COUB01
F014621

Predicting Inflation With Machine Learning

by

Ben M. Taylor

A Project Report

Submitted in partial fulfilment
of the requirements for the award of

B.Sc.
in
Computer Science
of
Loughborough University

1st May 2024

Acknowledgements

I would like to thank my supervisor Mohamad Saada for his excellent support and guidance throughout this project.

Abstract

In October 2022, the UK hit an inflation rate of 11.1%, the country's highest in over 40 years. Now more than ever, the ability to accurately predict inflation and other financial indicators is a crucial skill required by the government and the individual to prepare themselves for the future financially. In a time where Artificial Intelligence and Machine Learning are ever flourishing, it is only natural to attempt to use these tools at our disposal to predict and combat the issues we face.

In this paper I will attempt to predict inflation through the use of machine learning eventually presenting my findings and evaluations.

Keywords: Inflation, Artificial Intelligence, Machine Learning

Contents

List of Figures

Chapter 1

Introduction

1.1 Motivation

Prices for goods and services are ever-changing, constantly affecting individuals, organizations, and governments. The clients or individuals interacting with these services would all benefit from the ability to predict the rise and fall of prices as it can impact their choices to spend, invest, or save. Ultimately allowing them to make the most out of the resources they have.

However, the value of inflation is an incredibly difficult indicator to predict to the point that even establishments like the Bank of England, who have direct control over interest rates, fail to correctly forecast it. With the ability to accurately predict inflation having so many stakeholders and huge organizations working on the problem, it is extremely unlikely that this project will be able to find a solution that is far better than all those before it. Instead, this project aims to understand which machine learning (ML) methods produce the best results when faced with this task and potentially find out why this is the case. The main goal is for this project's results to ultimately contribute to the ongoing research into inflation forecasting and help clarify which ML methods may be most suitable.

1.2 Aims and Objectives

1.2.1 Aims

This project aims to create multiple machine learning models and compare their predictive abilities when it comes to UK inflation. The findings will then be presented in this report, outlining the advantages and disadvantages of each tested model. A conclusion will then be made about which models performed best and in which scenarios certain models should be used over others.

1.2.2 Objectives

This project can be broken down into a list of objectives. Objectives not only provide a path to follow to implement the project but also a way to evaluate the project's success post-completion. The objectives are placed into phases corresponding to the project's work plan.

1. Phase 1: Research
    (a) Conduct a literature review to understand the current landscape of inflation and other economic indicator forecasting.
    (b) Research prominent models in the literature.
2. Phase 2: Source Data
    (a) Source the data to be used in the models.
    (b) Evaluate the usefulness and appropriateness of the data.
    (c) Clean the data: this includes dropping and retaining specific variables based on how appropriate they are for predicting inflation.
    (d) Preprocess the data further until it is appropriate for use in various machine learning models.
3. Phase 3: Creating Models
    (a) Choose at least three appropriate models to use.
    (b) Develop and tune the models.
    (c) Train the models on the dataset.
4. Phase 4: Evaluation and Report
    (a) Evaluate the models using statistical tests.
    (b) Conclude the findings.
    (c) Present the findings in this final report.

Chapter 2

Literature Review

2.1 Motivation

Embarking on a literature review before developing a project offers numerous benefits. Understanding existing knowledge in Machine Learning, specifically when used to predict financial indicators, helps to contextualise the research and position it within the existing field. Reviewing previous literature also provides the benefits of identifying gaps in current research and finding supporting arguments that can provide a guide for the work that needs to be done during the project. Additionally, the literature review helps the project avoid redundancy between it and other similar works. Completing the literature review will provide a strong foundation to start and guide the project.

2.2 Available Literature and Context

There is a strong monetary incentive to produce research that outlines how best to predict financial indicators. The correct predictions can not only allow organizations and individuals to profit greatly but also to avoid loss. This has resulted in a myriad of papers being written that experiment with a variety of techniques to predict future values. Many such papers will provide helpful insight that will support the development of the models in this project.

This report's topic focuses on the prediction of inflation through the use of machine learning. To accomplish this it is important to view papers predominantly addressing two types of topics. The first type is papers that focus on the topic of predicting inflation or other economic indicators and time series. The second type is papers dealing with different machine-learning techniques. It will also be pertinent to survey the current literature on inflation: its causing factors, effects, and significance.

2.2.1 Financial Indicator Prediction Papers

According to "Analysis of Financial Time Series" by Ruey S. Tsay: "Financial time series analysis is concerned with the theory and practice of asset valuation over time."[1] There are many financial time series (FTS for short) prediction methods both theoretical and practical that have attracted attention. The taxonomy of these FTS analysis methods is shown in figure 2.1. The predominant analysis strategies for predicting financial market behaviour are fundamental analysis and technical analysis[2].

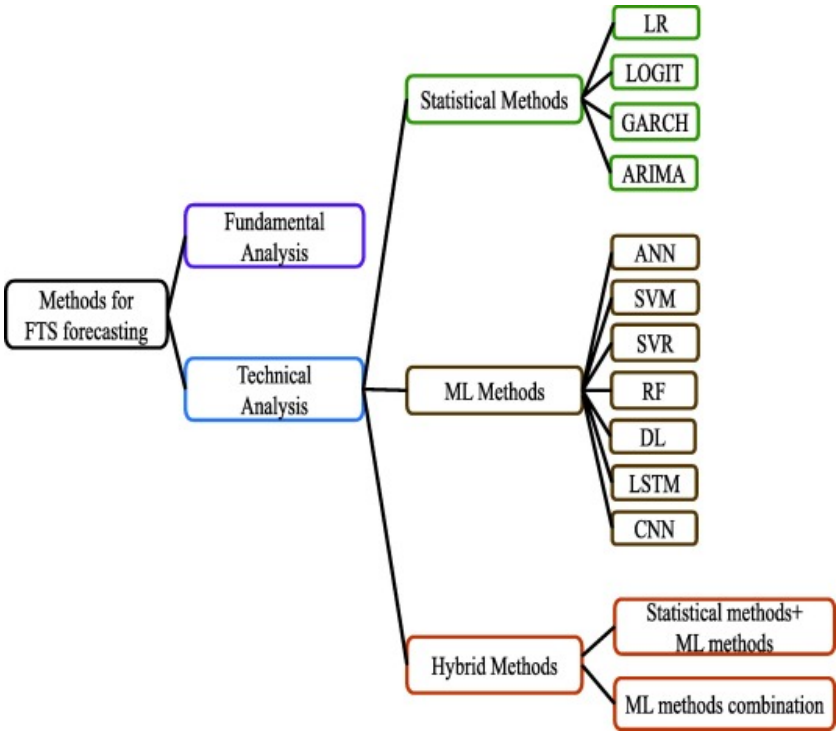


Figure 2.1: FTS Forecasting Methods.
Figure from page 3 of 'A survey on machine learning models for financial time series forecasting by Yajiao Tang et al.'[3]

Fundamental Analysis

Fundamental analysis[4] attempts to measure the intrinsic value of an asset by looking at current market and economic conditions. Additionally, fundamental analysis frequently makes use of techniques - such as sentiment analysis - that often deal with unstructured data. The success of fundamental analysis often relies on the financial efficiency of the target[5], which according to Tom Seegmiller is defined as "how successful your organization is at turning expenses into revenue"[6].

Technical Analysis

Technical analysis[7] attempts to identify opportunities and predict investments by viewing movements and trends in market data alongside using a variety of technical indicators. Unlike fundamental analysis, technical analysis does not take into account many of the same fundamentals that can help indicate an asset's current value such as quarterly revenue. This is partially important because it is often argued that technical indicators such as inflation or a stock's value are already priced according to the fundamentals that cause or contribute to them[8]. From this, we can come to the understanding that while fundamental analysis is the idea of looking at the current factors affecting an asset and using them to evaluate the asset's true value; Technical analysis is built upon the idea that past performance can predict future performance. Traditionally, technical analysis has relied heavily on statistical models to forecast the future performance of assets[9]. Furthermore, the act of utilising past values to predict future values has been widely implemented for years. One of the earliest uses of autoregressive models being used to predict time series was created by U.G.Yule in the 1920s[10]. However, with the increase of big data and the internet, ever-larger amounts of financial predictive data are continually being produced. Nowadays, simple statistical models may struggle to produce accurate future predictions when faced with big data sets containing complex characteristics[11].

2.2.2 Machine Learning Papers

This subsection will explore the literature on machine learning algorithms[12]. According to Mariette Awad et al. machine learning "is a branch of artificial intelligence that systematically applies algorithms to synthesize the underlying relationships among data and information"[13]. Currently, there is an abundance of fresh machine learning papers constantly being produced[14].

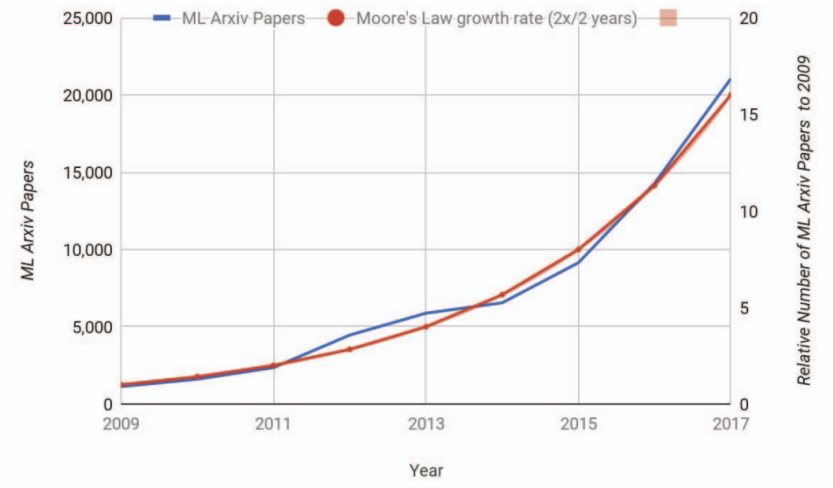


Figure 2.2: ML arXiv articles per year.
Figure from page 4 of 'A New Golden Age in Computer Architecture: Empowering the MachineLearning Revolution' by Jeff Dean, David Patterson, and Cliff Young[14]

As figure 2.2 indicates, articles on machine learning posted to arXiV (an archive for scholarly articles) have more than doubled every two years. Additionally, in 2018 the number of articles released peaked at over 100 per day, summing to more than 33,000 by the end of the year. The number of articles released has steadily continued to increase in the years since[15]. Naturally, to read this many articles is impossible, however, the sheer quantity bodes well for this project as it means there will be plenty of guidance on how best to select and develop predictive models.

2.2.3 Inflation Papers

According to Ceyda Oner at the International Monetary Fund "Inflation is the rate of increase in prices over a given period of time"[16]. Often inflation is used to broadly indicate a country's currency's global state of price fluctuation, however, inflation can also be used for certain services, goods, or food. Inflation affects everyone which leads to a vast amount of different papers linking to the topic. The papers focusing on inflation cover a variety of topics such as inflation's various effects, forecasting inflation, the causes and trends of inflation, and more. [17]

2.3 Problem Domain

2.3.1 UK Inflation

Inflation has been around since money has been used with one of the earliest recordings of inflation being caused by the death of Alexander the Great in around 300 BC[18]. Because of inflation's pervasiveness and its intrinsic link to both macro and microeconomics, there is a surplus of literature surrounding the subject. This report will focus on inflation in the UK as opposed to global inflation. Several indicators can be used to measure inflation, the most common of which are Consumer Price Index (CPI), Consumer Price Index with Housing (CPIH), and Retail Price Index (RPI). There are also more novel measurements of inflation as well such as viewing the Big Mac index[19] and recording its change over time.

In the UK, inflation is the responsibility of the Bank of England (BoE) who set monetary policy eight times a year with the goal of controlling and stablising inflation. The BoE calculates inflation using their own in-house models[20]. The BoE is not transparent with the specifics of the models or exactly how they calculate inflation, however, they state their models do factor in market expectations. In October 2022 inflation rates reached a peak of 11.1% the highest in over 40 years. Yet by October 2023, the annual rate was the lowest since October 2021 at 4.7%[21]. Granted this is still higher than the targeted 2% imposed on the BoE by the government. According to a research briefing published by the House of Commons Library, the main causes for the extremely high current inflation were: Covid-19 lockdowns causing supply chain disruptions and Russia's invasion of Ukraine causing increased energy prices due to the UK's prior reliance on Russian fuel[22]. Additionally, several pundits have suggested the BoE's slow reaction to combating high inflation rates and their increased money-printing during the pandemic added to the issue[23]. The main way that the BoE combats inflation is through the manipulation of interest rates. The general premise is that as inflation rates grow, the growth of inflation should slow and inflation should eventually decrease. This is because increased interest rates mean the overall spending in the economy lessens as money is instead being spent on trying to mitigate the higher interest.

2.3.2 Machine Learning Models

Machine learning techniques have been used in financial forecasting endeavors to improve upon the performance of traditional statistical models. Generally, the goal of FTS forecasting can be placed into two main categories:

1. Price prediction.

2. Price movement prediction (this includes volatility predictions).
These two goals also reflect two types of machine-learning problems: regression and classification. This project will focus mainly on the price prediction/regression categories regarding inflation. This means that the aim of this project will be to predict future values of inflation as opposed to whether or not inflation will increase or decrease. Naturally, the areas of literature that will be studied will be with this regression problem in mind.

Artificial Neural Networks

An artificial neural network is a machine learning model that is made up of an interconnected group of nodes (also known as neurons) organised into layers. The model's inspiration stems from how neurons in the human brain interact with one another. In 1957 Frank Rosenblatt invented the perceptron, one of if not the first implementations of an artificial neural network [24].

There are many different types of ANNs. Two of the ANNs that commonly occur in the literature for predicting financial time series are feedforward neural networks (FNNS) and recurrent neural networks (RNNs). Feedforward neural networks and recurrent neural networks are differentiated by the direction through the network in which the data flows.

In FNNs the data flow is unidirectional, this means that it flows forward through the network from one layer to the next. This was the first type of neural network introduced in 1958 by Frank Rosenblatt[24]. In more recent times, a method called backpropagation[25] has been introduced to help train feedforward networks. Backpropagation essentially works by sending the error from an output back through the network and optimising the weights and biases along the way.

The data in RNNs can flow in a loop by using recurrent connections. This allows RNNs to keep an internal memory of past data that can affect future data and calculations. The ability to store past data makes RNNs useful for tasks involving sequential data such as speech recognition[26] or time series analysis[27].

ANNs are composed of three types of layers: the input, hidden, and output layers. The input layer receives input data and passes it through to the first hidden layer. Hidden layers receive weighted inputs, perform an activation function on said input, and then pass the new data to the next layer. The output layer receives data from the final hidden layer and then produces the resulting prediction. The neurons within an ANN can either be excited or inhibited. Neurons are connected between layers and the strength of these connections (the weight) is decided by how excited or inhibited a neuron is. Each neuron in the hidden and output layers contains biases and activation functions (with the activation function in

the output layer typically being different from the ones used in the hidden layer). Activation functions are used to introduce non-linearity into a neural network. The purpose of this is to enable a network to model more complex patterns in the data. Activation functions can also be used to keep inputs within a specified range to decrease the processing time required for a network to run. Activation values are passed from node to node through the connections in the network. When a neuron receives the activation value, it sums and modifies it based on the neuron's activation function and bias. The predicted results can then be compared to the true values and the weights and biases of the network are updated accordingly. Artificial neural networks can be modified with a variety of techniques to alter their accuracy. One of these alterations is changing the activation function of the neurons. Figure 2.3 shows some common activation functions.

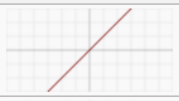
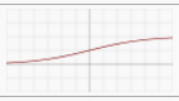
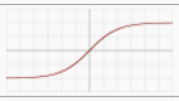
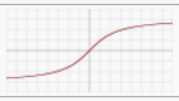
Name	Plot	Equation
Identity		$f(x) = x$
Binary step		$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$
Logistic (a.k.a Soft step)		$f(x) = \frac{1}{1+e^{-x}}$
TanH		$f(x) = \tanh(x) = \frac{2}{1+e^{-2x}} - 1$
ArcTan		$f(x) = \tan^{-1}(x)$
Rectified Linear Unit (ReLU)		$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$
Parameteric Rectified Linear Unit (PReLU) [2]		$f(x) = \begin{cases} \alpha x & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$
Exponential Linear Unit (ELU) [3]		$f(x) = \begin{cases} \alpha(e^x - 1) & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$
SoftPlus		$f(x) = \log_e(1 + e^x)$

Figure 2.3: Common Activation Functions [28]

Other commonly used activation functions include Sigmoid, Guassian, and leaky ReLU.

ANNs have many advantages:

- Thanks to the many interconnected neurons, ANNs have strong learning capabilities[29].
- ANNs do not have a fixed structural equation making them very adaptable.
- ANNs can be finely tuned with many changes to the network in order to find the best-fitting model. For example, changing the number of hidden layers, the number of nodes in a layer, the activation functions, and so on.

ANNs also come with some disadvantages:

- The higher complexity of ANNs means that they require more resources than traditional statistical models.
- Due to the nature of the hidden layers, ANNs can be hard to interpret.
- ANNs can be overfitted[30].

Support Vector Regression

Created by Vladimir Vapnik and Alexey Chervonenkis in the 1960s and later built upon further by Vapnik et al. with the addition of the kernel trick and soft margin [31], Support Vector Regression and Support Vector Machines are supervised learning methods used for regression and classification respectively. Both models use non-linear mapping to transform the dimension of the input data. Then utilise a hyperplane in order to either best fit or categorize the data. The hyperplane for these models is found by using an ε-insensitive tube, meaning that any errors within the range of the tube are ignored. This is unlike a standard line of best fit that takes into account the ε (distance) of all points to the line, instead, only errors outside of the tube are considered pertinent. The hyperplane is then placed in a way in which the sum of all points outside of the tube is minimised. The points outside of the tube are called support vectors hence the name support vector regressions.

Advantages of Support Vector Regression:

- SVRs are simple and easy to implement.
- SVRs can produce easily interpretable results.
- SVRs require less computational resources than other models.
- SVRs can maintain stability despite noisy input data thanks to the ε-incentive tube[32].

Disadvantages of Support Vector Regression:

- Deciding a suitable kernel function can cause difficulty [33].
- SVRs may struggle with big data[34].
- SVRs may struggle with very complex data and relationships.

Random Forest

The first random forest algorithm was created by Tin Kam Ho in 1995[35] which was later developed upon by Leo Breiman[36]. The random forest model makes predictions by consulting multiple decision trees. Each tree is trained on a random subset of data taken from the training set, this is called bagging or bootstrap aggregation. The final prediction is an average taken from all of the trees' predictions.

Advantages of random forest:

- Random forest can prevent overfitting by combining the results of several weak learners instead of using one powerful learner[37].
- Random forest models have good prediction accuracy as the result is an average, making it unlikely to be an outlier.
- Random forest models are stable as changes to the data set may affect one tree but are unlikely to affect many trees.

Disadvantages of random forest:

- Random forest models suffer from increased training time. This is due to the fact that to make a prediction you need predictions from all of the trees to get an average.
- Random forests distribute the data between trees randomly thus potentially losing some relationships or patterns in the data.

2.3.3 Regression Metrics and Model Evaluation

Evaluating a model is extremely important not only to know the quality of the model's predictions but also in order to make improvements to the model. Evaluation metrics can be used to evaluate the performance of the model. Some useful metrics that can be applied to measure a model's performance are: Mean absolute error (MAE), Mean squared error (MSE), Mean absolute percentage error (MAPE), Root mean absolute error (RMAE), Normalised mean square error (NMSE), Root mean squared error (RMSE), Relative root mean squared error (RRMSE), Correlation coefficient of prediction (R) The most commonly used metrics are MSE, MAE, MAPE, and R.

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|Y_i - X_i|$$
n is the number of data points.
Y_i is the ith true value.
X_i is the ith predicted value.

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(Y_i - X_i)^2$$
n is the number of data points.
Y_i is the ith true value.
X_i is the ith predicted value.

$$MAPE = \frac{1}{n}\sum_{i=1}^{n}\left|\frac{Y_i - X_i}{Y_i}\right|$$
n is the number of data points
Y_i is the true value.
X_i is the predicted value.

$$R = \frac{\sum_{i=1}^{n}(Y_i - \bar{Y})(X_i - \bar{X})}{\sqrt{\sum_{i=1}^{n}(X_i - \bar{X})^2 \sum_{i=1}^{n}(Y_i - \bar{Y})^2}}$$
X_i and Y_i are the data points.
$\bar{X}$ is the mean of the x-value and $\bar{Y}$ the mean of the y values.

FOR MSE, MAE, and MAPE a lower result indicates a more accurate model, and when a model has no error the value will be zero. R is always between -1 and 1, when R=0 it indicates that there is no linear relationship between the values. If R is -1 then there is a perfect negative linear relationship and if R is 1 then there is a perfect positive linear relationship. These formulas can be used to evaluate the predictive ability of a model.

2.4 Summary and Conclusion

This literature review has covered a sample of the literature on machine learning methods, financial time series forecasting, and inflation. There is a multitude of literature available in these areas due to the potential monetary gain as well as the new and emerging technologies being explored in the fields. Therefore, it would be nearly impossible to extensively cover all of the articles relevant to this project. Instead, the main focus was on understanding the most common techniques used to predict financial time series and which machine learning models are often applied. Additionally, the literature review also covered the UK's relationship with inflation, who is tasked with controlling it, how they attempt to do so, as well as some of the factors contributing to it.

The knowledge gained from completing the literature review will provide a strong backbone for solving the tasks presented in the implementation of the project going forward.

Chapter 3

Design

The design process for machine learning algorithms varies depending on the size, complexity, and use cases of the algorithm. According to Chip Huyen, author of the book "Designing Machine Learning Systems", the machine learning project flow has four distinct steps: project setup, data pipeline, modeling and training, and finally: serving. These four steps can be iterated through as seen in figure 3.1.

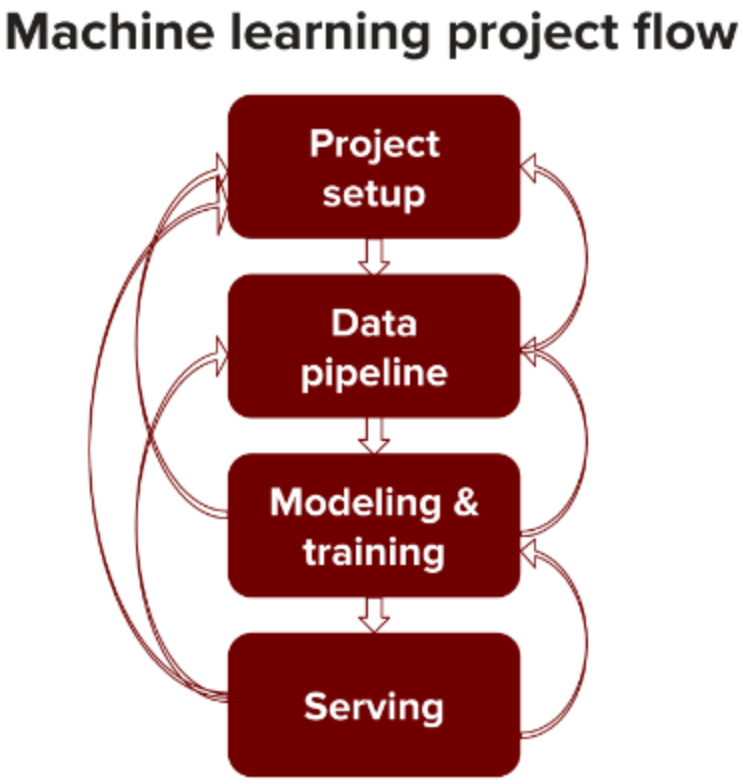


Figure 3.1: Machine Learning Design Pipeline by Chip Huyen.
Figure from https://huyenchip.com/machine-learning-systems-design/design-a-machine-learning-system.html[38]

This is the project flow that will be followed for this project and throughout this design chapter.

3.1 Project Setup

The project setup step of Heung's design flow mainly focuses on outlining the details of the project. Examples of these details are the projects: goals, user experience, evaluation, personalization, and constraints. This report's project setup shall focus mainly on the goals, evaluation, and constraints sections as user experience and personalization are not key aspects of the project.

3.1.1 Goals

As the title of the project outlines, the goal of the project is to predict inflation using machine learning. This project will evaluate the use of various machine learning methods and assess their ability to predict inflation. Further aims and objectives are outlined in Chapter 1. Introduction.

3.1.2 Expected Model Performance

As multiple models will be implemented in this project, it is unlikely that all models will perform to the same ability. Thus it is important to have a rough prediction as to which models will perform better or worse. By making these predictions it will prevent additional time from being wasted on optimizing or making changes to models that show less potential. This does not mean that these models should not be implemented properly but rather it means that the limited time for the implementation of the project should focus on the models with the best potential to predict inflation. Although all the models that will be implemented should be able to predict inflation to some degree, certain models have a higher likelihood of better performance. The models that are predicted to have the best performance are the LSTM and the FNN. This is because of their increased complexity and the fact that they can be modified more easily, giving more control over the architecture of the model. Of these two models, the LSTM is predicted to perform the best due to the fact it can memorise previous data which should help the LSTM to better analyse the patterns of a sequential time series such as inflation. The models that are predicted to perform worse are the linear regression model, the random forest model, and the support vector regression model. This is in view of their simplicity and the fact that inflation is a complex variable to predict and so is likely to require more complex models to create an

accurate prediction. This means that throughout the design and implementation, more time will be allocated to tweaking and improving the performance of the LSTM and FNN models as they are predicted to have a higher potential.

3.1.3 Evaluation

Evaluation outlines how the system's performance shall be evaluated: what metrics shall be used, what visualisations shall be produced, and so on. The models created in this project will be regression models attempting to predict an exact value of inflation. Thus the methods selected to evaluate the models should be suited for regression and continuous values. So methods like confusion matrixes that are better suited for classification would not be a good choice. As stated in the literature review, the most common regression metrics were MSE, MAE, MAPE, and R/ R^2. As each of these metrics is straightforward to implement, they will all be used for evaluating the models created. Furthermore, visualisations displaying each model's predictions alongside the values they were attempting to predict will be produced. This will aid the view in understanding the model's effectiveness beyond numerical metrics. Where appropriate, certain models should produce loss functions as these will be able to guide hyperparameter optimisation, such as selecting the correct number of epochs to train a model.

3.1.4 Constraints

The Constraints of a project can follow two veins: performance constraints and project constraints. Performance constraints address how a project must produce its results. Examples of performance constraints are how fast predictions should be made and how precise predictions should be. Of these two constraints, this project's main concern is with precision. The predictions of each model created will be evaluated and compared to one another in order to conclude the type of machine learning model best suited to predicting inflation.

On the other hand, project constraints are real-world constraints that may limit the project such as the time or manpower available. A typical project constraint may be the software available. However, since it is common practice to use Python for artificial intelligence research (even for billion-dollar companies such as Google[39]). Additionally, Python is a free-to-use, open-source language. The constraints of this project are mainly generic constraints that will apply to most final-year projects. One such generic constraint is the fact that only one student will be working on the project and the student has several other modules to focus on throughout the year. This means that a limited amount of time and resources will be dedicated to the project. In the context of predicting inflation with machine

learning, this means that it is extremely unlikely that any of the models produced by the end of the project will be more capable or effective than current models in use from large organisations such as the Bank of England. As these organisations will have dedicated more time, manpower, and resources toward producing models with greater sophistication and accuracy than those that could be created by a lone student. Even if models of similar complexity were to be created from this project, there would not be the hardware available to consistently and reasonably train and test such models. These factors mean that given the task of predicting inflation with machine learning, it is more suitable to create numerous models and compare their effectiveness rather than creating a single extremely sophisticated model that would require more resources to produce.

3.2 Data pipeline

The data pipeline section of designing machine learning algorithms deals with the selection and preprocessing of data.

3.2.1 The Systems Input and Output

One of the first questions that needs to be addressed is: "What is the system producing and from what information are these predictions being produced?" The input is real-world historical economic data relating to UK inflation. The output is an array of values predicting future inflation. As all of this data is publicly available the project should have very few privacy and bias concerns.

3.2.2 Data Selection

The saying "garbage in garbage out" succinctly illustrates the importance of selecting good data for machine learning models. It does not matter how powerful a model is if the data selected is poor or inappropriate. Selecting data is a key step to building effective machine-learning models. Thankfully, there are plenty of large open-source data sets available online, despite how time-consuming and expensive gathering data may be. Yet picking an appropriate set may still present a challenge. The criteria for a dataset for this project are as follows:

- A large number of data points (preferably in the thousands) as inflation is a complex feature to predict.
- Several related economic indicators to use as features for predicting inflation and to test their ability to predict inflation.

- Minimal missing or erroneous data as this data can lower a model's accuracy.
- Reliable data - the data should be taken from a reliable source in order to accurately predict real inflation.

During the project, several datasets were tested, for example, various Kaggle datasets and the "World Development Indicators" by the World Bank [40]. The dataset that was finally settled on was the UK financial statistics from OECD (The Organisation for Economic Co-operation and Development)[41]. The reason for selecting this set is due to its large variety of indicators, the reliable provider, and the fact that it is open-source (the dataset is classified as public under the access to information classification policy).

Initially, datasets from Kaggle were tested but they lacked a sufficient number of data points for multivariate machine learning analysis of inflation. Data from the World Bank was also preprocessed and tested with basic models. The World Bank is known for reliable and accurate data which should avoid issues of bias, or poor quality. However, the World Bank sourced a majority of its indicators annually dating back to the 1960s meaning that there would only be around 60 rows of data up to the present day. Thus, although the set had a large number of interesting features, the number of data points was insufficient for a machine-learning model. Additionally, due to the large number of features (over 1400), it would need a sufficient amount of cleaning/preprocessing in order to remove irrelevant features.

3.2.3 The Final Dataset

Due to the issues encountered with the other datasets, the OECD dataset was selected for use alongside monthly CPI data sourced from the World Bank. The first main benefit of the OECD dataset is that the data is taken monthly so there are a lot more data points than datasets that collect data annually. The features from the OECD dataset were taken from 1972 to 2022. Unfortunately, a few of the features had less data as they were taken from later than 1972 with the latest being taken in 1987. However, as this is only a small fraction of the overall dataset, replacing the missing values with zeros is unlikely to have a profound effect on the final results. The second benefit is that the dataset contains a moderate number of features to choose from (not too many that the features need to be drastically reduced (as was the case with the World Bank dataset)). There were twenty different features in total (not including CPI), these features were: composite business confidence, composite consumer confidence, composite leading indicator (CLI), consumer prices, hourly earnings, immediate interest rates

call money interbank rate, long-term interest rates, M1, M3, merchandise exports, merchandise imports, nominal exchange rates, passenger car registrations, production volume, GDP, retail trade volume, share prices, short-term interest rates, and unemployment Histograms were plotted for each feature in order to assess their distributions. Understanding the distribution of a feature can give some insight as to whether to use normalisation (rescaling the data between two set values) or standardisation (setting the mean of the data to 0 and the standard deviation to 1) before giving the data to the machine learning models.

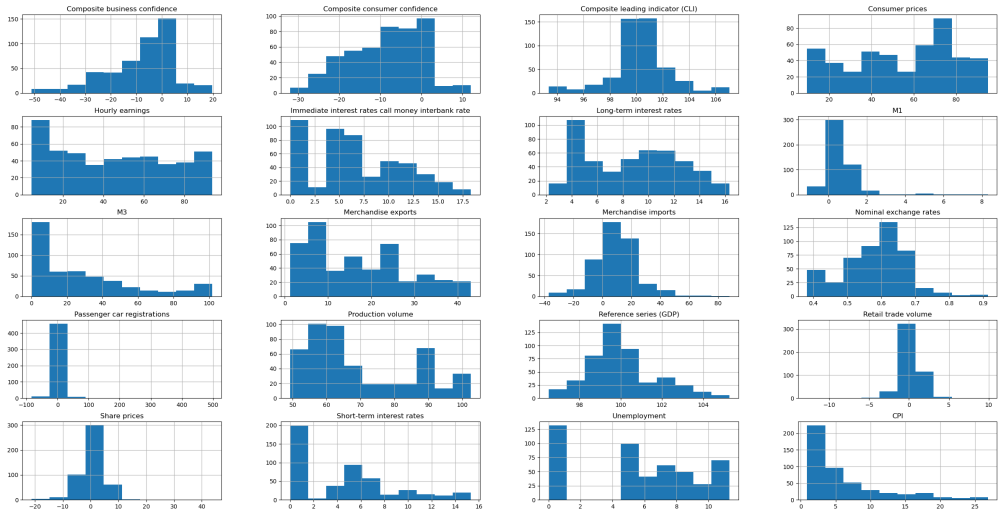


Figure 3.2: A histogram of each feature in the dataset

The histograms display a variety of different distributions, this is understandable as many time series do not follow a normal distribution. The distributions indicate that normalisation may be preferable to standardisation as normalisation is preferable when several features do not have a Gaussian (normal) distribution. This is because normalisation will preserve the distribution of the data by rescaling the data where as standardisation makes the assumption that the data has a Gaussian distribution and may modify the data's distribution if it is not already Gaussian. As the distribution of the dataset will remain non-normal, some precautions may need to be observed when using models that assume a normal distribution (such as linear regression).

In addition, a correlation matrix was formed for all of the features to determine if any features were overlapping. A correlation matrix gives us an indication of how the features relate to each other one on one.

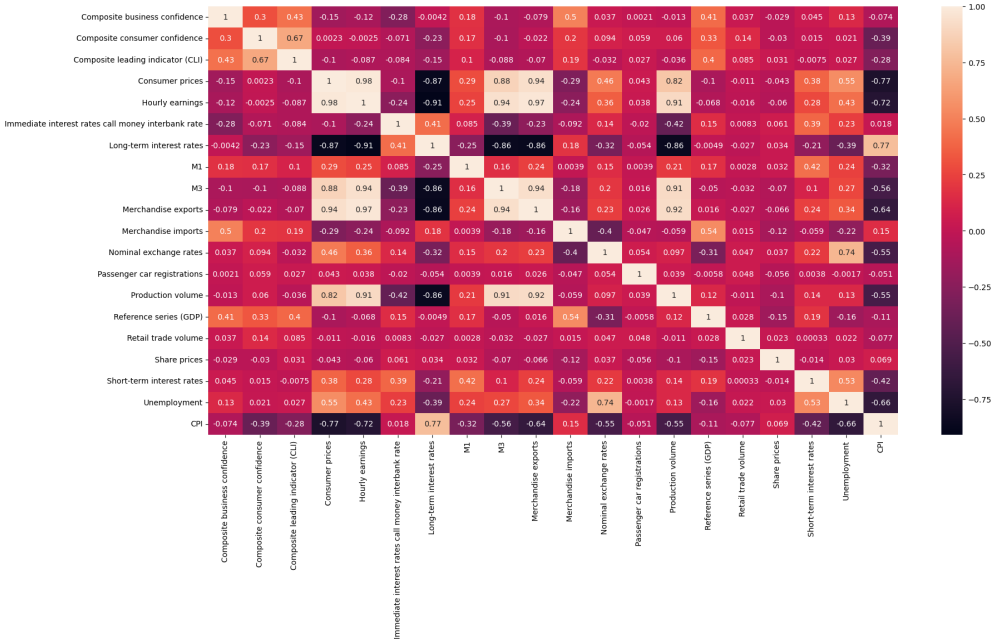


Figure 3.3: Correlation Matrix for the Features in the Dataset

Although, at first glance, the correlation matrix seems to show very little correlation between the features, this does not mean that the features are not correlated or cannot be used for predicting inflation. This is due to the fact that the heatmap figure was generated using a pairwise Pearson correlation function. Although pairwise Pearson correlation can often give a good representation of the correlation of two features it may not always be accurate. Pearson correlation attempts to draw a line of best fit between two features, meaning that if the two features do not share a linear correlation then Pearson correlation may produce a poor result. As the relationship between inflation and its causes is extremely complex, with many different factors contributing to the value of inflation in varying amounts, it is unlikely that inflation will have a strong linear correlation with the features that contribute to it. So despite the fact that the results of the heatmap were not ideal, this does not mean that the features are inappropriate to use in our models.

3.2.4 Data Pre-processing and Transformation

Often, datasets have several outstanding issues or properties that make them imperfect for use in a machine-learning model. Data pre-processing often includes steps such as data cleaning (handling outliers, noise, or missing values) and data integration (combining data from multiple sources). Data transformation is the task of molding the data to an appropriate size and dimensionality. Both of these processes are to create a dataset that is formatted in a manner that suits our models and objectives. Poorly processed data can be difficult for both machines

and humans to use and can lead to poor results.

Issues With The OECD Dataset

There were some outstanding issues with the original dataset taken from OECD and thus several steps were taken to clean the data. These issues included:

- Features containing missing data.
- Incorrect format of the dataset (required transformation).
- Duplicate features.
- Unnecessary/erroneous data.
- A large amount of metadata that needed to be removed.

Any data points that fell under these issues were removed/cleaned and all null values were replaced with zeros to improve readability.

Finally, a Granger causality test was carried out on the data to gather a better understanding of our data and its correlation to the target feature (CPI).

Granger Causality

Granger causality is a statistical concept in economics used to show if time series A is useful at forecasting time series B. Clive Granger originally proposed the test in 1969 in the article "Investigating Causal Relations by Econometric Models and Cross-spectral Methods[42]. The test only shows predictive causality and not true causality. Additionally, the test only provides information about forecasting ability and not the actual causal relationship. Another issue with the use of the Granger test in the context of inflation is that the test works best on stationary data. Whether inflation is best treated as stationary or non-stationary data is currently inconclusive[43]. But for our purposes, these limitations are fine as we only want to understand an indication of how useful the data will be for inflation forecasting.

Granger Testing The Data

With two indicators X and Y, X causes Y if a series of tests on lagged values of X produce a p-value of less than 0.05. The closer the p-value is to zero the more likely it is for X to granger cause Y. Lagged values are values from a time series shifted forwards or backward in time. In the case of the Granger test, Jeffery Woolriddge proposes that fewer lags should be used for annual data compared to quarterly or monthly data in order to not lose degrees of freedom[44]. The

Granger test was run on the preprocessed indicators comparing each of them to CPI. The function ran 12 lags to see if data has a potential causal relationship with inflation within the past year. This means that if at some point during the last 12 lags a p-value¡0.05 was produced the data will be labeled as having the potential to be useful in forecasting CPI. Of the 20 features tested, 9 produced results that indicate Granger Causality. These 9 were: 'Composite business confidence', 'Composite consumer confidence', 'Consumer prices', 'Long-term interest rates', 'Merchandise imports', 'Reference series (GDP)', 'Retail trade volume', 'Share prices', 'Unemployment' This indicates that almost half of the features have the potential to predict CPI. This, however, does not mean that the 9 features found to Granger Cause CPI definitely cause inflation as the Granger Causality test is only a measure of the potential of one value to cause another. Furthermore, just because a value does not Granger Cause inflation does not mean that the feature is not useful in predicting inflation. We now have a better understanding of the potential our dataset has to predict inflation.

3.3 Modelling and Training

3.3.1 Model Selection

The models that will be compared for their ability to predict inflation are a random forest model, a support vector regression model, a linear regression model, a long short-term memory (LSTM) model, and a feedforward neural network (FNN) model. Most of the models were discussed during the literature review and will be implemented due to their common use in similar regression models.

Linear regression was added to the list of models to be implemented as it is an extremely simple model that focuses on plotting a single line of best fit through the data. Including linear regression will provide a comparison to the results of a simple model, as well as potentially indicate any unexpected linear relationships between inflation and any of the dataset's features.

A long short-term memory network will also be implemented. LSTMs are a type of recurrent neural network (RNN). As mentioned in the literature review, the architecture of RNNs enables them to preserve an internal memory state that helps them to model temporal dependencies in the data. This characteristic means that RNNs excel at processing sequential data such as time series data. The LSTM was selected as the RNN model that will be implemented as they are considered to be one of the most powerful implementations of an RNN and are used to find patterns in numerical sequence data[45]. As the values of inflation are time series data, an LSTM could potentially provide good results despite the seeming lack of

literature relating LSTMs to economic time series prediction.
Several of the models selected have implementations provided through the use of libraries such as Scikit-learn[46]. Implementing a model through a high-level library lets the user create an already well-documented and tested model with only a few lines of code. This saves both time and resources (as the model implemented from the library is likely to have been optimised) while also providing additional support, if necessary, from the community and documentation.

3.3.2 Model Training

The models will be created using Python and several of its supported libraries and written in a Jupyter Notebook. The reason for using a Jupyter Notebook is that the cell structure allows for the steps to be run separately and the results to be displayed in real time without the entire file being run. This makes experimenting with hyperparameters and training models much more convenient. As the results of each cell are displayed separately this also improves the ability to quickly create and display visualisations through tools such as Matplotlib [47]. Jupyter also supports multiple languages and libraries enabling users to work in the preferred environment.

3.4 Serving

Serving is the process in which the models and their results are presented to the end user. As this is an academic project this report will act as the presentation method. This means that both the model and their results need to be suitably presented within a report. This will be done through the use of graph visualisations created in Matplotlib. These visualisations should display and compare the model's predictive ability along with their learning rates. This will be done by creating a graph for each model that displays their predictions compared to the actual values of inflation, indicating how well they can predict both training and testing data. Loss graphs will be used to show each model's learning rate over time.

Chapter 4

Implementation and Testing

The goal of the project is to implement a machine-learning model that has the ability to predict inflation. In order to find the most suitable model, several different models were implemented and their results were compared. This chapter of the report will cover: which models were implemented, how they were implemented, how the models were tuned and tested, the results produced by the models, and some of the issues encountered in the implementation process. This chapter will not contain the evaluation of the models' results or a comparison of the results.

4.1 Tools

4.1.1 Coding Environemnt

The IDE used for this project was Jupyter Notebook provided through the use of Anaconda. Anaconda is a platform that includes a packet manager, this makes it easy to control packages and their dependencies that are installed in an environment. Each environment in Anaconda is isolated which allows version control and preventing conflicts between the installed versions and packages of different projects. This is valuable when working on multiple projects simultaneously. Furthermore, Anaconda provides Jupyter Notebook integration which is beneficial as it is a common IDE for data science and has an active community making it easy to learn.

4.1.2 Visualisation

Jupyter Notebook provides visualisation as output from individual cells. This was useful for debugging throughout the project. Both Matplotlib and Seaborn were used for further graphical visualisation.

4.1.3 Libraries

The project's code was written in Python and several libraries were used to assist the coding process.

NumPy[48] and Pandas[49] - These libraries were used for data manipulation and cleaning. NumPy provides several tools for dealing with multidimensional arrays. Pandas is built upon NumPy and is used for dealing with tabular data (data that is organised into a table with rows and columns). Pandas also has built-in tools for dealing with time-series data which is useful as inflation is time-series data.

TensorFlow[50] and Keras[51] - TensorFlow is a platform developed by Google for training machine learning models. It makes defining a model straightforward while still providing enough control and flexibility of the model's structure, inputs, and outputs. Using pre-written functions and nodes that were written by a reliable source both saves time and resources during development. Keras is built on top of TensorFlow and aims to provide a simplified and more user-friendly interface for creating and training machine learning models.

SciKit-Learn[46] - Sklearn is a machine learning library that provides a wide range of easy-to-implement machine learning models and metrics. Its extensive documentation makes it easy to learn and implement.

Matplotlib[47] and Seaborn[52] - Matplotlib and Seaborn were used for all graphical data visualisation. These libraries provide a plethora of graphing options along with plenty of community support.

Sktime[53] - Sktime is a library for machine learning with time series data. This was used to implement the univariate time series forecasting of inflation.

4.2 Data Preperation

Features relating to inflation were sourced from the OECD Databank and Monthly CPI indicators were taken from the World Bank's online statistics. All of this data is relating to the United Kingdom's general economic indicators. The data was compiled into a tabular format to facilitate the use of the Pandas library and to streamline further data manipulation tasks. By using a pandas dataframe several functions can be used to quickly assess the state of the data such as the head or shape functions. This is useful in the context of machine learning as model layers

require inputs of certain types or shapes.

4.2.1 Maintaining Time Consistency

Using time series data in a machine learning model requires additional attention compared to regular data, especially when testing the model. General practice in machine learning is to split the dataset into arrays for training and testing and into X and y (with X being the data that predicts the target data y). However, in time series forecasting the data is created or released over time. This means that using X data from the year 2000 to predict the value of inflation from the same year is redundant as the inflation data will have already been released around the same time as the X data meaning that it is already available. Instead, the aim of time series forecasting should be to use past data to predict future data. In creating a dataset for a machine learning model this takes the form of shifting the target forward in time so that the X data from 2000 is being used to predict the y data from 2001. Often, the further forward the target data is shifted the less accurate the model becomes. In this project, the target values of inflation were shifted 12 months ahead of the X data. This means that the model uses data from the current year to predict the inflation values of the next year. Additionally, a feature was implemented in the project where the user can adjust the number of months that inflation is shifted to provide additional predictive flexibility if needed.

4.2.2 Normalising the Data

Normalisation, also known as feature scaling, is used to rescale data between a given range. The importance of normalisation is due to the fact that many of the features may have varying ranges, this could lead to features with a large range having a greater impact on the result. By adjusting all of the features to range between the same values, no single feature will dominate and all features should contribute equally to the final result. Additionally, normalisation increases the speed at which gradient descent converges, which will decrease the runtime of the models.[54] Normalising the data is often done either in the range [0,1] or [-1,1]. The dataset for this project was normalised between the range of [0,1] with the use of the MinMaxScaler from the sklearn library. There are several benefits to scaling to a small range such as reducing the impact of outliers and increasing the speed of the algorithms.

4.2.3 Splitting the Data into Training and Testing Splits

It is common practice to split a dataset into training and testing splits. The training split is then used for training the machine learning model and the testing split is used for testing the model's performance on unseen data. If the model was tested on data it had already been trained on it would become difficult to evaluate the model's generalization (its ability to perform on new/unseen data). The difficulty in deciding the size of the training and testing splits lies in the fact that too small a training split and our model may not perform well but too small a testing split and our performance metrics will have a greater variance. The project dataset was split into 70% training data and 30% as this is a common split. The models will not utilize a validation set. As the dataset is made up of time series data, splitting the data carelessly can lead to suboptimal results. This is because time series data often has temporal dependencies. Temporal dependency is when data contains stronger associations between events that happened within the same time period. To preserve the temporal dependency of the data the dataset will not be shuffled (randomised) when it is split into the training and testing sets. The dataset was split using the "train_test_split" function from sklearn.

4.2.4 Inflation's Historic Outliers

The history of inflation is riddled with outliers caused by unexpected or unprecedented events. Covid-19, or Russia's declaration of war on Ukraine to name a few. However, the UK's largest spike of inflation in recent history happened in the 1970s due to spikes in oil prices and a steep rise in wages. The result of this is inflation reaching as high as 25%. Although the purpose of predicting inflation is to also be able to foresee such large events, including an outlier of this magnitude in our training data will likely harm our model's predictive ability as well as its generalization.

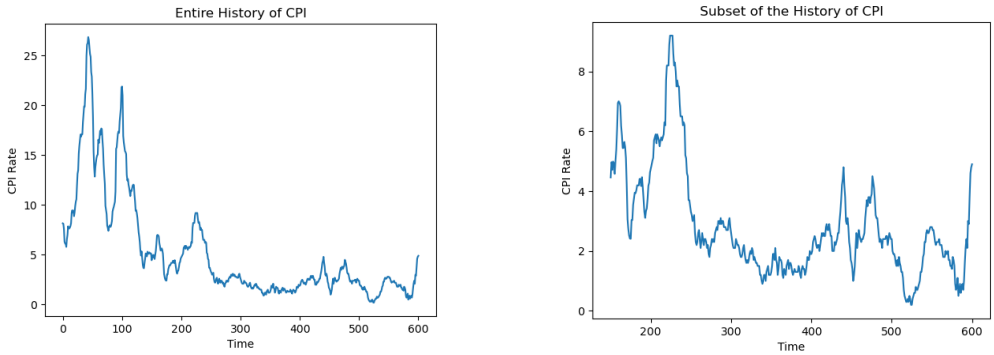


Figure 4.1: History of CPI vs History of CPI Excluding the First 150 Datapoints.

The graphs in figure 4.1 As such, after training and testing the models on the entire dataset and comparing the results to those of models trained on a subset of the data, the decision was made to remove the first 150 data points from the dataset. This would remove the large outlying period of the 70s while still retaining some peaks in inflation in order to give the model a realistic expectation for how inflation may peak again. Removing the first 150 data points reduces the size of the dataset by a significant portion, however, the subset of data still produced more accurate results when testing the models.

4.3 The Machine Learning Models

Five different models were implemented and tested for the predictive ability on the same data set. The different machine learning models were linear regression, random forest, support vector regression, long short-term memory network, and a feedforward neural network. The reason for these five models was because they are some of the most common models used for regression analysis and, as found in the literature, many of the models are used for economic indicator analysis (such as stock predictions). Three of the five models were implemented with the Scikit-learn library. These were linear regression, random forest, and support vector regression. The remaining two models were implemented through the use of TensorFlow and Keras.

4.4 Univariate Implementation

Univariate analysis is the practice of analysing a single variable to understand its patterns and characteristics. Univariate analysis does not involve relationships or causes from other variables. Before Implementing each machine learning model with the full dataset, a subset of the models were trained and tested using inflation as the only variable. The models that were implemented were linear regression, random forest regression (using 20 trees), and support vector regression. This was a quick implementation to test if predicting inflation solely based on its past values would suffice. As expected, the results produced were lackluster, likely due to inflation's complexity and the fact that many outside factors affect it. Of the three models, random forest regression produced the best results although the quality of the predictions was still not satisfactory.

4.5 Multivariate Implementation

Multivariate analysis is the practice of compiling multiple unique data variables in order to create a more holistic forecast of a related variable.

4.5.1 The Features and Dataset

The curated dataset that was used for this project contained 20 different features, each with monthly data dating back to 1972 at the earliest and 1987. The OECD features are macroeconomic indicators that focus on broad trends from the UK with global implications. Macroeconomic features were selected as inflation itself happens on a macro scale thus it stands to reason that macro features will likely predict inflation better than micro ones.

Upon analysis of the features selected for the dataset (as seen in figures 3.3 and 3.2) very few features share a strong correlation or similar distribution to inflation. This could be worrying as it may indicate that, although on an intuitive surface level, many of these features should contribute to inflation, the features may not possess a strong ability to predict inflation. However, this is not the case as explored in the previous design chapter, the Pearson correlation calculated in the correlation matrix and the distribution of data shown in the histograms are not the be-all and end-all of value forecasting. This is demonstrated in the Granger causality tests carried out in the Design chapter as several features are shown to have the potential to predict inflation. Therefore, as the indication as to whether or not any of the features are redundant, all of the features will be kept in the dataset for training the models.

4.5.2 The Models Used for Multivariate Analysis

All five of the previously stated models (see 3.3.1 Model Selection) were implemented for multivariate analysis. Linear regression, random forest regression, and support vector regression were implemented with the help of the respective sklearn libraries. Long short-term memory network (LSTM) and the feedforward neural network (FNN) were implemented with the use of TensorFlow and Keras. The feedforward neural network, somewhat expectedly, produced the best predictions. The results of each model will be explored further in the following chapter.

4.5.3 Model Architecture

The architecture of a model is the structure of the model including the types of layers, number of nodes, and connections between layers as well as the input and

output of the model. This subsection will cover the architecture of the LSTM model and the feedforward neural network model. The linear regression, random forest regression, and support vector regression models will not be covered in this section as they are algorithm-based models. The parameters used for these three models were sklearn's default recommended parameters outlined in the sklearn documentation.

Feedforward Neural Network Model Architecture

An artificial neural network (ANN) is a machine learning model inspired by the human brain. ANNs feed data through layers of interconnected artificial neurons (nodes) in order to compute complex problems. ANNs can vary in size and shape as well as the different types of nodes they contain. The feedforward neural network (FNN) was implemented as one type of ANN. After some research and experimentation with various structures, the structure shown in figure 4.2 was implemented.

Layer (type)	Output Shape	Param #
dense_486 (Dense)	(None, 64)	1344
dropout_324 (Dropout)	(None, 64)	0
dense_487 (Dense)	(None, 64)	4160
dropout_325 (Dropout)	(None, 64)	0
dense_488 (Dense)	(None, 1)	65

Total params: 5569 (21.75 KB)
Trainable params: 5569 (21.75 KB)
Non-trainable params: 0 (0.00 Byte)

Figure 4.2: FNN Model Architecture Summary.

Figure 4.2 shows the structure of the FNN model, including each layer's output shape, and parameters. The layer column of the figure indicates which types of neural network layers were implemented. The output shape column states the shape of the data, this also indicates the number of nodes in a layer. After hyperparameter optimisation, it was found that for this particular model, sixteen nodes per layer produced the best results. The params column is the result of the inputs * weights + bias.

The structure of the FNN model contains three dense layers (an input layer, a hidden layer, and an output layer) each with a dropout layer in between. A dense layer is a layer where every neuron in the layer connects to every neuron in

the previous layer. Each dense layer contains an activation function. The purpose of activation functions is covered in the 'Artificial Neural Networks' section of the literature review. The hidden layers of the network utilise ReLU activation functions while the output uses a linear activation function. These functions were selected based on the advice offered in the article 'How to Choose an Activation Function for Deep Learning' by Jason Brownlee, PhD[55]. The article states the most common activation function for hidden layers is the ReLU function due to being simple to implement and being less susceptible to vanishing gradient issues compared to other functions. The output layer uses a linear activation function. The linear activation function allows the model to produce an output that is a linear combination of its inputs. This does not introduce any additional non-linearity as the goal is to predict a continuous value so there is no need to apply constraints through another activation function.

Dropout layers set the input values of some nodes in the layer to 0 at the specified dropout rate. Setting a limited number of inputs to 0 or 'dropping' them prevents the model from overfitting (becoming too familiar with the training data and losing generalization). However, the dropout rate needs to be monitored as a dropout rate that is too high may lead to a model that trains poorly on the data. Dropout rates are only applied during the training process to prevent overfitting, they are not applied during the testing or prediction process. After hyperparameter optimisation, the most successful dropout rate was found to be 0.1 (or 10%).

The final dense layer only contains an output shape of 1, this is the predicted value of inflation given the inputs.

LSTM Model Architecture

An LSTM network is a type of recurrent neural network that is designed to excel in predicting sequential data (including time series data). The LSTM for this project was implemented through the use of TensorFlow and Keras.

```
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_9 (InputLayer)        [(None, 12, 20)]          0

 lstm_16 (LSTM)              (None, 12, 32)            6784

 lstm_17 (LSTM)              (None, 32)                8320

 dense_8 (Dense)             (None, 1)                 33

=================================================================
Total params: 15137 (59.13 KB)
Trainable params: 15137 (59.13 KB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

Figure 4.3: LSTM Model Architecture Summary.

Figure 4.3 shows the structure of the LSTM model, including each layer's output shape, and parameters. The input shape of the LSTM differs from that of the FNN. This is because the FNN takes just one row of the X and y variables (X is the features being used to predict and y is the CPI value from the same time period of the given features). Whereas the LSTM model takes 12 consecutive rows of data, this is the equivalent of taking in a year's worth of data all at the same time. The reasoning for this is that the LSTM model contains hidden cells that store past values in order to predict future values. This should, in theory, improve the LSTM model's ability to predict sequential data.

The LSTM was not coded with any additional activation functions. This is due to the fact that the Keras' LSTM implementation already contains multiple activation functions. This is due to the hidden state and the cell states of each LSTM cell both containing activation functions (sigmoid and tanh respectively). The hidden state contains the memorized past data and the cell state contains the current data being given to the cell. Thus adding an additional activation function to the output is not necessary as there is already ample non-linearity being implemented.

Furthermore, the final LSTM model did not include dropout layers as when tested these had a significant negative impact on the results produced. This is possibly due to the fact that dropping values in an LSTM may break down temporal dependency relationships.

The final layer of the LSTM is a dense layer. This layer serves as the output by compiling the results from all the nodes in the previous layer into a single node.

4.5.4 Hyperparameter Optimisation

Both the LSTM and FNN underwent hyperparameter optimisation in order to produce the best performance for each model. The hyperparameter optimisation was done through a grid search technique. The grid search method is done by exhaustively searching a manually specified set of hyperparameters. The purpose of optimising the hyperparameters in this way is to ensure that the hyperparameters are the best possible values within the given subset. The downside to optimising hyperparameters using a grid search technique is that it is computationally expensive. This is because the model needs to be trained and tested on every possible combination of hyperparameters in order to ascertain which combination is optimal. Grid search is not the only algorithm for hyperparameter optimisation, alternatives to the grid search method are a random search algorithm and a gradient descent algorithm. Random search randomly samples combinations of parameters so it is less computationally expensive but it is unlikely to produce an optimal combination. Gradient descent optimisation updates hyperparameters to minimise the result of a loss function but there is a chance of the algorithm getting stuck in a local minimum which may not be the global minimum of the function, thus producing a sub-optimal result. Because of these factors, the grid search technique was chosen because it is more comprehensive than the other algorithms and is guaranteed to produce the optimal combination of hyperparameters from the given set.

The hyperparameters that were optimised for the FNN model were the number of nodes per layer, the dropout rate, the learning rate, and the size of the batches. For the LSTM model a smaller subset of hyperparameters were used these were the number of nodes per layer, the learning rate, and the size of the batches.
The dropout rate is the rate at which during training a random node will be dropped (have its inputs set to 0). The LSTM model did not have dropout functions so there was no need to include dropout probability as a hyperparameter.
The learning rate dictates the step size while moving towards the minimum of the optimizer loss function. Both models used the ADAM optimizer provided by Keras as it is computationally efficient and does not require lots of memory (which was limited as the models were being run from a standard laptop).
The batch size is the number of rows of data given to the model at once, using smaller batches can speed up the training time of a model as it only has to store the error values for a subset of the data instead of all of the data. Additionally, small batch sizes have been shown to increase the generalization performance of a model.

The set of hyperparameters tested in the grid search were as follows: number

of nodes per layer = [16,32,64,128], dropout rate = [0,0.1,0.2], learning rate = [0.01,0.005,0.001], batch size = [8,32,64]. The values were selected as they are commonly used or default settings for these hyperparameters. More values could be tested but this would take up exponentially more time and processing power in order to compare every combination.
For the FNN model, it was found that the ideal hyperparameters were 16 nodes per layer, a dropout rate of 0, a learning rate of 0.005, and a batch size of 32.
For the LSTM model, it was found that the ideal hyperparameters were 32 nodes per layer, a learning rate of 0.01, and a batch size of 8.
These hyperparameters were used as the default for the respective models going forward.

4.6 Testing and Model Results

4.6.1 Testing Objectives

Before beginning testing it is beneficial to understand what the objectives of the tests are. The main objectives are: identifying any errors or bugs in the code, ensuring the results are of an acceptable standard, and identifying potential areas of improvement. Identifying and analysing areas of improvement will be carried out in Chapter 5.

4.6.2 The Testing Set

The testing set is the set of previously unseen data that the models will be tested on. The set was made up of the final 20% of the original unshuffled data. As the testing set was not shuffled, it provides a better perspective on how the models will perform for future inflation data as it will also be sequential.

4.6.3 Identifying Errors

Throughout the implementation process, the code was continuously tested to ensure it was working as expected. This included both unit testing (testing a single piece of code) and integration testing (testing that multiple pieces of code work together as intended). Continuous testing was made easier by the fact that Jupyter Notebook allows the user to divide their code into cells and run the cells individually. Being able to see the output of an individual cell makes unit testing much faster as the entire code does not need to be run. This is particularly true for machine learning projects as training the algorithm can often be time-consuming.

When dealing with the errors found by the continuous testing, documentation was often the first port of call. Due to the well-known nature of the libraries this project utilised, documentation was thorough and well-written, making identifying and understanding errors much easier. Furthermore, most of the libraries in use such as Sklearn or TensorFlow have active communities that help provide solutions through websites such as stack overflow. This meant that the errors produced during the implementation of the project had already been identified and solved by other users, once again making the implementation process much smoother.

4.6.4 Quality Assurance

Another purpose of testing is to ensure that the outputs of the code are of a high enough quality regardless of whether they are the correct output or not. This is particularly important as the value of a machine learning model rests on the quality of results it produces. For quality assurance testing, several regression metrics were used such as MAE, RMSE, MAPE, and R-squared. These metrics evaluate the effectiveness of the models. The metrics were calculated using the testing set against the models' predictions of the testing set. The ideal values for each metric are MAE=0, RMSE=0, MAPE=0, R-squared=1.

Models	MAE	RMSE	MAPE	R2
Linear Regression	0.1526	0.2611	9188948002912	-4.4515
Random Forest Regression	0.1040	0.1272	1949910168334	-0.2941
Support Vector Regression	0.1537	0.1784	2210975015699	-1.5471
Long Short-Term Memory	0.0504	0.0642	5815110419921	0.7082
Feedforward Neural Network	0.0225	0.0304	4617012820526	0.9207

Of the models shown, linear regression performs the worst and the feedforward neural network performs the best. However, there are some strange outliers within the regression metrics. To understand these outliers better there will be a brief overview of each metric.

MAE (Mean Absolute Error) measures the sum of absolute errors divided by the number of observations. The value of MAE is a positive real number.

RMSE (Root Mean Square Error) is the square root of the MSE which is the average of the squared difference between actual values and predicted values. RMSE was used as it is easier to interpret than MSE because MSE is in squared units. An RMSE value will be a positive real number.

MAPE (Mean Absolute Percentage Error) is the average of the absolute percentage error of each observation. MAPE values are on a scale of 0 to 1 with 0 being a 0% deviation between the predicted and the actual values. Yet all of the models,

including the two best-performing models, produced exceptionally high MAPE values despite the fact that they would be expected to produce the lower values. The reason for this is that if values are 0 or close enough to 0, it can cause the MAPE calculation to divide by 0 which is undefined. The sklearn metric deals with this by producing an arbitrarily high output. Including MAPE as a metric proves to be ineffective in this case as normalising the data between the the range [0,1] most likely caused this error.
Another of the errors appears in the r-squared collum, where it produces negative values for some of the models. R-squared measures the proportion of the variance for a dependent variable that is predicted by an independent variable, this is also known as the coefficient of determination. Despite what the name suggests, R-squared is not always squared so a negative value is not necessarily an error. Rather, the R-squared metric produces a negative value when the model's fit is worse than the fit of the mean of the data.

Ensuring the Models Do Not Over Train

Another part of quality assurance in machine learning is ensuring the models do not overtrain. Overtraining is when a model learns the training set so well that it impacts the model's overall generalisation. This is obviously negative, as the model's job is not to predict data it has already been given, but rather to predict data it has never seen before. Deciding the number of epochs to train the model on requires a balance between a high number of epochs which can produce better results but will take more processing time and potentially cause overtraining. During the implementation of the models, a few methods were used to ensure that overtraining was not happening. The first method was producing a loss graph that tracks the loss value over the number of epochs. The loss should gradually fall over time, however, if the loss begins to rise it can be an indication of overtraining.

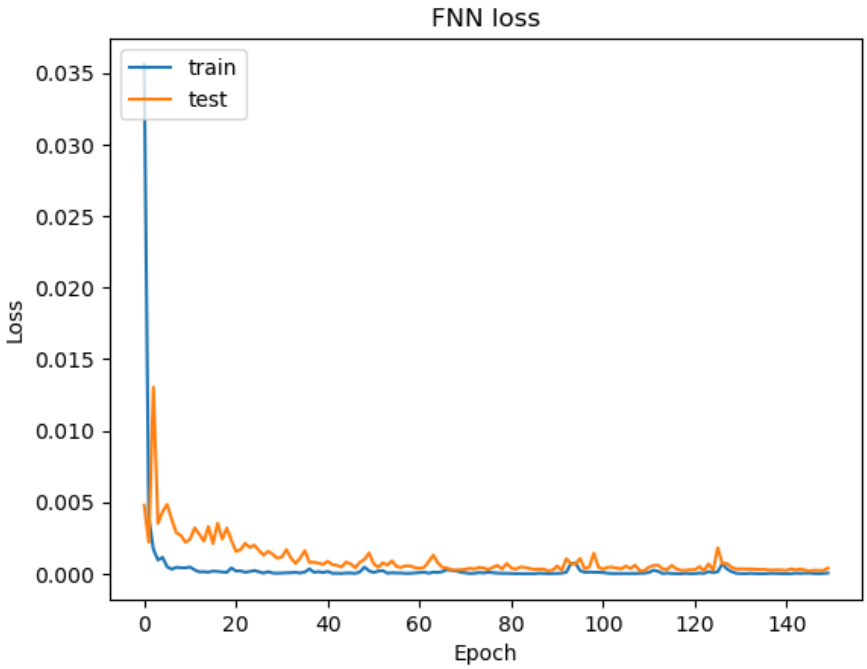


Figure 4.4: FNN Model Loss Over Time (epochs).

By observing 4.4 it can be seen that the loss plateaus after 50 to 60 epochs. This means that the model could be trained for much fewer epochs while still producing similar results and saving time. After the loss plateaus, at no point does it begin to rise, therefore we can assume that the model has not overtrained.

The second method used to test for overtraining was by making the model predict the training data after it had already been trained. This will test the model's performance on data it has already seen. If it has not overtrained then it is likely that the model will not be 100% accurate. If the model predictes the training data with near 100% accuracy it may have overtrained. This method is less accurate than the previous method, however, it is quick and easy to produce after training the model.

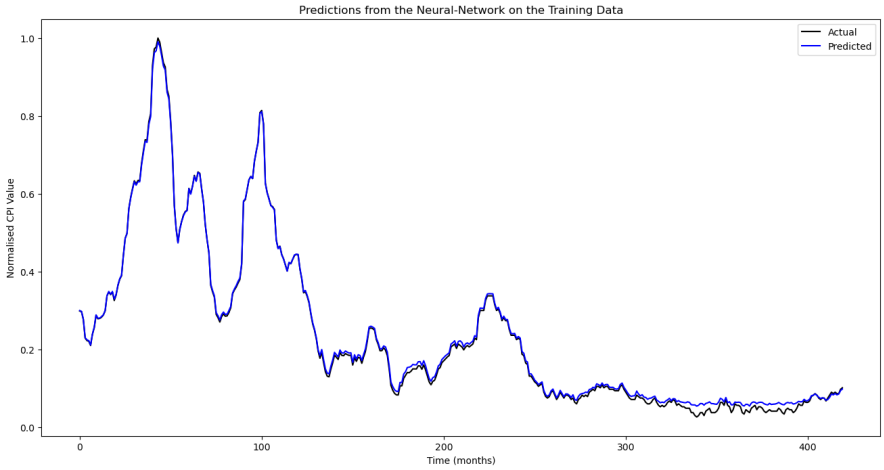


Figure 4.5: FNN Model's Predictions of Seen Data.

Figure 4.5 shows that the model was extremely effective at predicting the initial portion of the training data, but its accuracy tapers off towards the end of the data. This is possibly because the final portion of the training data has much finer increments in change compared to the large movements at the start of the data. The model is likely struggling with the change in momentum between the highly volatile first portion and the smaller changes in the second portion of the data. However, as the model does not seem to be able to predict the training data with nearly 100% accuracy, then it is further indication that the model is unlikely to have overtrained.

Likely due to the complexity of the data, the models could be run for a relatively large number of epochs without overtraining. Overtraining began occuring around 300 epochs, furthermore training the models for this many epochs became increasingly time-consuming whilst using the grid method for hyperparameter optimisation. Ultimately, the models were each trained for around 100 epochs as this seemed to be the sweet spot between saving time and producing results that are towards the higher end of what the model is capable of.

4.7 Final User Friendly Model Implementation

Although further comparisons and evaluation will be completed, based on the regression metrics taken the feedforward neural network produced the best results for predicting inflation. As such a more user-friendly implementation was created using this model. The implementation consists of a checkbox list of the dataset's features. These features can be selected and then a model will be trained and tested using the selected features. Additionally, the number of months the X data is shifted from the target value can be altered using a slider. If the slider is set

to 12 then the dataset will be aligned so that the X data is predicting the target values 12 months in the future. This implementation allows for more flexibility in producing predictions, however, a large majority of the predictions will be worse than the default settings of the model.

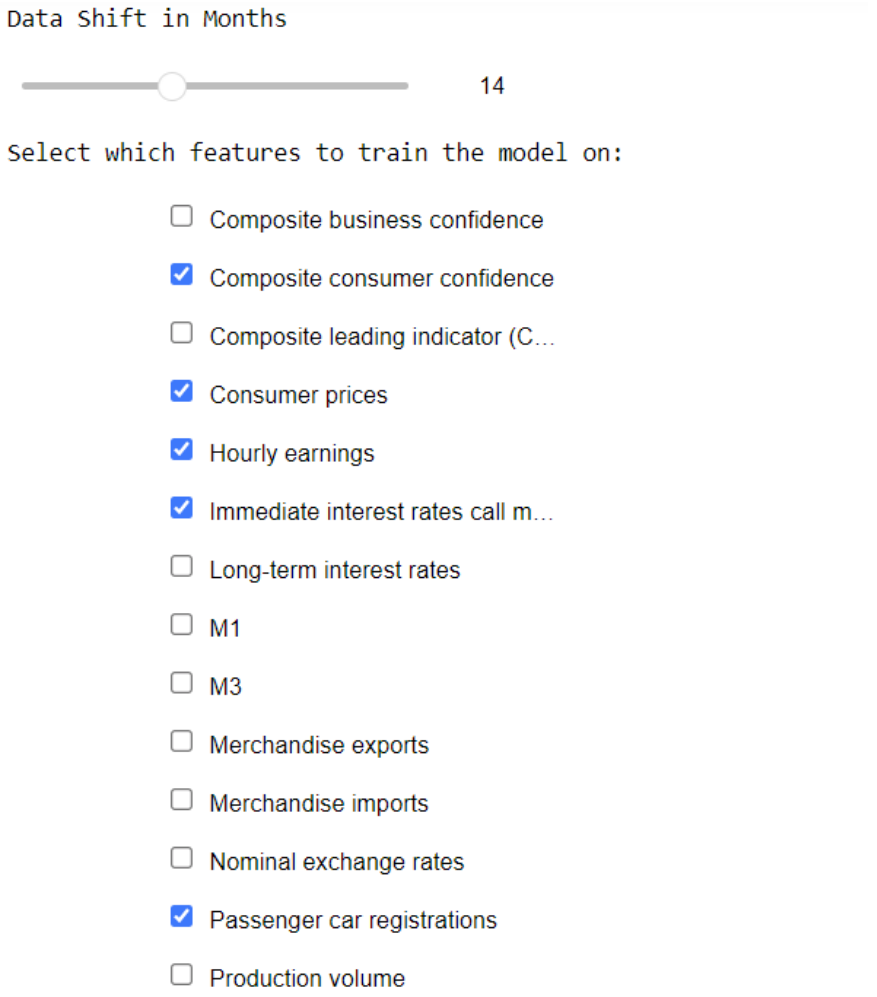


Figure 4.6: A Screen Shot of a Section of the Input Screen for the Adjustable Model.

Figure 4.6 shows part of the input section of the adjustable model. After running these inputs the model will output a graph of the predicted values of inflation compared to the actual values along with a loss graph from the model's training.

Chapter 5

Results and Evaluation

This chapter covers the overall evaluation of the products developed. This includes a comparison and evaluation of the results produced by the models as well as a reflection of what could be done differently if the project were to start over.

5.1 Results From the Univariate Implementation

The Univariate implementation was only implemented on linear regression, random forest regression, and support vector regression. This was because it was unlikely for the results of the univariate analysis to be of a high standard due to the complexity of inflation as such it was deemed unnecessary to implement any more models.

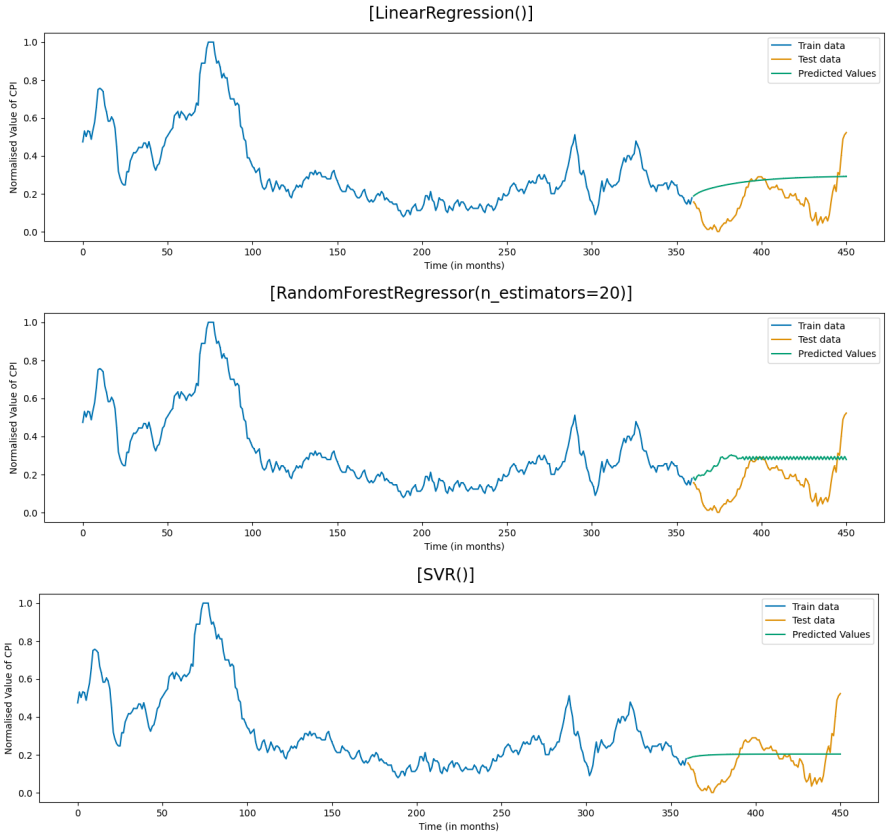


Figure 5.1: Univariate Implementation of Linear Regression, Random Forest Regression, and Support Vector Regression.

As expected, the results from attempting to predict inflation solely based on its history were extremely poor. None of the three models implemented produced predictions that could be helpful in any capacity. These results serve to cement the complexity of inflation and how it is affected by many outside factors, thus making it necessary to use a large range of relevant features in order to produce an accurate forecast.

5.2 The Multivariate Models' Results

5.2.1 Linear Regression Model, Random Forest Regression Model, and Support Vector Regression Model Predictions

The table of regression metrics in the previous chapter showed how the linear regression, random forest, and support vector regression algorithms all performed worse prediction methods than the mean of the data (as shown in the r-squared score). Although this was a disappointing result, it was somewhat expected as in-

flation is an extremely complex variable and simpler algorithms will likely struggle to form an understanding of its properties.

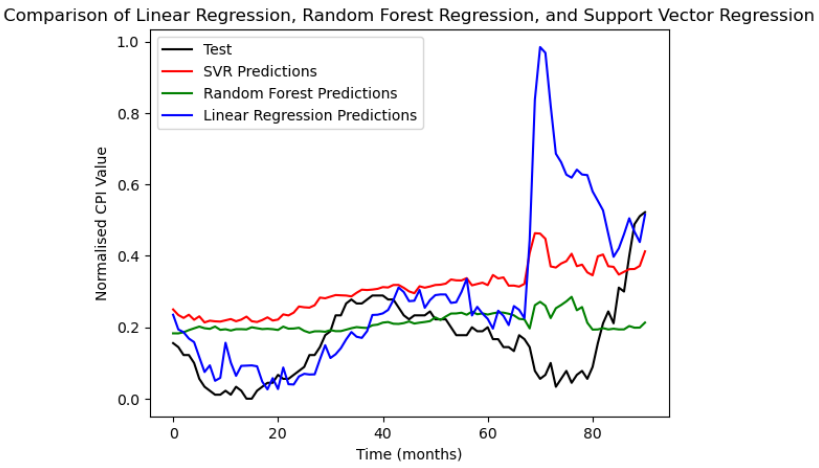


Figure 5.2: A Comparison of the Predictions of Linear Regression, Random Forest Regression, and Support Vector Regression Models.

Figure 5.2 shows the predictions of each model compared to the actual normalised value of inflation (shown in black). It can be seen that all three models struggle to accurately predict inflation. Interestingly, around the 65th-70th month mark, all three models predict an increase in inflation where there is a decrease. This could be due to a linear property indicating that inflation was likely to increase, however, possibly due to factors outside of the feature set, the increase was held off.

It was expected for linear regression to struggle in predicting inflation as the relationship between it and the other features is complex. This was shown in the correlation matrix created that showed very little Pearson pairwise correlation between CPI and the other features. Although it has to be said that despite linear regression not being expected to perform well, the predictions graph still outperformed random forest regression and support vector regression if the large outlier around month 70 is ignored.

Random forest models seemed to have a good reputation for predictions during the literature review yet still failed to provide an accurate prediction for inflation. This is possibly due to the fact that each tree in the random forest model is trained on a random subset of the data. This may not be a good method for predicting inflation as it is sequential data so training random sections of the data on different trees likely removes any temporal dependencies present in the original data.

Support Vector Regression (SVR) predicts data by creating a hyperplane. However, it is difficult to fit a hyperplane on data with a high number of dimensions or lots of noise. Both of these caveats apply to the dataset used to predict

inflation as it contained many different features each likely to have noise of some kind such as price fluctuations. Furthermore, although SVRs create a hyperplane they are still made to work on linearly separable data so if the data contains too many complexities and non-linear relationships then SVRs can encounter issues in providing accurate predictions.

Overall, these three models are likely inappropriate use for predicting inflation when using a complex dataset.

5.2.2 LSTM Network Model Predictions

According to the metric testing, the LSTM's results were an improvement upon the results of the previous three models, however, it still struggled to accurately predict inflation. This is seen further in the graph comparing the LSTM's predictions to the actual values. The predictions fail to predict the more erratic highs and lows of inflation, instead producing a much smoother curve. This could potentially be due to the number of past values the LSTM is given. As it stores 12 past values it may struggle to predict the sharp gains and losses of inflation.

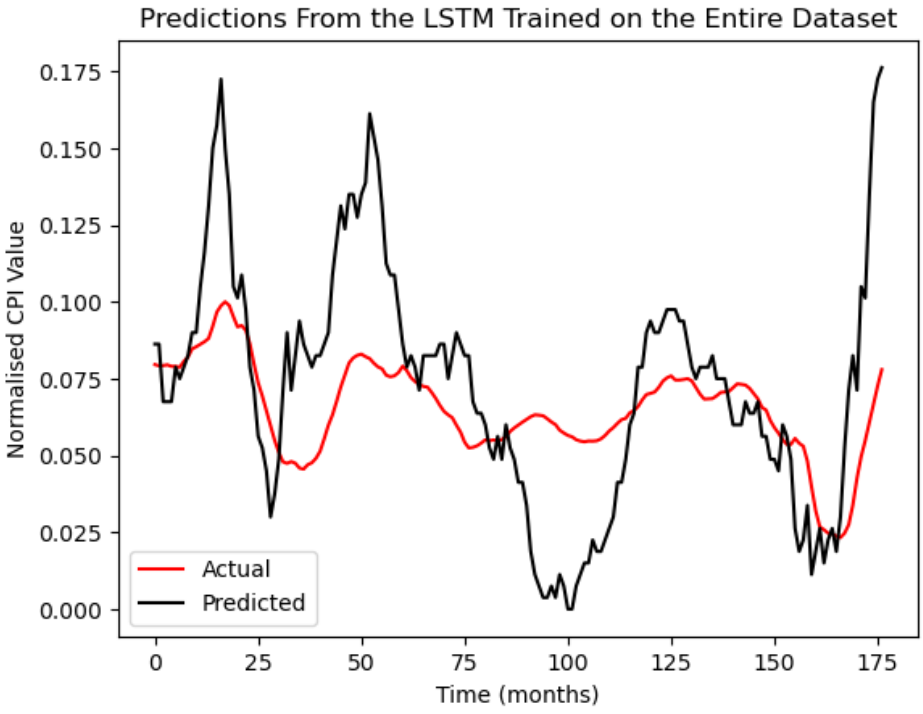


Figure 5.3: LSTM Network Trained on the Entire Dataset.

The predictions from the model trained on the entire dataset, produce a smooth curve that follows the trends of inflation but fails to depict the magnitude and precision of the finer changes.

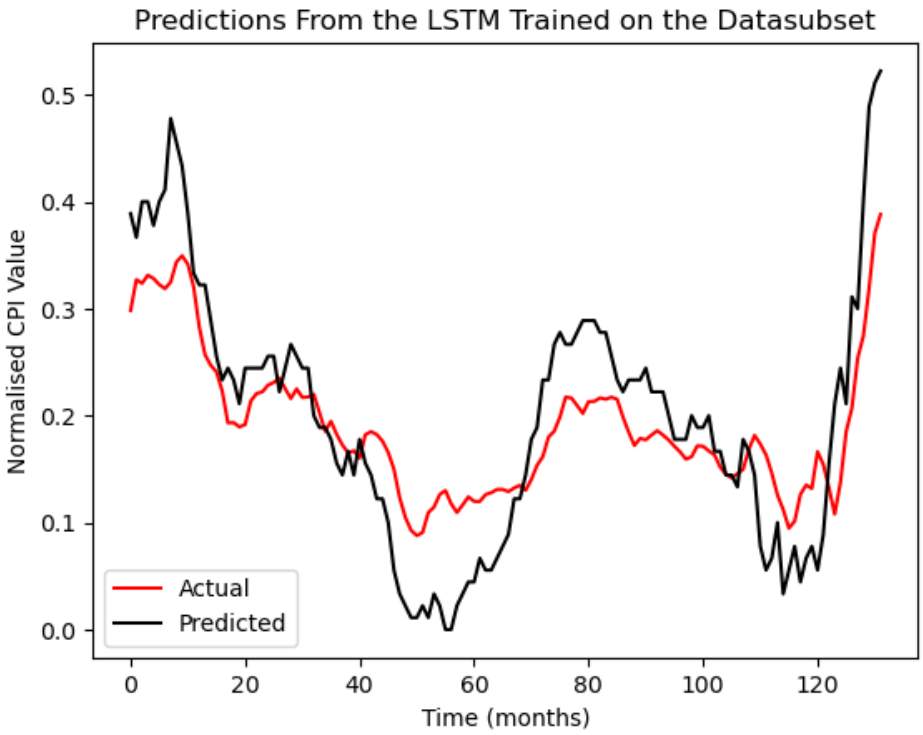


Figure 5.4: LSTM Network Trained on a subset of the Dataset.

The predictions of the LSTM trained on the subset of the data more accurately depict the small volatile increments but still fail to predict the magnitude of inflation.
The graph produced by the LSTM that was trained on the subset of the dataset is preferable to the graph produced by the LSTM trained on the entire dataset. However, neither model produces close accurate predictions of the actual value of inflation. Both LSTMs can follow the trends of inflation (although somewhat delayed) but struggle to produce predictions that are of the correct magnitude.

5.2.3 Feedforward Neural Network Predictions

Of the models implemented, the FNN produced the best results. This model, like the others, was implemented on both the whole dataset and a subset of the original data that did not include values from before the 1980s.

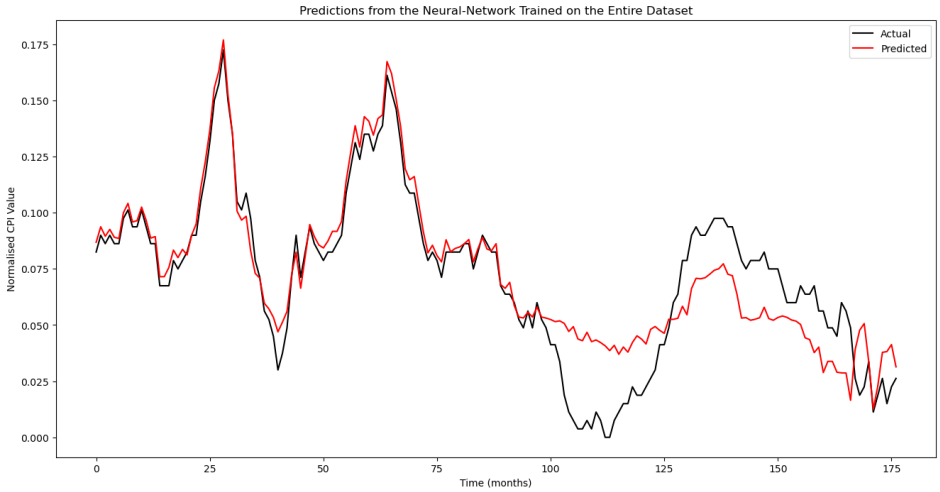


Figure 5.5: Neural-Network Trained on the Entire Dataset.

The predictions from the model trained on the entire dataset start well but lose accuracy towards the end of the training set.

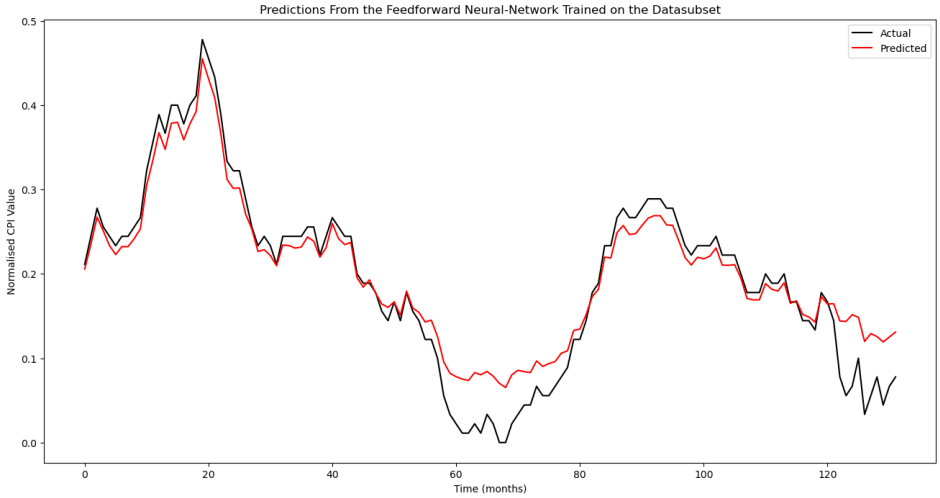


Figure 5.6: Neural-Network Trained on a subset of the Dataset.

Figures 5.5 and 5.6 show the results produced by the model with the entire data vs the subset of the data respectively. The figures show how the models trained on the subset of the dataset produced more accurate predictions than the models trained on the entire dataset.

5.2.4 Why the Feedforward Neural Network Had the Most Success

The artificial neural network shown in figure 5.6 produced the best results out of all of the models implemented to predict inflation. This is likely due to several factors.

Removing Data Outliers

This method was applied to all models successfully. The first 150 samples of CPI contained large outliers compared to the rest of the dataset. Including these samples would have negatively affected the model's predictive abilities. This was confirmed by the fact that the other models also showed an increase in performance after the first 150 rows of data were removed.

Hyperparameter Tuning

Another reason for the FNN's success is that it was the most flexible of the models with the largest number of hyperparameters that could be adjusted. The only other model that underwent similar hyperparameter tuning was the LSTM and interestingly the hyperparameters that were found to be optimal were similar for both models. However, some of the hyperparameters found to be optimal by the grid search were somewhat unexpected.

The model performed best with 128 nodes in the hidden layer, which was the highest option in the grid and higher than expected. The reason for the model preferring a higher number of nodes could be that with more nodes the model could better capture the problems' more intricate non-linear patterns. However, such a large number of nodes comes at a computational cost as it will require more processing power and take more time to run the model.

The preferred dropout rate was 0% meaning that it is quite unlikely the model came close to overtraining as if it did the model would have been more successful with a higher dropout rate. Another possibility is that the other dropout rates available in the grid were already too high and dropouts were occurring too often. The next smallest dropout rate available was 10% or 0.1 so this theory is unlikely as 0.1 is considered relatively low.

The ideal learning rate was 0.005 which was unexpected as s it is higher than Keras' default learning rate of 0.001. A higher learning rate often allows for faster convergence but can lead to the model oscillating or diverging if it overshoots the optimal point. Despite this, it does not seem like the model oscillated or diverged based on the loss graph as it did not display any additional inflections.

The final hyperparameter was a batch size of 8, this was the smallest batch size option. The batch size dictates the number of data samples the model works through before it updates its internal parameters. Thus a smaller batch size means that the model's parameters are being updated more frequently. Research[56] indicates that smaller batch sizes produce models with better generalisation. This can even be true for models trained using mini-batch sizes as small as 2. A model that updates its parameters more frequently is likely to perform well on data that has

a lot of complex relationships. Therefore, the fact that the smallest batch size produced the best results is not surprising as the data the model is training on likely contains many such relationships.

Recurrent Neural Networks vs None Recurrent Neural Networks

The previously mentioned factors still do not explain why the feedforward network produced superior results to the LSTM. During the 'Expected Model Performance' section of this report, the LSTM had been predicted to produce the best results. This prediction was made on the basis that an LSTM has the ability to retain historical data and use it in conjunction with present data in order to tune its parameters. Having this ability should in theory make the LSTM better suited for utilising the sequential time series data that made up the dataset for this project. However, as shown in the results of both the metrics and graphs presented, the LSTM performed worse than a non-recurrent network: the feedforward neural network.

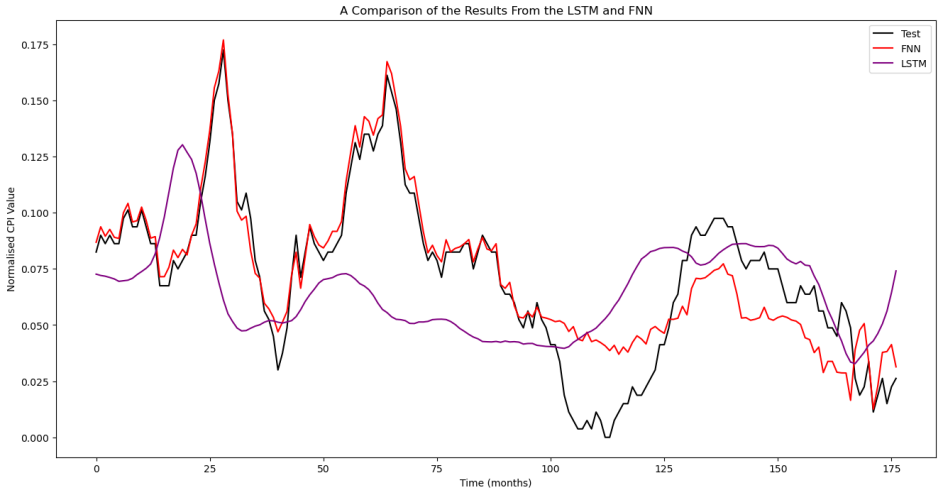


Figure 5.7: A Comparison of the LSTM and FNN Predictions.

Chapter 6

Conclusion

References

[1] R. S. Tsay, *Analysis of financial time series*. John wiley & sons, 2005.

[2] C. Harrington, "Fundamental vs. technical analysis," 2003.

[3] Y. Tang, Z. Song, Y. Zhu, H. Yuan, M. Hou, J. Ji, C. Tang, and J. Li, "A survey on machine learning models for financial time series forecasting," *Neurocomputing*, vol. 512, pp. 363–380, 2022.

[4] M. C. Thomsett, *Getting started in fundamental analysis*. John Wiley & Sons, 2006.

[5] A. S. Wafi, H. Hassan, and A. Mabrouk, "Fundamental analysis models in financial markets – review study," *Procedia Economics and Finance*, vol. 30, pp. 939–947, 2015. IISES 3rd and 4th Economics and Finance Conference.

[6] T. Seegmiller, "10 metrics to measure the financial efficiency of your organization," Sep 2023.

[7] S. B. Achelis, "Technical analysis from a to z," 2001.

[8] J. J. Murphy, *Technical analysis of the financial markets: A comprehensive guide to trading methods and applications*. Penguin, 1999.

[9] B. Rockefeller, *Technical analysis for dummies*. John Wiley & Sons, 2019.

[10] G. U. Yule, "Why do we sometimes get nonsense-correlations between time-series?–a study in sampling and the nature of time-series," *Journal of the Royal Statistical Society*, vol. 89, no. 1, pp. 1–63, 1926.

[11] O. Akbilgic, H. Bozdogan, and M. E. Balaban, "A novel hybrid rbf neural networks model as a forecaster," *Statistics and Computing*, vol. 24, pp. 365–375, 2014.

[12] R. S. Michalski, J. G. Carbonell, and T. M. Mitchell, *Machine learning: An artificial intelligence approach*. Springer Science & Business Media, 2013.

[13] M. Awad and R. Khanna, *Machine Learning*, pp. 1–18. Berkeley, CA: Apress, 2015.

[14] J. Dean, D. Patterson, and C. Young, "A new golden age in computer architecture: Empowering the machine-learning revolution," *IEEE Micro*, vol. 38, no. 2, pp. 21–29, 2018.

[15] J. Dean, "The deep learning revolution and its implications for computer architecture and chip design," *ArXiv*, vol. abs/1911.05289, 2019.

[16] C. Oner, "Inflation: Prices on the rise," Jul 2019.

[17] M. Parkin, *Inflation*, pp. 1–10. London: Palgrave Macmillan UK, 2016.

[18] N. G. L. Hammond, *Alexander the Great*. Chatto & Windus London, 1981.

[19] T. Economist, "Our big mac index shows how burger prices are changing."

[20] J. H. Clapham, *The bank of England*. CUP Archive, 1939.

[21] D. Beckett, "Consumer price inflation, uk: October 2023," Nov 2023.

[22] B. Francis-Devine, "Rising cost of living in the uk." `https://commonslibrary.parliament.uk/research-briefings/cbp-9428/`, 2023. [Accessed 17-12-2023].

[23] E. Nolsøe, "Bank of England's Covid money-printing spree 'drove up inflation' — telegraph.co.uk." `https://www.telegraph.co.uk/business/2023/04/18/bank-of-england-covid-money-printing-drove-up-inflation/#:~:text=During%20the%20pandemic%2C%20the%20Bank,to%20a%20record%20%C2%A3895bn.`, 2023. [Accessed 17-12-2023].

[24] F. Rosenblatt, "The perceptron - a perceiving and recognizing automaton," Tech. Rep. 85-460-1, Cornell Aeronautical Laboratory, Ithaca, New York, January 1957.

[25] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Nature*, vol. 323, pp. 533–536, Oct. 1986.

[26] H. Sak, A. Senior, and F. Beaufays, "Long short-term memory based recurrent neural network architectures for large vocabulary speech recognition," 2014.

[27] G. Petneházi, "Recurrent neural networks for time series forecasting," 2019.

[28] G. Ciaburro, V. Ayyadevara, and A. Perrier, *Hands-On Machine Learning on Google Cloud Platform: Implementing smart and efficient analytics using Cloud ML Engine*. Packt Publishing, 2018.

[29] B. E. Hansen, "Threshold effects in non-dynamic panels: Estimation, testing, and inference," *Journal of Econometrics*, vol. 93, no. 2, pp. 345–368, 1999.

[30] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting," *The journal of machine learning research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[31] C. Cortes and V. Vapnik, "Support-vector networks," *Machine learning*, vol. 20, pp. 273–297, 1995.

[32] Z. Wei, *A SVM approach in forecasting the moving direction of Chinese stock indices*. Lehigh University, 2012.

[33] L. Cao and F. E. Tay, "Financial forecasting using support vector machines," *Neural Computing and Applications*, vol. 10, no. 2, p. 192, 2001. Cited by: 272.

[34] E. Sadrfaridpour, K. Palmer, and I. Safro, "Aml-svm: Adaptive multilevel learning with support vector machines," 2020.

[35] T. K. Ho, "Random decision forests," in *Proceedings of 3rd international conference on document analysis and recognition*, vol. 1, pp. 278–282, IEEE, 1995.

[36] L. Breiman, "Random forests," *Machine learning*, vol. 45, pp. 5–32, 2001.

[37] L. Breiman, "Bagging predictors," *Machine learning*, vol. 24, pp. 123–140, 1996.

[38] C. Huyen, "Design a machine learning system," 2022.

[39] Google, "Using the python client library — ai platform training — google cloud."

[40] W. Bank, "World devlopment indicators," 2023.

[41] OECD, "Data warehouse," 2014.

[42] C. W. J. Granger, "Investigating causal relations by econometric models and cross-spectral methods," *Econometrica*, vol. 37, no. 3, pp. 424–438, 1969.

[43] D. H. . Wojciech W. Charemza and P. Burridge, "Is inflation stationary?," *Applied Economics*, vol. 37, no. 8, pp. 901–903, 2005.

[44] J. M. Wooldridge, *Introductory Econometrics: A Modern Approach*. ISE - International Student Edition, South-Western, 2009.

[45] C. V. Nicholson, "A beginners guide to lstms and recurrent neural networks," 2023.

[46] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[47] J. D. Hunter, "Matplotlib: A 2d graphics environment," *Computing in Science & Engineering*, vol. 9, no. 3, pp. 90–95, 2007.

[48] C. R. Harris, K. J. Millman, S. J. van der Walt, R. Gommers, P. Virtanen, D. Cournapeau, E. Wieser, J. Taylor, S. Berg, N. J. Smith, R. Kern, M. Picus, S. Hoyer, M. H. van Kerkwijk, M. Brett, A. Haldane, J. F. del Río, M. Wiebe, P. Peterson, P. Gérard-Marchant, K. Sheppard, T. Reddy, W. Weckesser, H. Abbasi, C. Gohlke, and T. E. Oliphant, "Array programming with NumPy," *Nature*, vol. 585, pp. 357–362, sep 2020.

[49] T. pandas development team, "pandas-dev/pandas: Pandas," feb 2020.

[50] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015. Software available from tensorflow.org.

[51] F. Chollet *et al.*, "Keras." `https://keras.io`, 2015.

[52] M. L. Waskom, "seaborn: statistical data visualization," *Journal of Open Source Software*, vol. 6, no. 60, p. 3021, 2021.

[53] M. "Löning, F. Király, T. Bagnall, M. Middlehurst, S. Ganesh, G. Oastler, J. Lines, M. Walter, ViktorKaz, L. Mentel, chrisholder, L. Tsaprounis, RNKuhns, M. Parker, T. Owoseni, P. Rockenschaub, danbartl, jesellier,

eenticott-shell, C. Gilbert, G. Bulatova, Lovkush, P. Schäfer, S. Khrapov, K. Buchhorn, K. Take, S. Subramanian, S. M. Meyer, AidenRushbrooke, and B. Rice, ""sktime/sktime: v0.13.4"," 2022.

[54] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," *CoRR*, vol. abs/1502.03167, 2015.

[55] J. Brownlee, "How to Choose an Activation Function for Deep Learning - MachineLearningMastery.com — machinelearningmastery.com." `https://machinelearningmastery.com/choose-an-activation-function-for-deep-learning/`. [Accessed 22-04-2024].

[56] D. Masters and C. Luschi, "Revisiting small batch training for deep neural networks," 2018.