE=2331.633
ARIMA(0, 1, 3) RMSE=4350.217	ARIMA(0, 0, 10) RMSE=29550.234	ARIMA(0, 1, 3) RMSE=2360.357
ARIMA(0, 1, 4) RMSE=4278.099	ARIMA(0, 0, 11) RMSE=29854.595	ARIMA(0, 1, 4) RMSE=2333.758
ARIMA(0, 1, 5) RMSE=4263.540	ARIMA(0, 0, 12) RMSE=29501.568	ARIMA(0, 1, 5) RMSE=2259.079
ARIMA(0, 1, 6) RMSE=4220.855	ARIMA(0, 1, 0) RMSE=48903.679	ARIMA(0, 1, 6) RMSE=2273.491
ARIMA(0, 2, 0) RMSE=5270.850	ARIMA(0, 1, 1) RMSE=35066.235	ARIMA(0, 2, 0) RMSE=2176.505
ARIMA(0, 2, 1) RMSE=4421.570	ARIMA(0, 1, 2) RMSE=35124.836	ARIMA(0, 2, 1) RMSE=2166.145
ARIMA(0, 2, 2) RMSE=4338.681	ARIMA(0, 1, 3) RMSE=33838.257	ARIMA(0, 2, 2) RMSE=2121.042
ARIMA(0, 2, 3) RMSE=4340.233	ARIMA(0, 1, 4) RMSE=28979.596	ARIMA(0, 2, 3) RMSE=1916.518
ARIMA(0, 2, 4) RMSE=4422.404	ARIMA(0, 1, 5) RMSE=29073.999	ARIMA(0, 2, 4) RMSE=1915.314
ARIMA(0, 2, 5) RMSE=4328.470	ARIMA(0, 1, 6) RMSE=29172.452	ARIMA(0, 2, 5) RMSE=1918.518
ARIMA(0, 2, 6) RMSE=4325.545	ARIMA(0, 1, 7) RMSE=28878.098	ARIMA(0, 2, 6) RMSE=1909.031
ARIMA(0, 3, 0) RMSE=9208.577	ARIMA(0, 1, 8) RMSE=29304.454	ARIMA(0, 3, 0) RMSE=3199.234
ARIMA(0, 3, 1) RMSE=5282.052	ARIMA(0, 1, 9) RMSE=29376.748	ARIMA(0, 3, 1) RMSE=2181.505
ARIMA(0, 3, 2) RMSE=4244.757	ARIMA(0, 1, 10) RMSE=29736.427	ARIMA(0, 3, 2) RMSE=2182.992
ARIMA(0, 3, 3) RMSE=4280.677	ARIMA(0, 1, 11) RMSE=29834.611	ARIMA(0, 3, 3) RMSE=2186.518
ARIMA(0, 3, 4) RMSE=4286.497	ARIMA(0, 1, 12) RMSE=30072.760	ARIMA(0, 3, 4) RMSE=2018.042
ARIMA(0, 3, 5) RMSE=4266.909	ARIMA(0, 2, 0) RMSE=87750.160	ARIMA(0, 3, 5) RMSE=2022.673
ARIMA(0, 3, 6) RMSE=4242.624	ARIMA(0, 2, 1) RMSE=49014.710	ARIMA(0, 3, 6) RMSE=2027.707
ARIMA(1, 0, 0) RMSE=4634.577	ARIMA(0, 2, 2) RMSE=35374.745	ARIMA(1, 0, 0) RMSE=2790.248
ARIMA(1, 0, 1) RMSE=4463.534	ARIMA(0, 2, 3) RMSE=35693.897	ARIMA(1, 0, 1) RMSE=2309.573
ARIMA(1, 0, 2) RMSE=4433.572	ARIMA(0, 2, 4) RMSE=37562.577	ARIMA(1, 0, 2) RMSE=1716.448
ARIMA(1, 0, 3) RMSE=4367.620	ARIMA(0, 2, 5) RMSE=29622.891	ARIMA(1, 0, 3) RMSE=1711.851
ARIMA(1, 0, 4) RMSE=4279.633	ARIMA(0, 2, 6) RMSE=29667.443	ARIMA(1, 0, 4) RMSE=1745.707
ARIMA(1, 0, 5) RMSE=4326.883	ARIMA(0, 2, 7) RMSE=29481.432	ARIMA(1, 0, 5) RMSE=1765.377
ARIMA(1, 0, 6) RMSE=4088.991	ARIMA(0, 2, 8) RMSE=29313.254	ARIMA(1, 0, 6) RMSE=1781.548
ARIMA(1, 1, 0) RMSE=4458.212	ARIMA(0, 2, 9) RMSE=29802.371	ARIMA(1, 1, 0) RMSE=2312.669
ARIMA(1, 1, 1) RMSE=4164.155	ARIMA(0, 2, 10) RMSE=30642.641	ARIMA(1, 1, 1) RMSE=2297.091
ARIMA(1, 1, 2) RMSE=4189.817	ARIMA(0, 2, 11) RMSE=30763.977	ARIMA(1, 1, 2) RMSE=2265.280
ARIMA(1, 1, 3) RMSE=4187.779	ARIMA(0, 2, 12) RMSE=31002.741	ARIMA(1, 1, 3) RMSE=2220.982
ARIMA(1, 1, 4) RMSE=4154.217	ARIMA(0, 3, 0) RMSE=167162.089	ARIMA(1, 1, 4) RMSE=2196.728
ARIMA(1, 1, 5) RMSE=4081.730	ARIMA(0, 3, 1) RMSE=88127.579	ARIMA(1, 1, 5) RMSE=2219.340
ARIMA(1, 1, 6) RMSE=4175.439	ARIMA(0, 3, 2) RMSE=50217.836	ARIMA(1, 1, 6) RMSE=2140.092
ARIMA(1, 2, 0) RMSE=4791.932	ARIMA(0, 3, 3) RMSE=38301.653	ARIMA(1, 2, 0) RMSE=2172.823
ARIMA(1, 2, 1) RMSE=4308.868	ARIMA(0, 3, 4) RMSE=45119.183	ARIMA(1, 2, 1) RMSE=2050.911
ARIMA(1, 2, 2) RMSE=4295.696	ARIMA(0, 3, 5) RMSE=38023.676	ARIMA(1, 2, 2) RMSE=2065.877
ARIMA(1, 2, 3) RMSE=4403.192	ARIMA(0, 3, 6) RMSE=31292.076	ARIMA(1, 2, 3) RMSE=1916.759
ARIMA(1, 2, 4) RMSE=4390.757	ARIMA(0, 3, 7) RMSE=31438.823	ARIMA(1, 2, 4) RMSE=1915.549
ARIMA(1, 2, 5) RMSE=4283.961	ARIMA(0, 3, 8) RMSE=31472.934	ARIMA(1, 2, 5) RMSE=1911.776
ARIMA(1, 2, 6) RMSE=4329.991	ARIMA(0, 3, 9) RMSE=32308.144	ARIMA(1, 2, 6) RMSE=1897.427
ARIMA(1, 3, 0) RMSE=7137.109	ARIMA(0, 3, 10) RMSE=33906.751	ARIMA(1, 3, 0) RMSE=2779.376
ARIMA(1, 3, 1) RMSE=4619.227	ARIMA(0, 3, 11) RMSE=33450.813	ARIMA(1, 3, 1) RMSE=2239.316
ARIMA(1, 3, 2) RMSE=4322.520	ARIMA(0, 3, 12) RMSE=33837.102	ARIMA(1, 3, 2) RMSE=2169.884
ARIMA(1, 3, 3) RMSE=4327.522	ARIMA(1, 0, 0) RMSE=35120.232	ARIMA(1, 3, 3) RMSE=2205.358
ARIMA(1, 3, 4) RMSE=4335.961	ARIMA(1, 0, 1) RMSE=35323.307	ARIMA(1, 3, 4) RMSE=2019.016
ARIMA(1, 3, 5) RMSE=4334.264	ARIMA(1, 0, 2) RMSE=30467.998	ARIMA(1, 3, 5) RMSE=2000.957
ARIMA(1, 3, 6) RMSE=4297.384	ARIMA(1, 0, 3) RMSE=28782.141	ARIMA(1, 3, 6) RMSE=2009.263
ARIMA(2, 0, 0) RMSE=4367.603	ARIMA(1, 0, 4) RMSE=29010.963	ARIMA(2, 0, 0) RMSE=2024.678
ARIMA(2, 0, 2) RMSE=26135.196	ARIMA(1, 0, 5) RMSE=28789.182	ARIMA(2, 0, 2) RMSE=1727.786
ARIMA(2, 0, 3) RMSE=4233.084	ARIMA(1, 0, 6) RMSE=29052.998	ARIMA(2, 0, 3) RMSE=1730.415
ARIMA(2, 0, 4) RMSE=4199.906	ARIMA(1, 0, 7) RMSE=29193.089	ARIMA(2, 0, 4) RMSE=1682.468
ARIMA(2, 0, 5) RMSE=4188.681	ARIMA(1, 0, 8) RMSE=29326.597	ARIMA(2, 0, 5) RMSE=1698.301
ARIMA(2, 0, 6) RMSE=4152.761	ARIMA(1, 0, 9) RMSE=29315.809	ARIMA(2, 0, 6) RMSE=1713.194
ARIMA(2, 1, 0) RMSE=4345.277	ARIMA(1, 0, 10) RMSE=29615.421	ARIMA(2, 1, 0) RMSE=2323.322
ARIMA(2, 1, 1) RMSE=4187.712	ARIMA(1, 0, 11) RMSE=29999.188	ARIMA(2, 1, 1) RMSE=2253.258
ARIMA(2, 1, 2) RMSE=4199.498	ARIMA(1, 0, 12) RMSE=29638.970	ARIMA(2, 1, 2) RMSE=2297.345
ARIMA(2, 1, 3) RMSE=4201.337	ARIMA(1, 1, 0) RMSE=39051.822	ARIMA(2, 1, 3) RMSE=2253.600
ARIMA(2, 1, 4) RMSE=3646249.022	ARIMA(1, 1, 1) RMSE=38920.590	ARIMA(2, 1, 4) RMSE=2240.246
ARIMA(2, 1, 6) RMSE=4144.579	ARIMA(1, 1, 2) RMSE=35986.729	ARIMA(2, 1, 5) RMSE=2182.943
ARIMA(2, 2, 0) RMSE=4532.992	ARIMA(1, 1, 3) RMSE=30509.195	ARIMA(2, 1, 6) RMSE=2177.839
ARIMA(2, 2, 1) RMSE=4309.221	ARIMA(1, 1, 4) RMSE=29013.909	ARIMA(2, 2, 0) RMSE=2173.926
ARIMA(2, 2, 2) RMSE=371023.499	ARIMA(1, 1, 5) RMSE=29366.872	ARIMA(2, 2, 1) RMSE=1995.287
ARIMA(2, 2, 3) RMSE=4297.661	ARIMA(1, 1, 6) RMSE=29094.773	ARIMA(2, 2, 2) RMSE=1995.623
ARIMA(2, 2, 4) RMSE=4291.310	ARIMA(1, 1, 7) RMSE=29399.790	ARIMA(2, 2, 3) RMSE=1927.954
ARIMA(2, 2, 5) RMSE=4025.468	ARIMA(1, 1, 8) RMSE=29069.537	ARIMA(2, 2, 4) RMSE=1911.926
ARIMA(2, 2, 6) RMSE=4126.891	ARIMA(1, 1, 9) RMSE=29374.479	ARIMA(2, 2, 5) RMSE=1901.976
ARIMA(2, 3, 0) RMSE=6075.616	ARIMA(1, 1, 10) RMSE=29539.967	ARIMA(2, 2, 6) RMSE=1921.859
ARIMA(2, 3, 1) RMSE=4447.204	ARIMA(1, 1, 11) RMSE=29728.057	ARIMA(2, 3, 0) RMSE=2847.092

```
ARIMA(2, 3, 2) RMSE=4265.125
ARIMA(2, 3, 3) RMSE=4238.151
ARIMA(2, 3, 4) RMSE=4311.125
ARIMA(2, 3, 5) RMSE=243864207889.0
20
ARIMA(2, 3, 6) RMSE=4242.612
ARIMA(3, 0, 1) RMSE=21416.647
ARIMA(3, 0, 2) RMSE=4282.072
ARIMA(3, 0, 3) RMSE=4476.959
ARIMA(3, 0, 4) RMSE=4266.966
ARIMA(3, 0, 5) RMSE=4157.581
ARIMA(3, 0, 6) RMSE=4329.861
ARIMA(3, 1, 0) RMSE=4265.297
ARIMA(3, 1, 1) RMSE=4185.542
ARIMA(3, 1, 2) RMSE=4205.743
ARIMA(3, 1, 3) RMSE=4084.238
ARIMA(3, 1, 4) RMSE=4129.957
ARIMA(3, 1, 5) RMSE=4093.228
ARIMA(3, 1, 6) RMSE=4119.524
ARIMA(3, 2, 0) RMSE=4408.914
ARIMA(3, 2, 1) RMSE=4275.123
ARIMA(3, 2, 2) RMSE=4323.785
ARIMA(3, 2, 3) RMSE=4307.359
ARIMA(3, 2, 4) RMSE=4180.617
ARIMA(3, 2, 5) RMSE=4238.368
ARIMA(3, 2, 6) RMSE=4157.450
ARIMA(3, 3, 0) RMSE=5358.036
ARIMA(3, 3, 1) RMSE=4357.096
ARIMA(3, 3, 2) RMSE=4323.041
ARIMA(3, 3, 3) RMSE=4326.182
ARIMA(3, 3, 4) RMSE=4305.726
ARIMA(3, 3, 5) RMSE=4236.355
ARIMA(3, 3, 6) RMSE=52328.625
ARIMA(4, 0, 1) RMSE=4095.688
ARIMA(4, 0, 2) RMSE=4338.428
ARIMA(4, 0, 3) RMSE=52480.253
ARIMA(4, 0, 4) RMSE=4390.346
ARIMA(4, 0, 5) RMSE=4269.755
ARIMA(4, 0, 6) RMSE=4369.679
ARIMA(4, 1, 0) RMSE=4211.750
ARIMA(4, 1, 1) RMSE=4150.925
ARIMA(4, 1, 2) RMSE=4213.104
ARIMA(4, 1, 3) RMSE=4134.920
ARIMA(4, 1, 5) RMSE=4071.183
ARIMA(4, 1, 6) RMSE=4134.102
ARIMA(4, 2, 0) RMSE=4407.685
ARIMA(4, 2, 1) RMSE=4267.403
ARIMA(4, 2, 2) RMSE=4278.988
ARIMA(4, 2, 3) RMSE=4257.633
ARIMA(4, 2, 4) RMSE=4145.173
ARIMA(4, 2, 5) RMSE=4133.716
ARIMA(4, 3, 0) RMSE=5145.132
ARIMA(4, 3, 1) RMSE=4355.288
ARIMA(4, 3, 2) RMSE=4362.620
ARIMA(4, 3, 3) RMSE=4358.490
ARIMA(4, 3, 4) RMSE=4180.290
ARIMA(4, 3, 5) RMSE=4280.375
ARIMA(4, 3, 6) RMSE=4022.147
Best ARIMA(4, 3, 6) RMSE=4022.147
```

```
ARIMA(1, 1, 12) RMSE=30012.293
ARIMA(1, 2, 0) RMSE=51224.754
ARIMA(1, 2, 1) RMSE=39217.860
ARIMA(1, 2, 2) RMSE=39219.883
ARIMA(1, 2, 3) RMSE=151995.299
ARIMA(1, 2, 4) RMSE=30965.446
ARIMA(1, 2, 5) RMSE=29528.199
ARIMA(1, 2, 6) RMSE=29399.004
ARIMA(1, 2, 7) RMSE=29144.086
ARIMA(1, 2, 8) RMSE=29554.531
ARIMA(1, 2, 9) RMSE=29273.018
ARIMA(1, 2, 10) RMSE=29750.250
ARIMA(1, 2, 11) RMSE=30098.363
ARIMA(1, 2, 12) RMSE=30305.520
ARIMA(1, 3, 0) RMSE=78338.368
ARIMA(1, 3, 1) RMSE=51377.635
ARIMA(1, 3, 2) RMSE=39753.996
ARIMA(1, 3, 3) RMSE=4443964.890
ARIMA(1, 3, 4) RMSE=40361.613
ARIMA(1, 3, 5) RMSE=32251.344
ARIMA(1, 3, 6) RMSE=36218.354
ARIMA(1, 3, 7) RMSE=29908.524
ARIMA(1, 3, 8) RMSE=29728.790
ARIMA(1, 3, 9) RMSE=29847.063
ARIMA(1, 3, 10) RMSE=30084.528
ARIMA(1, 3, 11) RMSE=30120.999
ARIMA(1, 3, 12) RMSE=30549.257
ARIMA(2, 0, 0) RMSE=34410.167
ARIMA(2, 0, 1) RMSE=33542.400
ARIMA(2, 0, 2) RMSE=29026.877
ARIMA(2, 0, 3) RMSE=29015.736
ARIMA(2, 0, 4) RMSE=29258.805
ARIMA(2, 0, 5) RMSE=29344.023
ARIMA(2, 0, 6) RMSE=29923.973
ARIMA(2, 0, 7) RMSE=29295.489
ARIMA(2, 0, 8) RMSE=29059.617
ARIMA(2, 0, 9) RMSE=29398.504
ARIMA(2, 0, 10) RMSE=29394.124
ARIMA(2, 0, 11) RMSE=29788.777
ARIMA(2, 0, 12) RMSE=29977.528
ARIMA(2, 1, 0) RMSE=39315.363
ARIMA(2, 1, 1) RMSE=34472.468
ARIMA(2, 1, 2) RMSE=33511.524
ARIMA(2, 1, 3) RMSE=29114.891
ARIMA(2, 1, 4) RMSE=29180.030
ARIMA(2, 1, 5) RMSE=29200.008
ARIMA(2, 1, 6) RMSE=29330.264
ARIMA(2, 1, 7) RMSE=28855.833
ARIMA(2, 1, 8) RMSE=29359.863
ARIMA(2, 1, 9) RMSE=29380.705
ARIMA(2, 1, 10) RMSE=29651.989
ARIMA(2, 1, 11) RMSE=29789.100
ARIMA(2, 1, 12) RMSE=29745.619
ARIMA(2, 2, 0) RMSE=50028.715
ARIMA(2, 2, 1) RMSE=39432.070
ARIMA(2, 2, 2) RMSE=39494.137
ARIMA(2, 2, 3) RMSE=38014.679
ARIMA(2, 2, 4) RMSE=30040.749
ARIMA(2, 2, 6) RMSE=29543.938
…………..
ARIMA(4, 0, 0) RMSE=32506.941
ARIMA(4, 0, 1) RMSE=32622.584
ARIMA(4, 0, 2) RMSE=29659.147
ARIMA(4, 0, 3) RMSE=29193.383
ARIMA(4, 0, 4) RMSE=29370.520
ARIMA(4, 0, 5) RMSE=29802.447
ARIMA(4, 0, 6) RMSE=28671.593
ARIMA(4, 0, 7) RMSE=29183.365
ARIMA(4, 0, 8) RMSE=29664.484
ARIMA(4, 0, 9) RMSE=29607.598
ARIMA(4, 0, 11) RMSE=29881.979
ARIMA(4, 0, 12) RMSE=29797.807
…………………..
Best ARIMA(4, 0, 6) RMSE= 28671.59
```

```
ARIMA(2, 3, 1) RMSE=2241.963
ARIMA(2, 3, 2) RMSE=2240.758
ARIMA(2, 3, 3) RMSE=2191.386
ARIMA(2, 3, 4) RMSE=2032.541
ARIMA(2, 3, 5) RMSE=2002.916
ARIMA(2, 3, 6) RMSE=71868071405.04
4
ARIMA(3, 0, 1) RMSE=2065.922
ARIMA(3, 0, 2) RMSE=1757.198
ARIMA(3, 0, 3) RMSE=1690.703
ARIMA(3, 0, 4) RMSE=1679.888
ARIMA(3, 0, 6) RMSE=1691.836
ARIMA(3, 1, 0) RMSE=2347.270
ARIMA(3, 1, 1) RMSE=2237.681
ARIMA(3, 1, 2) RMSE=2257.831
ARIMA(3, 1, 3) RMSE=2209.852
ARIMA(3, 1, 4) RMSE=2163.607
ARIMA(3, 1, 5) RMSE=2160.236
ARIMA(3, 1, 6) RMSE=2159.932
ARIMA(3, 2, 0) RMSE=1990.969
ARIMA(3, 2, 1) RMSE=1966.969
ARIMA(3, 2, 2) RMSE=1987.244
ARIMA(3, 2, 3) RMSE=1942.847
ARIMA(3, 2, 4) RMSE=1911.807
ARIMA(3, 2, 5) RMSE=1921.675
ARIMA(3, 2, 6) RMSE=1907.475
ARIMA(3, 3, 0) RMSE=2653.250
ARIMA(3, 3, 1) RMSE=2104.693
ARIMA(3, 3, 2) RMSE=2062.412
ARIMA(3, 3, 3) RMSE=2079.097
ARIMA(3, 3, 4) RMSE=2022.793
ARIMA(3, 3, 5) RMSE=2021.824
ARIMA(3, 3, 6) RMSE=1998.484
ARIMA(4, 0, 1) RMSE=2025.614
ARIMA(4, 0, 3) RMSE=1686.257
ARIMA(4, 0, 4) RMSE=1678.035
ARIMA(4, 0, 5) RMSE=1710.875
ARIMA(4, 0, 6) RMSE=1665.131
ARIMA(4, 1, 0) RMSE=2278.693
ARIMA(4, 1, 1) RMSE=2217.299
ARIMA(4, 1, 2) RMSE=2239.245
ARIMA(4, 1, 3) RMSE=2199.543
ARIMA(4, 1, 4) RMSE=2168.950
ARIMA(4, 1, 5) RMSE=2166.594
ARIMA(4, 1, 6) RMSE=2162.877
ARIMA(4, 2, 0) RMSE=1955.444
ARIMA(4, 2, 1) RMSE=1952.034
ARIMA(4, 2, 2) RMSE=1942.024
ARIMA(4, 2, 3) RMSE=1919.535
ARIMA(4, 2, 4) RMSE=1932.384
ARIMA(4, 2, 5) RMSE=1897.980
ARIMA(4, 2, 6) RMSE=1937.657
ARIMA(4, 3, 0) RMSE=2431.084
ARIMA(4, 3, 1) RMSE=2056.074
ARIMA(4, 3, 2) RMSE=2071.839
ARIMA(4, 3, 3) RMSE=2033.184
ARIMA(4, 3, 4) RMSE=2029.948
ARIMA(4, 3, 5) RMSE=2005.498
ARIMA(4, 3, 6) RMSE=2007.320
Best ARIMA(4, 0, 6) RMSE=1665.131
```

Table 18: Grid Search Result for ARIMA hyperparameter

7.1.2 VAR (p) selection

London	**Kensington & Chelsea**	**Barking & Dagenham**
Lag Order = 1	Lag Order = 1	Lag Order = 1
AIC : 47.58050592736312	AIC : 38.11347956501855	AIC : 32.08124659862008
BIC : 47.780498032666955	**BIC : 38.313471670322386**	**BIC : 32.28123870392392**
FPE : 4.6127343412966995e+20	FPE : 3.56846247377955e+16	FPE : 85647687842349.27
HQIC: 47.661464223466595	HQIC: 38.194437861122026	HQIC: 32.16220489472356
Lag Order = 2	Lag Order = 2	Lag Order = 2
AIC : 47.54695783551394	AIC : 38.115835929765424	AIC : 31.933211307901267
BIC : 47.89818440614578	BIC : 38.46706250039726	BIC : 32.284437878533105
FPE : 4.460884894960566e+20	FPE : 3.5771472296476024e+16	FPE : 73868122107584.48
HQIC: 47.68915118447908	HQIC: 38.25802927873056	**HQIC: 32.075404656866404**
Lag Order = 3	Lag Order = 3	Lag Order = 3
AIC : 47.47108764377704	AIC : 37.939078294858376	AIC : 31.901428975787574
BIC : 47.97462603740223	BIC : 38.44261668848357	BIC : 32.404967369412766
FPE : 4.1356972607521784e+20	FPE : 2.998124604749405e+16	FPE : 71570144757375.3
HQIC: 47.67496437510996	**HQIC: 38.142955026191295**	HQIC: 32.10530570712049
Lag Order = 4	**Lag Order = 4**	**Lag Order = 4**
AIC : 47.442757698943915	**AIC : 37.936762191007666**	**AIC : 31.88557334296802**
BIC : 48.099698256693756	BIC : 38.59370274875751	BIC : 32.542513900717864
FPE : 4.021517300649006e+20	**FPE : 2.992185729135487e+16**	FPE : 70467785164754.73
HQIC: 47.7087714152527	HQIC: 38.20277590731645	HQIC: 32.15158705927681
Lag Order = 5	Lag Order = 5	Lag Order = 5
AIC : 47.48769823694714	AIC : 37.96355810451341	AIC : 31.913542237951646
BIC : 48.29914450154524	BIC : 38.775004369111514	BIC : 32.72498850254975
FPE : 4.208644434599336e+20	FPE : 3.075110539313958e+16	FPE : 72505712638566.19
HQIC: 47.816307897679955	HQIC: 38.29216776524623	HQIC: 32.24215189868446
Lag Order = 6	Lag Order = 6	Lag Order = 6
AIC : 47.31630011516569	AIC : 37.95974229975819	AIC : 31.923763043946945
BIC : 48.283369053736436	BIC : 38.92681123832894	BIC : 32.890831982517696
FPE : 3.548581237580453e+20	FPE : 3.06586708668748e+16	FPE : 73309598241610.16
HQIC: 47.7079701219886	HQIC: 38.3514123065811	HQIC: 32.31543305076986
Lag Order = 7	Lag Order = 7	Lag Order = 7
AIC : 47.2842589536842	AIC : 37.93962271123499	AIC : 31.877149328737968
BIC : 48.40808118536524	BIC : 39.06344494291603	BIC : 33.000971560419
FPE : 3.440572801964094e+20	FPE : 3.008200619720997e+16	**FPE : 70049987106995.12**
HQIC: 47.73945923753565	HQIC: 38.394822995086436	HQIC: 32.33234961258941
Lag Order = 8	Lag Order = 8	Lag Order = 8
AIC : 47.29035342484404	AIC : 37.9791549893116	AIC : 31.921206560484066
BIC : 48.57207345332844	BIC : 39.260875017796	BIC : 33.20292658896847
FPE : 3.4668816966447025e+20	FPE : 3.1342737228004948e+16	FPE : 73316773994985.53
HQIC: 47.80955953470384	HQIC: 38.4983610991714	HQIC: 32.44041267034386
Lag Order = 9	Lag Order = 9	Lag Order = 9
AIC : 46.984431327860676	AIC : 38.00257813364342	AIC : 31.941366745588518
BIC : 48.4252077789983	BIC : 39.44335458478104	BIC : 33.38214319672613
FPE : **2.5582400580820084e+20**	FPE : 3.2149336586071748e+16	FPE : 74958580908860.17
HQIC: **47.568124521272345**	HQIC: 38.58627132705509	HQIC: 32.52505993900018

Table 19 : Lag Order Result for Identification of AIC

7.1.3 Summary of Algorithm Results

```
                               SARIMAX Results                                
==============================================================================
Dep. Variable:                      y   No. Observations:                  314
Model:                 ARIMA(4, 3, 6)   Log Likelihood               -2986.818
Date:                Wed, 01 Sep 2021   AIC                           5995.637
Time:                        01:33:44   BIC                           6036.775
Sample:                             0   HQIC                          6012.080
                                - 314                                         
Covariance Type:                  opg                                         
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
ar.L1         -2.6519      0.146    -18.204      0.000      -2.937      -2.366
ar.L2         -3.5686      0.269    -13.255      0.000      -4.096      -3.041
ar.L3         -2.5642      0.269     -9.525      0.000      -3.092      -2.037
ar.L4         -0.9167      0.138     -6.651      0.000      -1.187      -0.647
ma.L1          0.9346      0.189      4.950      0.000       0.565       1.305
ma.L2         -0.2180      0.188     -1.161      0.246      -0.586       0.150
ma.L3         -1.6142      0.164     -9.818      0.000      -1.936      -1.292
ma.L4         -0.9323      0.171     -5.441      0.000      -1.268      -0.596
ma.L5          0.2065      0.121      1.701      0.089      -0.031       0.444
ma.L6          0.6235      0.119      5.224      0.000       0.390       0.857
sigma2      1.401e+07   5.13e-08   2.73e+14      0.000     1.4e+07     1.4e+07
===================================================================================
Ljung-Box (L1) (Q):                   2.12   Jarque-Bera (JB):                14.70
Prob(Q):                              0.15   Prob(JB):                         0.00
Heteroskedasticity (H):               4.04   Skew:                            -0.02
Prob(H) (two-sided):                  0.00   Kurtosis:                         4.06
===================================================================================

Warnings:
[1] Covariance matrix calculated using the outer product of gradients (complex-step).
[2] Covariance matrix is singular or near-singular, with condition number 9.01e+29. Standard erro
rs may be unstable.
```

Table 20 : ARIMA (4. 3. 6) Summary of Result for London

```
                               SARIMAX Results                                
==============================================================================
Dep. Variable:                      y   No. Observations:                  314
Model:                 ARIMA(4, 0, 6)   Log Likelihood               -3623.337
Date:                Fri, 27 Aug 2021   AIC                           7270.675
Time:                        03:48:03   BIC                           7315.667
Sample:                             0   HQIC                          7288.653
                                - 314                                         
Covariance Type:                  opg                                         
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
const       7.663e+05   1.37e-09   5.59e+14      0.000    7.66e+05    7.66e+05
ar.L1          1.4542      0.105     13.863      0.000       1.249       1.660
ar.L2          0.0692      0.109      0.634      0.526      -0.145       0.283
ar.L3         -1.4170      0.104    -13.560      0.000      -1.622      -1.212
ar.L4          0.8932      0.090      9.940      0.000       0.717       1.069
ma.L1         -0.3420      0.098     -3.504      0.000      -0.533      -0.151
ma.L2         -0.3016      0.078     -3.854      0.000      -0.455      -0.148
ma.L3          0.3330      0.096      3.465      0.001       0.145       0.521
ma.L4          0.1354      0.077      1.752      0.080      -0.016       0.287
ma.L5          0.3486      0.068      5.140      0.000       0.216       0.482
ma.L6         -0.3717      0.055     -6.788      0.000      -0.479      -0.264
sigma2      6.617e+08   8.06e-11   8.21e+18      0.000    6.62e+08    6.62e+08
===================================================================================
Ljung-Box (L1) (Q):                   0.77   Jarque-Bera (JB):               126.59
Prob(Q):                              0.38   Prob(JB):                         0.00
Heteroskedasticity (H):              17.40   Skew:                             0.06
Prob(H) (two-sided):                  0.00   Kurtosis:                         6.11
===================================================================================

Warnings:
[1] Covariance matrix calculated using the outer product of gradients (complex-step).
[2] Covariance matrix is singular or near-singular, with condition number 3.09e+35. Standard erro
rs may be unstable
```

Table 21: ARIMA (4,0,6) Summary of Result for Kensington & Chelsea

```
                               SARIMAX Results
==============================================================================
Dep. Variable:                      y   No. Observations:                  314
Model:                 ARIMA(4, 0, 6)   Log Likelihood               -2772.995
Date:                Wed, 01 Sep 2021   AIC                           5569.990
Time:                        01:19:49   BIC                           5614.983
Sample:                             0   HQIC                          5587.968
                                - 314
Covariance Type:                  opg
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
const       1.658e+05   5.58e+04      2.971      0.003    5.64e+04    2.75e+05
ar.L1          0.2938      0.309      0.952      0.341      -0.311       0.899
ar.L2          0.1713      0.161      1.065      0.287      -0.144       0.487
ar.L3          0.5393      0.156      3.467      0.001       0.234       0.844
ar.L4         -0.0081      0.269     -0.030      0.976      -0.535       0.519
ma.L1          1.4002      0.309      4.538      0.000       0.795       2.005
ma.L2          1.6434      0.468      3.508      0.000       0.725       2.562
ma.L3          0.7080      0.615      1.152      0.250      -0.497       1.913
ma.L4          0.4430      0.325      1.365      0.172      -0.193       1.079
ma.L5          0.1917      0.177      1.085      0.278      -0.155       0.538
ma.L6          0.2218      0.077      2.865      0.004       0.070       0.374
sigma2      2.703e+06   4655.060    580.756      0.000    2.69e+06    2.71e+06
===================================================================================
Ljung-Box (L1) (Q):                   0.60   Jarque-Bera (JB):                60.92
Prob(Q):                              0.44   Prob(JB):                         0.00
Heteroskedasticity (H):               5.90   Skew:                             0.12
Prob(H) (two-sided):                  0.00   Kurtosis:                         5.14
===================================================================================

Warnings:
[1] Covariance matrix calculated using the outer product of gradients (complex-step).
[2] Covariance matrix is singular or near-singular, with condition number 4.72e+17. Standard errors may
be unstable.
```

Table 22 : ARIMA (4,0,6) Summary of Result for Barking & Dagenham

```
Summary of Regression Results
==================================
Model:                         VAR
Method:                        OLS
Date:           Mon, 30, Aug, 2021
Time:                     22:52:09
--------------------------------------------------------------------
No. of Equations:         3.00000    BIC:                    48.0997
Nobs:                     194.000    HQIC:                   47.7088
Log likelihood:          -5388.77    FPE:                4.02152e+20
AIC:                      47.4428    Det(Omega_mle):     3.31043e+20
--------------------------------------------------------------------
Results for equation average_price
===================================================================================
                      coefficient       std. error           t-stat            prob
-----------------------------------------------------------------------------------
const                  446.667371       208.223041            2.145           0.032
L1.average_price         0.106525         0.077379            1.377           0.169
L1.houses_sold           0.549618         0.130950            4.197           0.000
L1.no_of_crimes         -0.029567         0.031812           -0.929           0.353
L2.average_price         0.146565         0.074723            1.961           0.050
L2.houses_sold           0.099181         0.132810            0.747           0.455
L2.no_of_crimes          0.020958         0.031918            0.657           0.511
L3.average_price         0.374300         0.080421            4.654           0.000
L3.houses_sold          -0.199431         0.129807           -1.536           0.124
L3.no_of_crimes          0.013964         0.032744            0.426           0.670
L4.average_price        -0.025344         0.079427           -0.319           0.750
L4.houses_sold          -0.182736         0.132699           -1.377           0.168
L4.no_of_crimes          0.042542         0.032595            1.305           0.192
===================================================================================

Results for equation houses_sold
===================================================================================
                      coefficient       std. error           t-stat            prob
-----------------------------------------------------------------------------------
const                  -30.440968       114.276896           -0.266           0.790
L1.average_price         0.012332         0.042467            0.290           0.772
L1.houses_sold          -0.014654         0.071868           -0.204           0.838
L1.no_of_crimes          0.006971         0.017459            0.399           0.690
L2.average_price         0.013107         0.041010            0.320           0.749
L2.houses_sold          -0.098286         0.072889           -1.348           0.178
L2.no_of_crimes          0.061321         0.017517            3.501           0.000
L3.average_price         0.129741         0.044137            2.940           0.003
L3.houses_sold           0.161176         0.071241            2.262           0.024
L3.no_of_crimes         -0.021057         0.017971           -1.172           0.241
L4.average_price        -0.162219         0.043591           -3.721           0.000
L4.houses_sold          -0.090012         0.072828           -1.236           0.216
L4.no_of_crimes          0.029682         0.017889            1.659           0.097
===================================================================================

Results for equation no_of_crimes
===================================================================================
                      coefficient       std. error           t-stat            prob
-----------------------------------------------------------------------------------
const                  388.212903       481.588132            0.806           0.420
L1.average_price        -0.113654         0.178965           -0.635           0.525
L1.houses_sold           0.128188         0.302867            0.423           0.672
L1.no_of_crimes         -0.092931         0.073577           -1.263           0.207
L2.average_price         0.298439         0.172824            1.727           0.084
L2.houses_sold          -0.195991         0.307169           -0.638           0.523
L2.no_of_crimes          0.100083         0.073820            1.356           0.175
L3.average_price        -0.022552         0.186002           -0.121           0.903
L3.houses_sold          -0.036168         0.300224           -0.120           0.904
L3.no_of_crimes         -0.041393         0.075732           -0.547           0.585
L4.average_price        -0.212742         0.183703           -1.158           0.247
L4.houses_sold          -0.484855         0.306912           -1.580           0.114
L4.no_of_crimes          0.080687         0.075388            1.070           0.284
===================================================================================

Correlation matrix of residuals
                 average_price  houses_sold  no_of_crimes
average_price         1.000000     0.153098      0.051533
houses_sold           0.153098     1.000000     -0.039047
no_of_crimes          0.051533    -0.039047      1.000000
```

Table 23: Summary of the VAR(4) Model Result for London

```
Summary of Regression Results
==================================
Model:                         VAR
Method:                        OLS
Date:           Fri, 27, Aug, 2021
Time:                     02:38:30
--------------------------------------------------------------------
No. of Equations:         3.00000    BIC:                    38.5937
Nobs:                     194.000    HQIC:                   38.2028
Log likelihood:          -4466.69    FPE:                2.99219e+16
AIC:                      37.9368    Det(Omega_mle):     2.46310e+16
--------------------------------------------------------------------
Results for equation average_price
===================================================================================
                      coefficient       std. error           t-stat            prob
-----------------------------------------------------------------------------------
const                 3329.393661      1249.077000            2.665           0.008
L1.average_price         0.280773         0.073080            3.842           0.000
L1.houses_sold          41.606354        25.133258            1.655           0.098
L1.no_of_crimes         10.335437         5.693939            1.815           0.069
L2.average_price         0.080928         0.068022            1.190           0.234
L2.houses_sold          51.259266        27.186094            1.885           0.059
L2.no_of_crimes          2.519228         5.920548            0.426           0.670
L3.average_price        -0.528388         0.067260           -7.856           0.000
L3.houses_sold          50.625894        27.063368            1.871           0.061
L3.no_of_crimes          1.686010         6.008526            0.281           0.779
L4.average_price         0.243599         0.077244            3.154           0.002
L4.houses_sold          29.555509        24.817946            1.191           0.234
L4.no_of_crimes          1.505928         5.732406            0.263           0.793
===================================================================================

Results for equation houses_sold
===================================================================================
                      coefficient       std. error           t-stat            prob
-----------------------------------------------------------------------------------
const                    1.798229         3.711970            0.484           0.628
L1.average_price        -0.000529         0.000217           -2.436           0.015
L1.houses_sold          -0.375393         0.074690           -5.026           0.000
L1.no_of_crimes          0.012482         0.016921            0.738           0.461
L2.average_price         0.000079         0.000202            0.389           0.698
L2.houses_sold          -0.074330         0.080791           -0.920           0.358
L2.no_of_crimes          0.040035         0.017595            2.275           0.023
L3.average_price        -0.000013         0.000200           -0.067           0.946
L3.houses_sold           0.135514         0.080426            1.685           0.092
L3.no_of_crimes          0.006645         0.017856            0.372           0.710
L4.average_price        -0.000300         0.000230           -1.308           0.191
L4.houses_sold           0.131942         0.073753            1.789           0.074
L4.no_of_crimes          0.005365         0.017035            0.315           0.753
===================================================================================

Results for equation no_of_crimes
===================================================================================
                      coefficient       std. error           t-stat            prob
-----------------------------------------------------------------------------------
const                    9.627831        16.239026            0.593           0.553
L1.average_price         0.001231         0.000950            1.295           0.195
L1.houses_sold           0.192148         0.326753            0.588           0.556
L1.no_of_crimes         -0.294812         0.074026           -3.983           0.000
L2.average_price         0.000645         0.000884            0.729           0.466
L2.houses_sold          -0.062980         0.353442           -0.178           0.859
L2.no_of_crimes          0.116288         0.076972            1.511           0.131
L3.average_price        -0.001800         0.000874           -2.058           0.040
L3.houses_sold           0.004119         0.351846            0.012           0.991
L3.no_of_crimes          0.022068         0.078116            0.283           0.778
L4.average_price         0.000394         0.001004            0.392           0.695
L4.houses_sold           0.155707         0.322654            0.483           0.629
L4.no_of_crimes         -0.112639         0.074526           -1.511           0.131
===================================================================================

Correlation matrix of residuals
                 average_price  houses_sold  no_of_crimes
average_price         1.000000     0.180717     -0.035938
houses_sold           0.180717     1.000000      0.063208
no_of_crimes         -0.035938     0.063208      1.000000
```

Table 24: Summary of the VAR(4) Model Result for Kensington & Chelsea

```
  Summary of Regression Results
==================================
Model:                         VAR
Method:                        OLS
Date:           Wed, 01, Sep, 2021
Time:                     04:35:27
--------------------------------------------------------------------
No. of Equations:         3.00000    BIC:                    32.5425
Nobs:                     194.000    HQIC:                   32.1516
Log likelihood:          -3879.72    FPE:                7.04678e+13
AIC:                      31.8856    Det(Omega_mle):     5.80076e+13
--------------------------------------------------------------------
Results for equation average_price
===================================================================================
                      coefficient       std. error           t-stat            prob
-----------------------------------------------------------------------------------
const                  111.998843        95.144504            1.177           0.239
L1.average_price         0.538495         0.071266            7.556           0.000
L1.houses_sold           6.645145         2.445163            2.718           0.007
L1.no_of_crimes          0.179119         0.532508            0.336           0.737
L2.average_price         0.303654         0.075875            4.002           0.000
L2.houses_sold           4.125571         2.578705            1.600           0.110
L2.no_of_crimes         -0.660724         0.557960           -1.184           0.236
L3.average_price        -0.361830         0.075049           -4.821           0.000
L3.houses_sold           7.529531         2.531645            2.974           0.003
L3.no_of_crimes          0.086831         0.567102            0.153           0.878
L4.average_price         0.304852         0.070151            4.346           0.000
L4.houses_sold           0.439783         2.468777            0.178           0.859
L4.no_of_crimes          0.233240         0.533007            0.438           0.662
===================================================================================

Results for equation houses_sold
===================================================================================
                      coefficient       std. error           t-stat            prob
-----------------------------------------------------------------------------------
const                    0.385071         2.874266            0.134           0.893
L1.average_price         0.000337         0.002153            0.156           0.876
L1.houses_sold          -0.286458         0.073867           -3.878           0.000
L1.no_of_crimes          0.020753         0.016087            1.290           0.197
L2.average_price        -0.000278         0.002292           -0.122           0.903
L2.houses_sold          -0.170766         0.077901           -2.192           0.028
L2.no_of_crimes          0.037826         0.016856            2.244           0.025
L3.average_price         0.000600         0.002267            0.265           0.791
L3.houses_sold           0.079079         0.076480            1.034           0.301
L3.no_of_crimes         -0.005081         0.017132           -0.297           0.767
L4.average_price        -0.002416         0.002119           -1.140           0.254
L4.houses_sold           0.036762         0.074580            0.493           0.622
L4.no_of_crimes          0.003738         0.016102            0.232           0.816
===================================================================================

Results for equation no_of_crimes
===================================================================================
                      coefficient       std. error           t-stat            prob
-----------------------------------------------------------------------------------
const                   15.112199        13.380194            1.129           0.259
L1.average_price        -0.017489         0.010022           -1.745           0.081
L1.houses_sold           0.201101         0.343864            0.585           0.559
L1.no_of_crimes         -0.321308         0.074887           -4.291           0.000
L2.average_price        -0.003878         0.010670           -0.363           0.716
L2.houses_sold           0.312863         0.362644            0.863           0.388
L2.no_of_crimes          0.045124         0.078466            0.575           0.565
L3.average_price         0.018008         0.010554            1.706           0.088
L3.houses_sold           0.018415         0.356026            0.052           0.959
L3.no_of_crimes         -0.020413         0.079752           -0.256           0.798
L4.average_price        -0.003932         0.009865           -0.399           0.690
L4.houses_sold          -0.149333         0.347185           -0.430           0.667
L4.no_of_crimes         -0.019772         0.074957           -0.264           0.792
===================================================================================

Correlation matrix of residuals
                 average_price  houses_sold  no_of_crimes
average_price         1.000000     0.030672      0.133497
houses_sold           0.030672     1.000000     -0.051087
no_of_crimes          0.133497    -0.051087      1.000000
```

Table 25: Summary of the VAR(4) Model Result for Barking & Dagenham

7.1.4 Forecast Result

	average_price_1d	houses_sold_1d	no_of_crimes_1d
2014-04-01	20627.29	-50.25	21.15
2014-05-01	15268.05	-1.03	-22.37
2014-06-01	21341.14	18.93	141.01
2014-07-01	-7374.39	-3.43	-36.23
2014-08-01	-791.42	-0.75	9.49
2014-09-01	-4115	-0.78	-29.89
2014-10-01	11386.3	0.51	21.8
2014-11-01	4768.35	-4.07	13.5
2014-12-01	7524.66	2.64	25.83
2015-01-01	-994.07	-0.6	-1.71
2015-02-01	3935.8	0.2	10.29
2015-03-01	1696.36	-1.8	-2.57
2015-04-01	6500.72	0.4	18.06
2015-05-01	3081.73	-1.15	6.14
2015-06-01	4790.66	0.5	14.96
2015-07-01	2027.63	-0.84	3.36
2015-08-01	4295.77	0.06	10.94
2015-09-01	3025.83	-0.95	5.5
2015-10-01	4670.92	-0.02	12.29
2015-11-01	3203.25	-0.66	7.2
2015-12-01	4110.91	-0.05	10.9
2016-01-01	3106.61	-0.62	6.7
2016-02-01	4044.33	-0.19	10.12
2016-03-01	3413.3	-0.59	7.59
2016-04-01	4039.6	-0.22	10.21
2016-05-01	3444.32	-0.5	7.98
2016-06-01	3873.74	-0.25	9.67
2016-07-01	3476.61	-0.49	8.03
2016-08-01	3854.09	-0.3	9.47
2016-09-01	3566.46	-0.46	8.35
2016-10-01	3821.06	-0.31	9.39
2016-11-01	3581.43	-0.43	8.46
2016-12-01	3771.84	-0.33	9.2
2017-01-01	3607.48	-0.42	8.54
2017-02-01	3759.98	-0.34	9.13
2017-03-01	3634.99	-0.41	8.65
2017-04-01	3741.51	-0.35	9.07
2017-05-01	3644.16	-0.4	8.69
2017-06-01	3725.72	-0.36	9.01
2017-07-01	3656.57	-0.4	8.73
2017-08-01	3718.9	-0.36	8.98
2017-09-01	3665.89	-0.39	8.77
2017-10-01	3710.8	-0.37	8.95
2017-11-01	3670.79	-0.39	8.79
2017-12-01	3705.16	-0.37	8.93
2018-01-01	3675.98	-0.39	8.81
2018-02-01	3701.73	-0.37	8.91
2018-03-01	3679.49	-0.39	8.83
2018-04-01	3698.42	-0.38	8.9
2018-05-01	3681.83	-0.38	8.83
2018-06-01	3696.21	-0.38	8.89
2018-07-01	3683.93	-0.38	8.84
2018-08-01	3694.63	-0.38	8.88
2018-09-01	3685.34	-0.38	8.85
2018-10-01	3693.3	-0.38	8.88
2018-11-01	3686.39	-0.38	8.85
2018-12-01	3692.39	-0.38	8.88
2019-01-01	3687.24	-0.38	8.86
2019-02-01	3691.7	-0.38	8.87
2019-03-01	3687.83	-0.38	8.86
2019-04-01	3691.16	-0.38	8.87
2019-05-01	3688.28	-0.38	8.86
2019-06-01	3690.78	-0.38	8.87
2019-07-01	3688.62	-0.38	8.86
2019-08-01	3690.49	-0.38	8.87
2019-09-01	3688.87	-0.38	8.86
2019-10-01	3690.27	-0.38	8.87
2019-11-01	3689.06	-0.38	8.86
2019-12-01	3690.11	-0.38	8.87
2020-01-01	3689.2	-0.38	8.86
2020-02-01	3689.98	-0.38	8.87

2020-03-01	3689.31	-0.38	8.86
2020-04-01	3689.89	-0.38	8.87
2020-05-01	3689.39	-0.38	8.86
2020-06-01	3689.82	-0.38	8.87
2020-07-01	3689.45	-0.38	8.86
2020-08-01	3689.77	-0.38	8.87
2020-09-01	3689.49	-0.38	8.86
2020-10-01	3689.73	-0.38	8.87
2020-11-01	3689.52	-0.38	8.86
2020-12-01	3689.71	-0.38	8.87
2021-01-01	3689.55	-0.38	8.86
2021-02-01	3689.68	-0.38	8.86
2021-03-01	3689.57	-0.38	8.86

Table 26: Forecast Result for VAR (4) of 1st differencing

	average_price	**houses_sold**	**no_of_crimes**
2014-04-01	863,603.29	252.75	1,694.15
2014-05-01	865,116.63	259.47	1,693.93
2014-06-01	887,971.11	285.11	1,834.72
2014-07-01	903,451.20	307.34	1,939.27
2014-08-01	918,139.87	328.8	2,053.31
2014-09-01	928,713.55	349.49	2,137.46
2014-10-01	950,673.52	370.69	2,243.41
2014-11-01	977,401.84	387.82	2,362.86
2014-12-01	1,011,654.82	407.59	2,508.14
2015-01-01	1,044,913.73	426.75	2,651.71
2015-02-01	1,082,108.44	446.12	2,805.57
2015-03-01	1,120,999.51	463.68	2,956.87
2015-04-01	1,166,391.30	481.65	3,126.23
2015-05-01	1,214,864.82	498.46	3,301.73
2015-06-01	1,268,129.01	515.77	3,492.19
2015-07-01	1,323,420.82	532.25	3,686.01
2015-08-01	1,383,008.40	548.78	3,890.77
2015-09-01	1,445,621.81	564.37	4,101.04
2015-10-01	1,512,906.14	579.94	4,323.59
2015-11-01	1,583,393.72	594.85	4,553.34
2015-12-01	1,657,992.20	609.7	4,794.00
2016-01-01	1,735,697.30	623.94	5,041.35
2016-02-01	1,817,446.72	637.98	5,298.82
2016-03-01	1,902,609.45	651.44	5,563.89
2016-04-01	1,991,811.77	664.68	5,839.17
2016-05-01	2,084,458.42	677.42	6,122.43
2016-06-01	2,180,978.80	689.9	6,415.36
2016-07-01	2,280,975.80	701.89	6,716.32
2016-08-01	2,384,826.88	713.59	7,026.74
2016-09-01	2,492,244.43	724.83	7,345.51
2016-10-01	2,603,483.03	735.75	7,673.67
2016-11-01	2,718,303.06	746.25	8,010.30
2016-12-01	2,836,894.93	756.41	8,356.12
2017-01-01	2,959,094.28	766.15	8,710.48
2017-02-01	3,085,053.61	775.54	9,073.97
2017-03-01	3,214,647.94	784.52	9,446.10
2017-04-01	3,347,983.77	793.15	9,827.31
2017-05-01	3,484,963.76	801.37	10,217.20
2017-06-01	3,625,669.47	809.24	10,616.11
2017-07-01	3,770,031.75	816.7	11,023.75
2017-08-01	3,918,112.92	823.81	11,440.37
2017-09-01	4,069,859.98	830.52	11,865.75
2017-10-01	4,225,317.85	836.86	12,300.09
2017-11-01	4,384,446.51	842.81	12,743.22
2017-12-01	4,547,280.32	848.38	13,195.27
2018-01-01	4,713,790.12	853.57	13,656.14
2018-02-01	4,884,001.65	858.39	14,125.91
2018-03-01	5,057,892.66	862.82	14,604.51
2018-04-01	5,235,482.10	866.88	15,092.02
2018-05-01	5,416,753.37	870.55	15,588.35
2018-06-01	5,601,720.85	873.84	16,093.58
2018-07-01	5,790,372.26	876.75	16,607.64
2018-08-01	5,982,718.30	879.29	17,130.60
2018-09-01	6,178,749.68	881.44	17,662.40
2018-10-01	6,378,474.37	883.21	18,203.08
2018-11-01	6,581,885.45	884.6	18,752.61
2018-12-01	6,788,988.92	885.61	19,311.02

2019-01-01	6,999,779.63	886.25	19,878.28
2019-02-01	7,214,262.04	886.5	20,454.42
2019-03-01	7,432,432.28	886.37	21,039.41
2019-04-01	7,654,293.68	885.86	21,633.27
2019-05-01	7,879,843.36	884.97	22,236.00
2019-06-01	8,109,083.82	883.7	22,847.59
2019-07-01	8,342,012.90	882.05	23,468.04
2019-08-01	8,578,632.47	880.02	24,097.37
2019-09-01	8,818,940.91	877.61	24,735.55
2019-10-01	9,062,939.61	874.82	25,382.60
2019-11-01	9,310,627.38	871.65	26,038.52
2019-12-01	9,562,005.25	868.1	26,703.30
2020-01-01	9,817,072.32	864.16	27,376.94
2020-02-01	10,075,829.38	859.85	28,059.45
2020-03-01	10,338,275.74	855.16	28,750.82
2020-04-01	10,604,411.99	850.09	29,451.06
2020-05-01	10,874,237.63	844.64	30,160.16
2020-06-01	11,147,753.10	838.8	30,878.13
2020-07-01	11,424,958.00	832.59	31,604.96
2020-08-01	11,705,852.68	826	32,340.66
2020-09-01	11,990,436.85	819.02	33,085.22
2020-10-01	12,278,710.76	811.67	33,838.65
2020-11-01	12,570,674.18	803.94	34,600.94
2020-12-01	12,866,327.31	795.82	35,372.10
2021-01-01	13,165,669.99	787.33	36,152.12
2021-02-01	13,468,702.35	778.46	36,941.01
2021-03-01	13,775,424.28	769.2	37,738.76

Table 27: Final Forecast for VAR (4) result (April 2014 - March 2021)

7.1.5 Algorithm design

```
# Load libraries
import numpy as np  # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
import seaborn as sns  # data visualisation
import matplotlib.pyplot as plt  # data visual
isation
%matplotlib inline
import datetime as dt  # working with time data
import plotly.graph_objs as go  # plotly graphical object
import plotly.express as px
from pylab import rcParams
rcParams['figure.figsize'] = 15, 12

from mpl_toolkits.axes_grid1 import host_subplot
import mpl_toolkits.axisartist as AA
import matplotlib.pyplot as plt

import statsmodels.api as sm  # time-series analysis for python
from statsmodels.tsa.stattools import adfuller  # Augmented Dickey-Fuller Test (ADF Test) to check for
stationarity
from statsmodels.tsa.api import VAR  # # Fitting the VAR model to the 2nd Differenced Data
from statsmodels.tsa.vector_ar.var_model import VAR
from statsmodels.tsa.statespace.varmax import VARMAX
from statsmodels.tsa.stattools import grangercausalitytests
from statsmodels.tsa.vector_ar.vecm import coint_johansen
from statsmodels.tools.eval_measures import rmse, aic
import warnings
warnings.filterwarnings('ignore')
import missingno as msno
from fbprophet import Prophet
#import geopandas as gpd
from sklearn.metrics import mean_squared_error  # MSE
from numpy import asarray as arr

# Allows to display all of the outputs of a cell
from IPython.display import display

# Set float data type format
pd.options.display.float_format = '{:,.2f}'.format

# Set the maximum number of row to be displayed
pd.options.display.max_rows = 999

# Set global visualisation settings
```

```
plt.rc('font', size=14)
```
In [20]:
```
# Load dataset

#monthly_data =
pd.read_csv('https://raw.githubusercontent.com/syuqranPSYQ/SmartCitiesHousingForecast/main/data/housing_in_
london_monthly_variables.csv')
file_name =
'https://raw.githubusercontent.com/syuqranPSYQ/SmartCitiesHousingForecast/99920201b354b96d64c26696c6d74d2ba
760ff2b/data/housing_in_london_monthly_variables_march2021.csv'
holdout_data =
pd.read_csv('https://raw.githubusercontent.com/syuqranPSYQ/SmartCitiesHousingForecast/main/data/housing_in_
london_monthly_variables_feb20_march21.csv')
monthly_data = pd.read_csv(file_name, parse_dates = ['date'])
# monthly_data[:10]
print ('The monthly data contains {} rows and {} columns.'.format(monthly_data.shape[0],
monthly_data.shape[1]))
#monthly_data.()
#print ('This holdout data contains {} rows and {} columns.'.format(holdout_data.shape[0],
holdout_data.shape[1]))
#holdout_data.head()
```
```
The monthly data contains 14175 rows and 7 columns.
```
In [21]:
```
monthly_data.head()
```
Out[21]:
In [22]:
```
# fraction values that are not null
monthly_data.notnull().sum()/len(monthly_data)
monthly_data.info()
```
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 14175 entries, 0 to 14174
Data columns (total 7 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   date           14175 non-null  datetime64[ns]
 1   area           14175 non-null  object
 2   average_price  14175 non-null  float64
 3   code           14175 non-null  object
 4   houses_sold    14083 non-null  object
 5   no_of_crimes   7776 non-null   float64
 6   borough_flag   14175 non-null  int64
dtypes: datetime64[ns](1), float64(2), int64(1), object(3)
memory usage: 775.3+ KB
```

Data Preprocess

In [23]:
```
display(monthly_data.describe());
msno.matrix(monthly_data);
```
In [24]:
```
# Set date as index for easier manipulation
monthly_data = monthly_data.set_index(pd.to_datetime(monthly_data['date']))

del monthly_data['date']

# Create dataset cuts
london = monthly_data[monthly_data['area'] == 'london']
london_boroughs = monthly_data[monthly_data['borough_flag'] == 1]
england = monthly_data[monthly_data['area'] == 'england']
north_east = monthly_data[monthly_data['area'] == 'north east']
south_west = monthly_data[monthly_data['area'] == 'south west']
london_expensive = [monthly_data['area'] == 'london']

# Calculate mean prices for the different cuts of data
london_mean_price = london_boroughs.groupby('date')['average_price'].mean()
england_mean_price = england.groupby('date')['average_price'].mean()
north_east_mean_price = north_east.groupby('date')['average_price'].mean()
south_west_mean_price = south_west.groupby('date')['average_price'].mean()


print('Processing Complete')
```
```
Processing Complete
```

Data Exploration

In [25]:
```
#data breakdown
monthly_data[monthly_data['borough_flag'] == 0]['area'].unique()
```
Out[25]:
```
array(['inner london', 'outer london', 'north east', 'north west',
       'yorks and the humber', 'east midlands', 'west midlands',
       'east of england', 'london', 'south east', 'south west', 'england'],
      dtype=object)
```
In [26]:
```
london_mean_price_area = london_boroughs.groupby('area')['average_price'].mean()
```

```
london_top10 = london_mean_price_area.sort_values(ascending = False).to_frame()
london_bottom10 = london_mean_price_area.sort_values(ascending = True).to_frame()
#lnd_b_prices = lnd.groupby('area')['average_price'].mean()
#lnd_top10_pr = lnd_b_prices.sort_values(ascending = False).to_frame()
print ('\nThe 10 most expensive boroughs in London are:')
london_top10.head(10)
```

```
The 10 most expensive boroughs in London are:
```

Out[26]:

In [27]:

```
print ('\nThe 10 cheapest boroughs in London are:')
london_bottom10.head(10)
```

```
The 10 cheapest boroughs in London are:
```

Out[27]:

In [28]:

```
#comparing England and london

fig = go.Figure()
fig.add_trace(go.Scatter(x=london_mean_price.index,
                         y=london_mean_price.values,
                         mode='lines',
                         name='London Mean House Price',
                        ))
fig.add_trace(go.Scatter(x=england_mean_price.index,
                         y=england_mean_price.values,
                         mode='lines',
                         name='England Mean House Price',
                        ))
fig.update_xaxes(
        tickangle = 90,
        title_font = {"size": 20},
        title_standoff = 25)
fig.update_xaxes(nticks=30)
fig.update_yaxes(nticks=10)
fig.update_layout(
    template='ggplot2',
    height=700,
    title='Average Monthly House Price',
    xaxis_title='Year',
    yaxis_title='Price (£)',
    xaxis_showgrid=True,
    yaxis_showgrid=True,
    legend=dict(y=-.3, orientation='h'),
    shapes=[
        dict(
            type="rect",
            x0='2016-06-01',
            y0=0,
            x1='2016-7-01',
            y1=london_mean_price.values.max()*1.2,
            fillcolor="yellow",
            opacity=0.5,
            layer="below",
            line_width=0,
        ),
        dict(
            type="rect",
            x0="2007-12-01",
            y0=0,
            x1="2009-06-01",
            y1=london_mean_price.values.max()*1.2,
            fillcolor="yellow",
            opacity=0.5,
            layer="below",
            line_width=0,
        ),
        dict(
            type="rect",
            x0="2001-03-01",
            y0=0,
            x1="2001-11-01",
            y1=london_mean_price.values.max()*1.2,
            fillcolor="yellow",
            opacity=0.5,
            layer="below",
            line_width=0,
        ),
        dict(
            type="rect",
            x0="2020-03-01",
            y0=0,
            x1="2021-01-01",
            y1=london_mean_price.values.max()*1.2,
            fillcolor="yellow",
            opacity=0.5,
            layer="below",
            line_width=0,
        )
    ],
    annotations=[
            dict(text="The 2008 Recession", x='2007-12-01', y=london_mean_price.values.max()*1.2),
            dict(text="Brexit Vote", x='2016-06-23', y=london_mean_price.values.max()*1.2),
            dict(text="Dot-Com Bubble Recession", x='2001-03-01', y=london_mean_price.values.max()*1.2),
            dict(text="Pandemic", x='2020-03-01', y=london_mean_price.values.max()*1.2)
    ]
)
fig.show()
```

In [29]:

```
fig = px.line(london_boroughs, x=london_boroughs.index, y="average_price", color='area')

fig.update_xaxes(
        tickangle = 90,
        title_font = {"size": 20},
        title_standoff = 25,
        nticks=30
)
fig.update_yaxes(nticks=20)
fig.update_layout(
    template='gridon',
    height=750,
    width =1000,
    title='Average Monthly London House Price by Borough',
    xaxis_title='Year',
    yaxis_title='Price (£)',
    legend=dict(y=-.2, orientation='h'),
    xaxis_showgrid=True,
    yaxis_showgrid=True
)
fig.update_layout(
    shapes=[
        dict(
            type="rect",
            x0='2016-06-01',
            y0=0,
            x1='2016-7-01',
            y1=london_mean_price.values.max()*3,
            fillcolor="yellow",
            opacity=0.5,
            layer="below",
            line_width=0,
        ),
        dict(
            type="rect",
            x0="2007-12-01",
            y0=0,
            x1="2009-06-01",
            y1=london_mean_price.values.max()*3,
            fillcolor="yellow",
            opacity=0.5,
            layer="below",
            line_width=0,
        ),
        dict(
            type="rect",
            x0="2001-03-01",
            y0=0,
            x1="2001-11-01",
            y1=london_mean_price.values.max()*3,
            fillcolor="yellow",
            opacity=0.5,
            layer="below",
            line_width=0,
        ),
        dict(
            type="rect",
            x0="2020-03-01",
            y0=0,
            x1="2021-01-01",
            y1=london_mean_price.values.max()*3,
            fillcolor="yellow",
            opacity=0.5,
            layer="below",
            line_width=0,
        )
    ],
    annotations=[
            dict(text="The 2008 Recession", x='2007-12-01', y=london_mean_price.values.max()*3),
            dict(text="Brexit Vote", x='2016-06-23', y=london_mean_price.values.max()*3),
            dict(text="Dot-Com Bubble Recession", x='2001-03-01', y=london_mean_price.values.max()*3),
            dict(text="Pandemic", x='2020-03-01', y=london_mean_price.values.max()*3)
    ]
)

fig.show()
```

London has 33 boroughs (including City of London). Visually, this makes the exploration of the graph difficult. Luckily, Plotly allows to dynamically explore the data. You can click on area to hide it or double click to hide all other areas. Some key observations:

Kensington & Chelsea historically has been and remains the most expensive borough to buy a house in The more expensive boroughs have greater volatility in average price Brent had a significant decline in house prices since 2019

In [30]:
```
options = ['newham','bexley', 'barking and dagenham','london', 'westminster', 'kensington and chelsea',
'camden']


df_options = london_boroughs.loc[london_boroughs['area'].isin(options)]


#df = data.loc[(data['houses_sold'] <= 272) &          (data['no_of_crimes'] >= 1640) &
(data['no_of_crimes'] <= 2356) &
   #           data['area'].isin(options) ]

#df.loc[df['area'].isin(options)]

fig0 = px.line(df_options, x=df_options.index, y="average_price", color='area',color_discrete_map={
                "kensington and chelsea": "#FECC55",
                "westminster": "#00CC96",
                "camden": "#FF6692",
                "newham": "#FFA15A",
                "bexley": "#AB63FA",
                "barking and dagenham": "#F1664F"},)

fig0.update_xaxes(
        tickangle = 90,
        title_font = {"size": 20},
        title_standoff = 25,
        nticks=30
)

fig0.update_yaxes(nticks=20)

fig0.update_layout(
    template='gridon',
    height=500,
    width =1000,
    title='Average Monthly London House Price by Borough',
    xaxis_title='Year',
    yaxis_title='Price (£)',
    legend=dict(y=-.2, orientation='h'),
    xaxis_showgrid=True,
    yaxis_showgrid=True
)

fig0.show()
```

In [31]:
```
# Calculate the mean yearly price per borough
yearly_prices_london = london_boroughs.groupby('area').resample('y')['average_price'].mean()

# Calculate the yealy average price percentage change
yearly_prices_london_pct_ch = yearly_prices_london.groupby(level='area').apply(lambda x: x.pct_change())

yearly_prices_london_pct_ch = yearly_prices_london_pct_ch.unstack()
yearly_prices_london_pct_ch = yearly_prices_london_pct_ch.iloc[::-1]

del yearly_prices_london_pct_ch['1995-12-31']
```

In [32]:
```
fig = go.Figure(data=go.Heatmap(
        z=yearly_prices_london_pct_ch.values,
        x=yearly_prices_london_pct_ch.columns,
        y=yearly_prices_london_pct_ch.index,
        colorscale='Cividis'))

fig.update_layout(
    title='YoY Average London House Price Percentage Change',
    title_x=0.5,
    yaxis_nticks=33,
    xaxis_title='Year',
    yaxis_title='Borough'
)

fig.show()
```

In [34]:
```
monthly_data['no_of_crimes'] = monthly_data['no_of_crimes'].fillna(0)
monthly_data['houses_sold'] = monthly_data['houses_sold'].fillna(0)
```

In [35]:
```
md_london = monthly_data[(monthly_data['area'] == 'camden')]
md_london.head(5)
```

Out[35]:

In [36]:

```
# Visualize the trends in data
# sns.set_style('darkgrid')
# df.plot(kind = 'line', legend = 'reverse', title = 'Visualizing Multivariate Time-Series')
# plt.legend(loc = 'upper right', shadow = True, bbox_to_anchor = (1.35, 0.8))
# plt.show()

# Dropping area, code, & borough flag as they do not change with Time
# df_time_series = df.drop(['area', 'code', 'borough_flag'], axis = 1)  # inplace = True

# Again Visualizing the time-series data
sns.set_style('darkgrid')
md_london.plot(kind = 'line', legend = 'reverse', title = 'Visualizing Multivariate Time-Series')
# df_time_series.plot(kind = 'line', legend = 'reverse', title = 'Visualizing Multivariate Time-Series')
plt.legend(loc = 'lower right', bbox_to_anchor = (1.35, 0.8))
plt.show()
```

In [118]:

```
area = ['kensington and chelsea']
df_kc = monthly_data.loc[monthly_data['area'].isin(area)]
```

In [119]:

```
df_kc.tail()
```

Out[119]:

In [120]:

```
df_kc.shape
```

Out[120]:

```
(315, 6)
```

In [121]:

```
display(df_kc.describe());
```

```
msno.matrix(df_kc);
```

```
df_kc.dropna(axis=0)
df_kc[(df_kc != 0).all(1)]
```

```
df_kc.drop(['area', 'code', 'borough_flag'], axis = 1, inplace= True)
```

```
df_kc.head()
```

In [125]:

```
df_kc['houses_sold'] = pd.to_numeric(df_kc['houses_sold'], downcast="float")
```

```
# Plot
fig, axes = plt.subplots(nrows=3, ncols=1, dpi=220, figsize=(10,6))
for i, ax in enumerate(axes.flatten()):
    data = df_kc[df_kc.columns[i]]
    ax.plot(data, color='red', linewidth=1)
    # Decorations
    ax.set_title(df_kc.columns[i])
    ax.xaxis.set_ticks_position('none')
    ax.yaxis.set_ticks_position('none')
    ax.spines["top"].set_alpha(0)
    ax.tick_params(labelsize=6)

plt.tight_layout();
```

Granger Causality

In [128]:

```
from statsmodels.tsa.stattools import grangercausalitytests
maxlag=12
test = 'ssr_chi2test'
def grangers_causation_matrix(data, variables, test='ssr_chi2test', verbose=False):

    df_kc = pd.DataFrame(np.zeros((len(variables), len(variables))), columns=variables, index=variables)
    for c in df_kc.columns:
        for r in df_kc.index:
            test_result = grangercausalitytests(data[[r, c]], maxlag=maxlag, verbose=False)
            p_values = [round(test_result[i+1][0][test][1],4) for i in range(maxlag)]
            if verbose: print(f'Y = {r}, X = {c}, P Values = {p_values}')
            min_p_value = np.min(p_values)
            df_kc.loc[r, c] = min_p_value
    df_kc.columns = [var + '_x' for var in variables]
    df_kc.index = [var + '_y' for var in variables]
    return df_kc

grangers_causation_matrix(df_kc, variables = df_kc.columns)
```

Out[128]:

Cointegration Test

In [148]:

```
from statsmodels.tsa.vector_ar.vecm import coint_johansen

def cointegration_test(df_kc, alpha=0.05):
    """Perform Johanson's Cointegration Test and Report Summary"""
    out = coint_johansen(df_kc,-1,5)
    d = {'0.90':0, '0.95':1, '0.99':2}
    traces = out.lr1
    cvts = out.cvt[:, d[str(1-alpha)]]
    def adjust(val, length= 6): return str(val).ljust(length)

    # Summary
    print('Name   ::  Test Stat > C(95%)    =>   Signif  \n', '--'*20)
    for col, trace, cvt in zip(df_kc.columns, traces, cvts):
        print(adjust(col), ':: ', adjust(round(trace,2), 9), ">", adjust(cvt, 8), ' =>  ' , trace > cvt)

cointegration_test(df_kc)
```

```
Name   ::  Test Stat > C(95%)    =>   Signif
 ----------------------------------------
average_price ::  18.3      > 24.2761   =>   False
houses_sold ::  5.27      > 12.3212   =>   False
no_of_crimes ::  0.01      > 4.1296    =>   False
```

Test harness

In [149]:

```
nobs0 = 32 #10percentofdata
dataset, validation = df_kc[:-nobs0], df_kc[-nobs0:]
print('Dataset %d, Validation %d' % (len(dataset), len(validation)))
```

```
Dataset 283, Validation 32
```

In [150]:

```
#Splitting the dataset into train & test subsets
nobs = 84
df_train, df_test = dataset[:-nobs], dataset[-nobs:]

# Check size
print(df_train.shape)  # (199, 3)
print(df_test.shape)  # (84, 3)
```

```
(199, 3)
(84, 3)
```

In [171]:

```
def adfuller_test(series, signif=0.05, name='', verbose=False):
    """Perform ADFuller to test for Stationarity of given series and print report"""
    r = adfuller(series, autolag='AIC')
    output = {'test_statistic':round(r[0], 4), 'pvalue':round(r[1], 4), 'n_lags':round(r[2], 4),
'n_obs':r[3]}
    p_value = output['pvalue']
    def adjust(val, length= 6): return str(val).ljust(length)

    # Print Summary
    print(f'    Augmented Dickey-Fuller Test on "{name}"', "\n   ", '-'*47)
    print(f' Null Hypothesis: Data has unit root. Non-Stationary.')
    print(f' Significance Level    {signif}')
    print(f' Test Statistic        {output["test_statistic"]}')
    print(f' No. Lags Chosen       {output["n_lags"]}')

    for key,val in r[4].items():
        print(f' Critical value {adjust(key)}  {round(val, 3)}')

    if p_value <= signif:
        print(f" P-Value  {p_value}. Rejecting Null Hypothesis.")
        print(f" Series is Stationary.")
    else:
        print(f" P-Value  {p_value}. Weak evidence to reject the Null Hypothesis.")
        print(f" Series is Non-Stationary.")
```

In [172]:

```
# ADF Test on each column
for name, column in df_train.iteritems():
    adfuller_test(column, name=column.name)
    print('\n')
```

```
    Augmented Dickey-Fuller Test on "average_price"
    -----------------------------------------------
 Null Hypothesis: Data has unit root. Non-Stationary.
 Significance Level    0.05
 Test Statistic        0.2124
 No. Lags Chosen       12
 Critical value 1%      -3.466
 Critical value 5%      -2.877
 Critical value 10%     -2.575
 P-Value  0.973. Weak evidence to reject the Null Hypothesis.
 Series is Non-Stationary.


    Augmented Dickey-Fuller Test on "houses_sold"
    -----------------------------------------------
 Null Hypothesis: Data has unit root. Non-Stationary.
 Significance Level    0.05
 Test Statistic        -2.0587
 No. Lags Chosen       12
 Critical value 1%      -3.466
 Critical value 5%      -2.877
 Critical value 10%     -2.575
 P-Value  0.2615. Weak evidence to reject the Null Hypothesis.
 Series is Non-Stationary.


    Augmented Dickey-Fuller Test on "no_of_crimes"
    -----------------------------------------------
 Null Hypothesis: Data has unit root. Non-Stationary.
 Significance Level    0.05
 Test Statistic        -1.5775
 No. Lags Chosen       12
 Critical value 1%      -3.466
 Critical value 5%      -2.877
 Critical value 10%     -2.575
 P-Value  0.4949. Weak evidence to reject the Null Hypothesis.
 Series is Non-Stationary.
```

In [173]:

```
# 1st difference
df_differenced = df_train.diff().dropna()

# ADF Test on each column of 1st Differences Dataframe
for name, column in df_differenced.iteritems():
    adfuller_test(column, name=column.name)
    print('\n')
```

```
    Augmented Dickey-Fuller Test on "average_price"
    -----------------------------------------------
 Null Hypothesis: Data has unit root. Non-Stationary.
 Significance Level    0.05
 Test Statistic        -4.1939
 No. Lags Chosen       11
 Critical value 1%      -3.466
 Critical value 5%      -2.877
 Critical value 10%     -2.575
 P-Value  0.0007. Rejecting Null Hypothesis.
 Series is Stationary.


    Augmented Dickey-Fuller Test on "houses_sold"
    -----------------------------------------------
 Null Hypothesis: Data has unit root. Non-Stationary.
 Significance Level    0.05
 Test Statistic        -4.2828
 No. Lags Chosen       11
 Critical value 1%      -3.466
 Critical value 5%      -2.877
 Critical value 10%     -2.575
 P-Value  0.0005. Rejecting Null Hypothesis.
 Series is Stationary.


    Augmented Dickey-Fuller Test on "no_of_crimes"
    -----------------------------------------------
 Null Hypothesis: Data has unit root. Non-Stationary.
 Significance Level    0.05
 Test Statistic        -3.8108
 No. Lags Chosen       11
 Critical value 1%      -3.466
 Critical value 5%      -2.877
 Critical value 10%     -2.575
 P-Value  0.0028. Rejecting Null Hypothesis.
 Series is Stationary.
```

VAR (p) model

In [154]:

```
model = VAR(df_differenced)
for i in [1,2,3,4,5,6,7,8,9]:
    result = model.fit(i)
    print('Lag Order =', i)
    print('AIC : ', result.aic)
    print('BIC : ', result.bic)
    print('FPE : ', result.fpe)
    print('HQIC: ', result.hqic, '\n')
```

```
Lag Order = 1
AIC :  38.11347956501855
BIC :  38.313471670322386
FPE :  3.56846247377955e+16
HQIC:  38.194437861122026

Lag Order = 2
AIC :  38.115835929765424
BIC :  38.46706250039726
FPE :  3.5771472296476024e+16
HQIC:  38.25802927873056
```

```
Lag Order = 3
AIC :  37.939078294858376
BIC :  38.44261668848357
FPE :  2.998124604749405e+16
HQIC:  38.142955026191295

Lag Order = 4
AIC :  37.936762191007666
BIC :  38.59370274875751
FPE :  2.992185729135487e+16
HQIC:  38.20277590731645
```

In [155]:

```
x = model.select_order(maxlags=12)
x.summary()
```

Out[155]:

In [156]:

```
model_fitted = model.fit(4)
model_fitted.summary()
```

Out[156]:

Check for Serial Correlation of Residuals (Errors) using Durbin Watson Statistic

In [157]:

```
from statsmodels.stats.stattools import durbin_watson
out = durbin_watson(model_fitted.resid)

for col, val in zip(df_kc.columns, out):
    print(col, ':', round(val, 2))
```

```
average_price : 1.99
houses_sold : 1.99
no_of_crimes : 1.98
```

Forecast VAR model using statsmodels

In [158]:

```
# Get the lag order
lag_order = model_fitted.k_ar
print(lag_order)  #> 4

# Input data for forecasting
forecast_input = df_differenced.values[-lag_order:]
forecast_input
```

```
4
```

Out[158]:

```
array([[ 7.9963e+04,  3.1000e+01, -1.4000e+02],
       [-2.0318e+04, -1.0800e+02,  3.0500e+02],
       [-2.7462e+04,  2.9000e+01, -2.7900e+02],
       [-3.4382e+04,  5.8000e+01,  1.0000e+00]])
```

In [159]:

```
# Forecast
fc = model_fitted.forecast(y=forecast_input, steps=nobs)
df_forecast = pd.DataFrame(fc, index=df_kc.index[-nobs:], columns=df_kc.columns + '_2d')
df_forecast
```

Out[159]:

Invert the transformation to get the real forecast

In [160]:

```
def invert_transformation(df_train, df_forecast, second_diff=False):
    """Revert back the differencing to get the forecast to original scale."""
    df_fc = df_forecast.copy()
    columns = df_train.columns
    for col in columns:
        # Roll back 2nd Diff
        if second_diff:
            df_fc[str(col)+'_1d'] = (df_train[col].iloc[-1]-df_train[col].iloc[-2]) +
df_fc[str(col)+'_2d'].cumsum()
        # Roll back 1st Diff
        df_fc[str(col)+'_forecast'] = df_train[col].iloc[-1] + df_fc[str(col)+'_1d'].cumsum()
    return df_fc
```

In [161]:

```
df_results = invert_transformation(df_train, df_forecast, second_diff=True)
df_results.loc[:, ['average_price_forecast', 'houses_sold_forecast', 'no_of_crimes_forecast']]
```

Out[161]:

In [175]:

```
fig, axes = plt.subplots(nrows=int(len(validation.columns)/1), ncols=1, dpi=300, figsize=(10,10))
for i, (col,ax) in enumerate(zip(validation.columns, axes.flatten())):
    df_results[col+'_forecast'].plot(legend=True, ax=ax).autoscale(axis='x',tight=True)
    df_test[col][-nobs:].plot(legend=True, ax=ax);
    ax.set_title(col + ": Forecast vs Actuals")
    ax.xaxis.set_ticks_position('none')
    ax.yaxis.set_ticks_position('none')
    ax.spines["top"].set_alpha(0)
    ax.tick_params(labelsize=6)

plt.tight_layout();
```

In [163]:

```
from statsmodels.tsa.stattools import acf
def forecast_accuracy(forecast, actual):
    mape = np.mean(np.abs(forecast - actual)/np.abs(actual))  # MAPE
    me = np.mean(forecast - actual)             # ME
    mae = np.mean(np.abs(forecast - actual))    # MAE
    mpe = np.mean((forecast - actual)/actual)   # MPE
    rmse = np.mean((forecast - actual)**2)**.5  # RMSE
    corr = np.corrcoef(forecast, actual)[0,1]   # corr
    mins = np.amin(np.hstack([forecast[:,None],
                              actual[:,None]]), axis=1)
    maxs = np.amax(np.hstack([forecast[:,None],
                              actual[:,None]]), axis=1)
    minmax = 1 - np.mean(mins/maxs)             # minmax
    return({'mape':mape, 'me':me, 'mae': mae,
            'mpe': mpe, 'rmse':rmse, 'corr':corr, 'minmax':minmax})

print('Forecast Accuracy of: average_price')
accuracy_prod = forecast_accuracy(df_results['average_price_forecast'].values, df_test['average_price'])
for k, v in accuracy_prod.items():
    print((k), ': ', round(v,4))

print('\nForecast Accuracy of: houses_sold')
accuracy_prod = forecast_accuracy(df_results['houses_sold_forecast'].values, df_test['houses_sold'])
for k, v in accuracy_prod.items():
    print((k), ': ', round(v,4))

print('\nForecast Accuracy of: no_of_crimes')
accuracy_prod = forecast_accuracy(df_results['no_of_crimes_forecast'].values, df_test['no_of_crimes'])
for k, v in accuracy_prod.items():
    print((k), ': ', round(v,4))
```

```
Forecast Accuracy of: average_price
mape :  3.0641
me :  4013759.9777
mae :  4018368.9863
mpe :  3.0591
rmse :  5518427.4863
corr :  0.7323
minmax :  0.5766

Forecast Accuracy of: houses_sold
mape :  3.6221
me :  543.8342
mae :  543.8342
mpe :  3.6221
rmse :  581.8347
corr :  -0.2576
minmax :  0.7181

Forecast Accuracy of: no_of_crimes
mape :  7.7002
me :  13011.9737
mae :  13019.1368
mpe :  7.6966
rmse :  16869.4743
corr :  0.2316
minmax :  0.7516
```

In [40]:

```
## Evaluation
fevd = results.fevd(10)
fevd.summary()
```

ARIMA (p, d, q) Model

```
from statsmodels.graphics.tsaplots import plot_acf
from statsmodels.graphics.tsaplots import plot_pacf
from matplotlib import pyplot

pyplot.figure()
pyplot.subplot(221)
plot_acf(df_differenced, ax=pyplot.gca())
pyplot.subplot(222)
```

```
plot_pacf(df_differenced, ax=pyplot.gca())
pyplot.show()
                                                                                                In [72]:
# evaluate an ARIMA model for a given order (p,d,q) and return RMSE
def evaluate_arima_model(X, arima_order):
    # prepare training dataset

    X = X.astype('float32')
    train_size = int(len(X) * 0.70)
    train, test = X[0:train_size], X[train_size:]
    history = [x for x in train]
    # make predictions
    predictions = list()
    for t in range(len(test)):
        model = ARIMA(history, order=arima_order)
        model_fit = model.fit()
        yhat = model_fit.forecast()[0]
        predictions.append(yhat)
        history.append(test[t])
    # calculate out of sample error
    rmse = sqrt(mean_squared_error(test, predictions))
    return rmse

# evaluate combinations of p, d and q values for an ARIMA model
def evaluate_models(dataset, p_values, d_values, q_values):
    dataset = dataset.astype('float32')
    best_score, best_cfg = float("inf"), None
    for p in p_values:
        for d in d_values:
            for q in q_values:
                order = (p,d,q)
                try:
                    rmse = evaluate_arima_model(dataset, order)
                    if rmse < best_score:
                        best_score, best_cfg = rmse, order
                    print('ARIMA%s RMSE=%.3f' % (order,rmse))
                except:
                    continue
    print('Best ARIMA%s RMSE=%.3f' % (best_cfg, best_score))
 #evaluate parameters
p_values = range(0, 5)
d_values = range(0, 4)
q_values = range(0, 7)
warnings.filterwarnings("ignore")
evaluate_models(dataset.values, p_values, d_values, q_values)
ARIMA(4, 0, 0) RMSE=79905987.563
ARIMA(4, 0, 1) RMSE=31872.903
ARIMA(4, 0, 2) RMSE=30183.078
ARIMA(4, 0, 4) RMSE=29332.731
ARIMA(4, 0, 5) RMSE=29545.545
ARIMA(4, 2, 4) RMSE=30325.652
ARIMA(4, 2, 6) RMSE=29615.519
ARIMA(4, 3, 0) RMSE=42416.946
ARIMA(4, 3, 1) RMSE=35477.922
ARIMA(4, 3, 2) RMSE=33763.369
ARIMA(4, 3, 3) RMSE=34509.511
ARIMA(4, 3, 4) RMSE=33290.296
ARIMA(4, 3, 5) RMSE=29493.111
ARIMA(4, 3, 6) RMSE=31003.782
Best ARIMA(4, 0, 6) RMSE=28973.799

# predict
model = ARIMA(dataset, order=(4,0,6))
model_fit = model.fit()
yhat = model_fit.forecast()[0]
predictions.append(yhat)
print('Predicted: %.3f' % yhat)
Predicted: 1402101.223

# evaluate the finalized model on the validation dataset
from pandas import read_csv
from matplotlib import pyplot
from statsmodels.tsa.arima.model import ARIMA
from statsmodels.tsa.arima.model import ARIMAResults
from scipy.stats import boxcox
from sklearn.metrics import mean_squared_error
from math import sqrt
from math import exp
from math import log
import numpy

# load and prepare datasets
X = dataset.values.astype('float32')
history = [x for x in X]
y = validation.values.astype('float32')
predictions = list()
```

```
# predict
model = ARIMA(X, order=(4,0,6))
model_fit = model.fit()
yhat = model_fit.forecast()[0]
predictions.append(yhat)


# make first prediction
yhat = model_fit.forecast()[0]
predictions.append(yhat)
history.append(y[0])
print('>Predicted=%.3f, Expected=%.3f' % (yhat, y[0]))

# rolling forecasts
for i in range(1, len(y)):
    model = ARIMA(history, order=(4,0,6))
    model_fit = model.fit()
    yhat = model_fit.forecast()[0]
    predictions.append(yhat)
    # observation
    obs = y[i]
    history.append(obs)
    print('>Predicted=%.3f, Expected=%.3f' % (yhat, obs))
```

```
>Predicted=1402101.223, Expected=1418032.000
>Predicted=1446337.976, Expected=1388037.000
>Predicted=1339473.863, Expected=1348121.000
>Predicted=1329506.653, Expected=1363458.000
>Predicted=1387797.077, Expected=1345805.000
>Predicted=1285932.070, Expected=1322480.000
>Predicted=1332498.740, Expected=1341825.000
>Predicted=1389842.408, Expected=1335987.000
>Predicted=1331143.849, Expected=1265639.000
>Predicted=1238126.922, Expected=1332134.000
```

```
# report performance
pyplot.plot(y)
pyplot.plot(predictions, color='red')
pyplot.show()
```

RMSE =57758.47155

In [123]:

```
# plot forecasts against actual outcomes
pyplot.plot(test)
pyplot.plot(predictions, color='red')
pyplot.show()
```

Review Residual Errors

```
# errors
residuals = [test[i]-predictions[i] for i in range(len(test))]
residuals = DataFrame(residuals)
pyplot.figure()
pyplot.subplot(211)
residuals.hist(ax=pyplot.gca())
pyplot.subplot(212)
residuals.plot(kind='kde', ax=pyplot.gca())
pyplot.show()
```

# errors

```
pyplot.subplot(221)
plot_acf(residuals, lags=25, ax=pyplot.gca())
pyplot.subplot(222)
plot_pacf(residuals, lags=25, ax=pyplot.gca())
pyplot.show()
```

Finalize Model

```
# Plot residual errors
residuals = pd.DataFrame(model_fit.resid)
fig, ax = plt.subplots(1,2)
residuals.plot(title="Residuals", ax=ax[0])
residuals.plot(kind='kde', title='Density', ax=ax[1])
plt.show()

# Actual vs Fitted
model_fit.plot_predict(dynamic=False)
plt.show()
```