UNIVERSITY OF LEEDS

School of Mathematics

Optimised Budget Allocation in an Uncertain Economy: A Deep Reinforcement Learning Approach

Written by:
Nathan Scarrott

Supervised by:
Prof. Onno Bokhove

2023/24 Project in Mathematics

Contents

1 Introduction

1.1 Summary

The UK government continually face uncertainty due to several reasons such as economic volatility, tax income fluctuations and political instability. From global pandemics to geopolitical tensions, the need for adaptable and resilient economic policy-making has never been more pronounced. This project will use machine learning techniques to optimise government budget allocation in a simulated environment. The specific technique that will be used is deep reinforcement learning.

Machine learning, in the context of optimising government budget allocation under revenue uncertainty, would involve creating a model that initially randomly makes decisions and then, over time, learns the optimal way to distribute the budget. The aim is to optimise these economic indicator variables; Gross Domestic Product (GDP), life expectancy and youth unemployment rate in correlation with funding in infrastructure, healthcare and education sectors respectively. We will use previous studies to determine the relationship between the funding in these sectors and the economic indicator's value. Expanding on this, as new data becomes available or economic events unfold, the machine learning model will use this data to adjust the allocation of government budget responsively. This essentially creates a real-time re-calibration of budgetary decisions. This ensures that the government can respond to economic fluctuations efficiently, maximising the effectiveness of government spending to achieve policy objectives and support economic stability. Over time the machine learning model will create an effective strategy to optimally distribute this funding.

The combination of ARIMA processes and Monte Carlo simulations will provide an accurate and near-exhaustive method for addressing the largely variant array of economic outcomes. This will both enhance the precision of our budget allocation by providing a detailed forecast of key economic variables and allow for dynamic adaption of budget allocation. Considering the variables' uncertainty will require modelling these variables followed by a forecast. Real UK historical data will be used to model these variables. We will use forecasting techniques that leverage Monte Carlo methods to generate a wide array of scenarios each showcasing different future possibilities. These forecasts will be based on the ARIMA modelling of these variables. By simulating several outcomes of these economic indicators, the robustness of different funding strategies can be assessed across multiple economic scenarios.

The project will demonstrate the efficiency and effectiveness of machine learning in devising an adaptable strategy for budget allocation for a government body. This project does not take into account all potential causes of uncertainty. However, this methodology has the potential to be extended to take into account a broader array of economic factors in which a calculated responsive action can be executed.

1.2 Ethical Consideration

While this project uses simulated future data based on historical UK data, it serves as a baseline example of how this could be implemented by government bodies globally. This methodology could be adapted to fit a plethora of societal and business-related issues where uncertainty occurs. Due to the multitude of applications of this methodology, we must consider the societal impact of our work. We do this by acknowledging the potential risk of misuse. The current problem does not model all significant variables which may need to be taken into account when used in practice. This means that our method may suggest allocation strategies that would, in reality, negatively affect unconsidered sectors or indicators. For this reason, caution must be exercised that the data behind this model is correct as well as the relationship between variables.

2 Literature Review

Machine learning has become a key technique in the modern world to address decision-making problems that involve randomness. Some of the earliest work on this topic includes Arthur Samuel's work [1] where he introduced the concept of machine learning in which he delved into pattern recognition in games such as checkers. Following this, in 1958 Frank Rosenblatt [2] introduced the concept of 'Perceptron'. This laid the foundations for the future of machine learning. This was further extended by Vapnik and Chervonenkis in the 1960s, illustrating its applicability to complex problems. Cumulatively, these works allowed further progression in the field of machine learning.

Recent advances in computational techniques have allowed for major expansion in this field. The potential capability of machine learning using algorithms has grown exponentially with technology innovation. Geoffrey Hinton, Yoshua Bengio and Yann LeCun's work [3] introduced deep learning architectures which leveraged modern technology, enhancing the scalability of machine learning models. Machine learning has also been adapted to include techniques facilitating optimisation models with vast amounts of data. This can be demonstrated by the work of James Manyika et al. at the McKinsey Global Institute [4], which highlighted the effects of big data analytics in machine learning applications across various industries.

The application of machine learning spans numerous fields and as this topic expands one would expect to see this further integration into society. This project will aim to leverage machine learning techniques with Monte Carlo simulations and ARIMA processes to create an effective model to distribute government budgets under uncertainty.

3 Methodology

3.1 General Outline

This project will use ARIMA processes, Monte Carlo simulations and deep reinforcement learning to optimally distribute the UK government budget while facing uncertainty. Using ARIMA processes, our time series for each economic indicator including GDP, life expectancy and youth unemployment rate can be modelled. This modelling will provide a basis for our Monte Carlo simulations variance. Monte Carlo simulations will allow us to generate a wide array of economic scenarios. This provides an understanding of potential budget allocations under various economic conditions. Machine learning will be used within the simulation.

As the Monte Carlo simulations run various, economic scenarios will unfold. Then as this new data becomes available, there will be real-time re-calibration of budgetary decisions to ensure an optimal distribution of funding into the sectors to maximise the correlated variables. We aim to optimise GDP, life expectancy, and youth unemployment rate with respect to funding towards education, healthcare and infrastructure respectively.

A study published in MDPI [5] finds the significant long-term effects of educational spending. It suggests that a 1% increase in government education funding could decrease youth unemployment rates by 10.81%. Also, research from the National Library of Medicine [6] suggests that a 1% increase in healthcare funding is correlated with an increase in life expectancy of 0.35 months. Finally, an MDPI analysis of the dynamics of infrastructure spending and economic growth [7] finds that a 1% increase in infrastructure investments has the potential to increase GDP by 0.5% over a decade. We will assume this to be linear over time therefore implying a monthly rate of 0.00417%. Within our model, we will assume these effects to be instantaneous. Although this would not be the case in reality, this will allow us to derive a solution without causing over-complexity in our model setup. These studies will collectively provide a basis for this project which will seek to maximise the given variables and allow us to quantify the relationship between government funding and these variables.

3.2 Problem Setup

Here we define some parameters and constraints which will be used throughout this project.

- f_e, f_h, f_i: Budget allocations towards education, healthcare, and infrastructure, respectively.
- G, L, Y: Indicators for GDP, life expectancy, and youth unemployment rate, respectively.

The objective is to maximise the economic indicators G, L, and Y with respect to funding in the correlated sectors. First we define our constraints:

1. Budget constraint:
$$f_e + f_h + f_i \leq B,$$
where B is the total available government budget.
2. Non-negativity constraints:
$$f_e \geq 0, \quad f_h \geq 0, \quad f_i \geq 0$$

We will now implement the quantified correlations found between funding in our sectors and the growth of the related economic indicator.

- A 1% increase in infrastructure spending (f_i) results in a 4.17×10^{-3}% growth in GDP (G).
$$G = 4.17 \times 10^{-5} \frac{f_i}{B} \cdot 100.$$
- For every 1% increase in education funding (f_e), the youth unemployment rate (Y) is expected to decrease by 10.81%.
$$Y = -0.1081 \frac{f_e}{B} \cdot 100.$$
- A 1% increase in healthcare funding (f_h) increases life expectancy (L) by 0.35 months.
$$L = 0.35 \frac{f_h}{B} \cdot 100.$$

Simplifying these terms and taking into account the uncertainty in the economic impacts ($\epsilon_G, \epsilon_L, \epsilon_Y$), the model incorporates stochastic elements to account for variability in outcomes:

$$G(f_i) = 4.17 \times 10^{-3} \frac{f_i}{B} + \epsilon_G, \tag{1}$$

$$L(f_h) = 35 \frac{f_h}{B} + \epsilon_L, \tag{2}$$

$$Y(f_e) = -10.81 \frac{f_e}{B} + \epsilon_Y. \tag{3}$$

By assumption, we will model ϵ to follow a normal distribution with a mean of 0 and variance represented by the historical data, that is $\epsilon_k \sim \mathcal{N}(o, \sigma_k^2)$, where σ_k is the standard deviation of k, our indicator variable. These equations allow us to quantify a relationship between the variables and funding in the correlated sector and will be used by our machine learning model later on in this project.

4 Data Collection and Analysis

4.1 Time Series Analysis

4.1.1 Introduction

The aim here will be to model these variables to an ARIMA(p, d, q) model and forecast based on this modelling. An ARIMA process, which stands for Autoregressive Integrated Moving Average process is the combination of an AR(p) and MA(q) process with differencing applied. The AR(p) component of this model represents the variable as a linear combination of its past values. This directly assumes that past values have an influence on its current and future values. The MA(q) component represents the linear combination of past error terms. This allows us to capture high volatility effects from previous values. An ARMA(p, q) process, X_t, takes the form:

$$X_t = \sum_{i=1}^{p} \alpha_i X_{t-i} + \varepsilon_t + \sum_{j=1}^{q} \beta_j \varepsilon_{t-j}, \tag{4}$$

where $\varepsilon_t \sim \mathcal{N}(0, \sigma^2)$ (i.e. ε_t is white noise with some variance to be determined) and α_i and β_i are constants which quantify the influence of previous data points. Furthermore, $\{X_t\}$ is an ARIMA$(p, d + o, q)$ process if $\nabla^o X_t$ is a stationary ARIMA(p, d, q) process ("d" $\sim$ "integrated") where $\nabla X_t = X_t - X_{t-1} \quad \forall t$. This will allow us to accurately create Monte Carlo simulations to provide realistic data for our machine learning model to be trained on.

For an AR(p) process we must ensure stationary holds in the data. This means the mean and variance remain constant over time. For this, we will use an Augmented Dickey-Fuller test. This assumes that the data is non-stationary as a null hypothesis and the aim will be to reject this hypothesis at a 5% critical value. For this, we may need to use transformation techniques on the data such as differencing, as defined above, or a non-linear transformation. Also, an AR(p) process cannot behave seasonally. Seasonality occurs when a time series is affected by a time period such as a month, or quarter of a year. For example, if we were to look at the sales of ice cream in a year there would be a clear spike around Q2 and Q3 (April - September) each year. Any seasonality must be removed from the data to fit it into an AR(p) model. We know an MA(q) process is naturally stationary due to $\{\varepsilon_t\}$ being stationary since it follows a normal distribution, which indicates a constant mean and variance. Therefore, it is only required that an MA(q) process must hold the condition of being invertible. This means that $|\beta_i| \geq 1$ $\forall i = 1, 2, \ldots, q$. Since the tools within Python ensure this when modelling the data to an MA(q) process, this does not require further analysis.

To efficiently and effectively find stationary transformations of our non-stationary data, we can use an algorithm to test many different transformations using an Augmented Dickey-Fuller test and return 'True' for transformations which are stationary. We take transformations including; logarithmic, square root, square, first-order differenced, logarithmic difference and square root differenced. Algorithm 1 describes this process:

Algorithm 1 Test Various Transformed Time Series for Stationarity

```
function DickeyFuller(series)
    return result of Dickey-Fuller test on series
end function

function TestTransformations(data, column)
    transforms ← list of transformations (e.g., log, diff)
    for each transform in transforms do
        transformed ← apply transform to data[column]
        results[transform] ← DickeyFuller(transformed)
    end for
    return results
end function
```

Autocorrelation and partial autocorrelation are key tools we use to model our data. Autocorrelation represents the correlation between our variable's data at a point in time and its previous values. This gives us insight into repetitive patterns within the data and tells us about the MA(q) component. The autocorrelation at lag k, ρ_k can represented mathematically as:

$$\rho_k = \frac{\sum_{t=k+1}^{T}(X_t - \bar{X})(X_{t-k} - \bar{X})}{\sum_{t=1}^{T}(X_t - \bar{X})^2}, \tag{5}$$

where X_t is the value of the time series at time t, $\bar{X}$ is the mean value of the time series and T is the number of observations. The partial autocorrelation measures the correlation between observations at two points without including points in between. This tells us how previous data points affect the current data point. This is useful for determining the AR(p) component of our model. The partial autocorrelation at lag k, ϕ_{kk}, can be represented mathematically as:

$$\phi_{kk} = \frac{\rho_k - \sum_{j=1}^{k-1} \phi_{k-1,j}\rho_{k-j}}{1 - \sum_{j=1}^{k-1} \phi_{k-1,j}\rho_j}, \tag{6}$$

where $\phi_{k-1,j}$ is the partial autocorrelation at the previous lag. Following, we can plot the Autocorrelation Function (ACF) and Partial Autocorrelation Function (PACF) between lags which are essential tools when modelling time series where lags are backward shifts in time with periods such as quarters or years. In practise, you usually use just under half the number of data points you have for the number of lags you consider. This provides insight into the order of our ARIMA process, p, d and q, while also indicating the presence of seasonality which would need to be removed if found. We will use these general rules to determine the best fit from the ACF and PACF as discussed in a 2019 Medium article [8]:

1. Look for tail-off pattern in either ACF or PACF.
2. If tail off at ACF → AR model → Cut off at PACF will provide order p for AR(p).
3. If tail off at PACF → MA model → Cut off at ACF will provide order q for MA(q).
4. Tail of at both ACF and PACF → ARMA model.

Where there is a discrepancy between models we will use Akaike Information Criterion (AIC). This quantifies the accuracy of the fit and the lowest value, in general, indicates a better fit. AIC represents the information lost by the model and it is a relative measure, meaning it is used comparatively. Python has tools which allow us to calculate this easily.

4.1.2 Gross Domestic Product (GDP)

We will model the GDP to an ARIMA process. After collecting this data from the Office of National Statistics [9], we must now plot and analyse this time series:

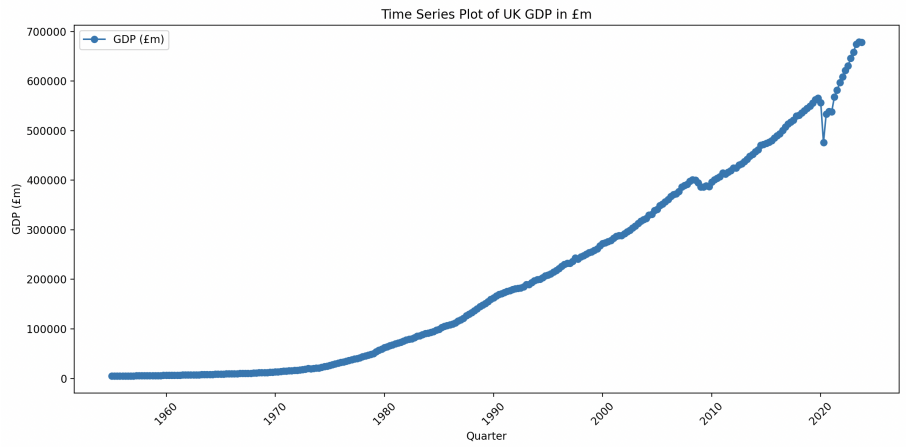

Figure 1: GDP data in the UK between 1955 Q1 and 2023 Q4.

Here we see a clear upward trend indicating an increase in the UK's GDP over time with no major fluctuations until recent times. These recent fluctuations are likely due to the effects of COVID-19 on the economy. We see no significant peaks or troughs. To further analyse this data we must remove the trend using regression techniques. This is done by fitting a trend line to the data and subtracting the actual data value from the fitted trend value for each point respectively. This data has already been seasonally adjusted meaning we don't have to be concerned with the data exhibiting seasonal behaviour.

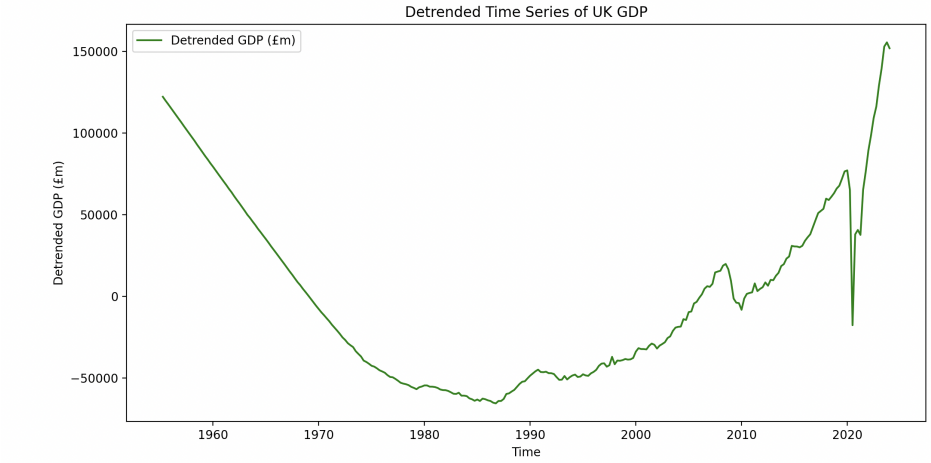

Figure 2: Detrended GDP data in the UK between 1955 Q1 and 2023 Q4.

This detrended data shows the fluctuations around the trend line. We can interpret from this that the data may not be stationary. However, to further assess this, an Augmented Dickey-Fuller test can be performed. A null hypothesis H_0 := the detrended GDP data is non-stationary and an alternative hypothesis H_1 := the detrended GDP data is stationary is taken. For this, a 5% significance level (or $\alpha = 0.05$) will be used. A test statistic of -0.388 is calculated, which is much higher than our 5% critical value of -2.872. Also, a p-value of 0.912 is calculated which is much greater than our α value. These factors, together, indicate that we cannot reject our null hypothesis and can conclude our data is not stationary. We now apply differencing to the data to attempt to stabilise the

mean and variance. The plot of the detrended singularly differenced data can be found below:

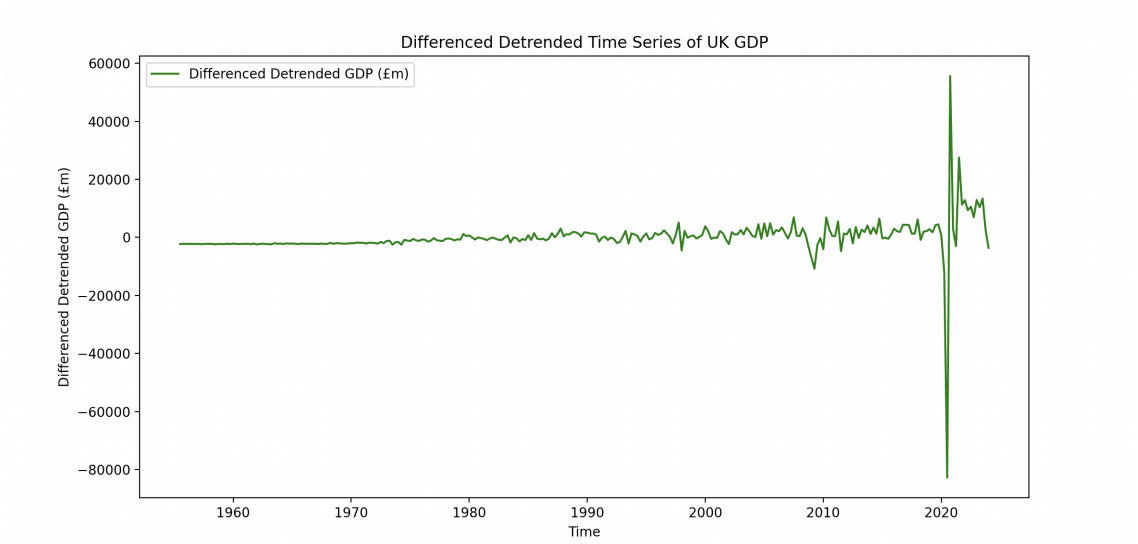

Figure 3: Differenced and detrended GDP data in the UK between 1955 Q1 and 2023 Q4.

This differenced data appears to take more of a stationary form. However, we have more pronounced outliers in the years 2021 and 2022 due to the global impact of COVID-19. Although there is a potential argument to remove these data points, there is also reasonable justification to keep these data points in the fact that economic volatility can potentially occur again in the future, especially in the current harsh economic conditions. This could be because of geopolitical issues, virus outbreaks, and many more occurrences. Following, it is essential to re-test this differenced data for stationarity using the Augmented Dickey-Fuller test, once again using a 5% critical value. A null hypothesis $H_0 :=$ the differenced detrended GDP data is non-stationary and an alternative hypothesis $H_1 :=$ the differenced detrended GDP data is stationary is taken. After analysis, a test statistic of -7.709, which is much lower than the 5% critical value of -2.872 and a p-value of 1.28×10^{-11}. This indicates that the null hypothesis can be rejected and the data is stationary; as required for an AR(p) process. We can now assign our ARIMA(p, d, q) process a d-value of 1, indicated by our use of first-order differencing.

To determine the order of the AR process, p, and the MA process, q, correlograms of the ACF and PACF can be used. The ACF and PACF plot can be found below:

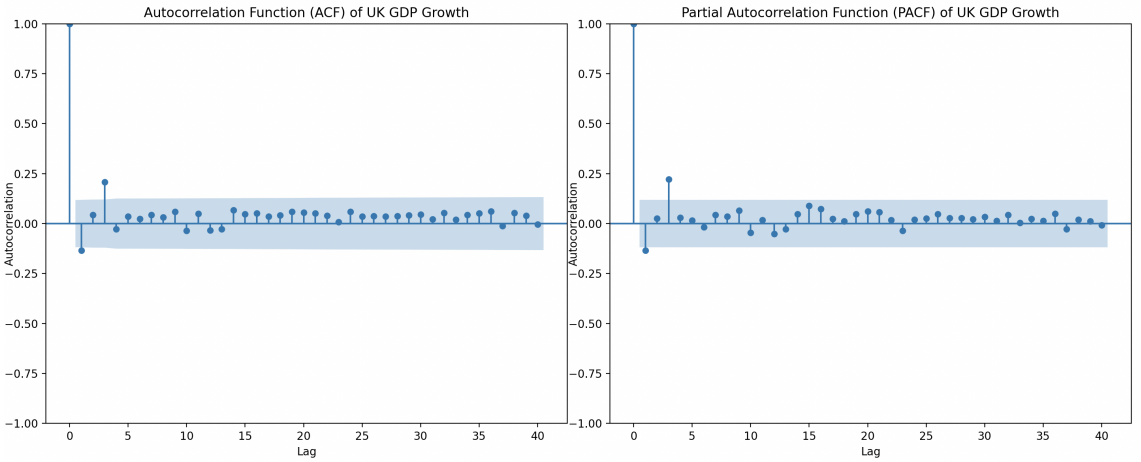

Figure 4: ACF and PACF plot for differenced detrended GDP values.

Both plots show the natural spike at lag 0 for a value of 1. This is as expected as this represents the autocorrelation between lag 0 and itself which is $\frac{\rho_0}{\rho_0} = 1$. These plots both show no significant autocorrelation between lags as we can tell by all further points falling within, or close to, the 5% significance level. This is represented by the blue area. This indicates no AR or MA behaviour is being exhibited by our data. Due to this, we suspect an ARIMA$(0, 1, 0)$ model may be appropriate to represent our data.

4.1.3 Life expectancy

Now, we must take similar action to model life expectancy data to an ARIMA(p, d, q) model. First, collecting and plotting this data from the Office of National Statistics [10], will allow us to analyse the initial time series:

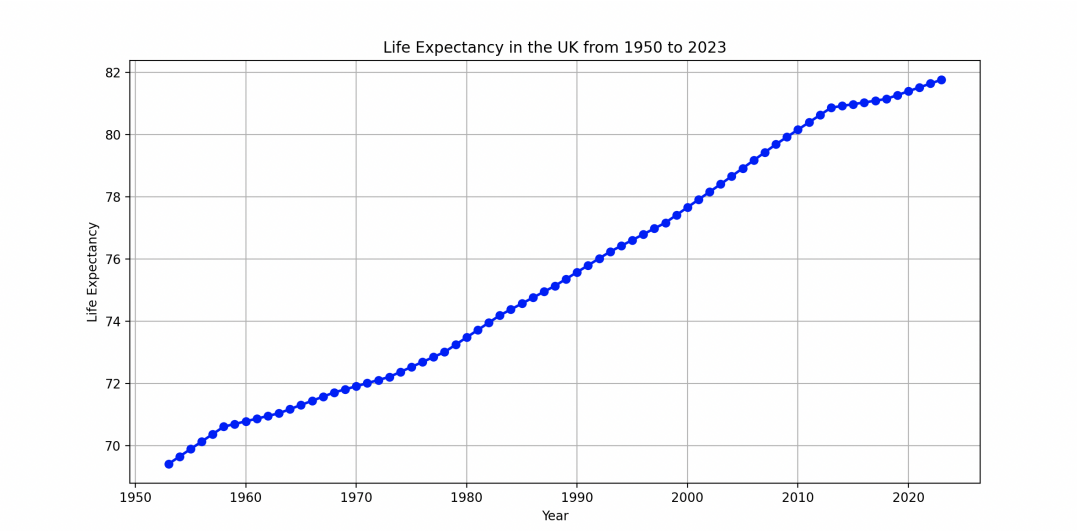

Figure 5: Life expectancy data in the UK between 1950 and 2023.

Analysing this graph shows us an increasing life expectancy between 1950 and 2023. By inspection, this graph also shows a decrease in the rate at which life expectancy increased shortly after 2011. This could be due to factors such as global funding into healthcare research decreasing or other factors which cause higher death rates. Detrending this data will allow us to analyse the residuals:

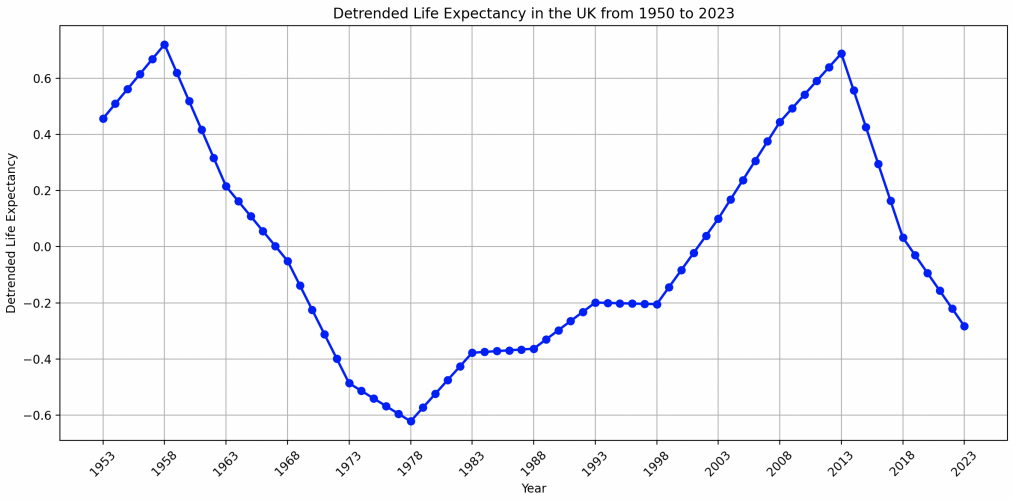

Figure 6: Detrended life expectancy data in the UK between 1950 and 2023.

Here we see the detrended life expectancy representing the residuals of our linear fit.

Visually, it is difficult to come to any kind of conclusion about the stationarity of this plot. Therefore, an Augmented Dickey-Fuller test can be used to determine this using a 5% critical value. A null hypothesis H_0 := the detrended life expectancy data is non-stationary and an alternative hypothesis H_1 := the detrended life expectancy data is stationary is taken. After analysis, a test statistic of -2.847, which is slightly higher than the 5% critical value of -2.904440 and a p-value of 0.0518. This indicates that the null hypothesis cannot be rejected and the data is non-stationary. However, since this is so close to the 5% level we can perform further analysis to check whether we can consider our data to be stationary. Analysing the ACF and PACF will allow us to do this:

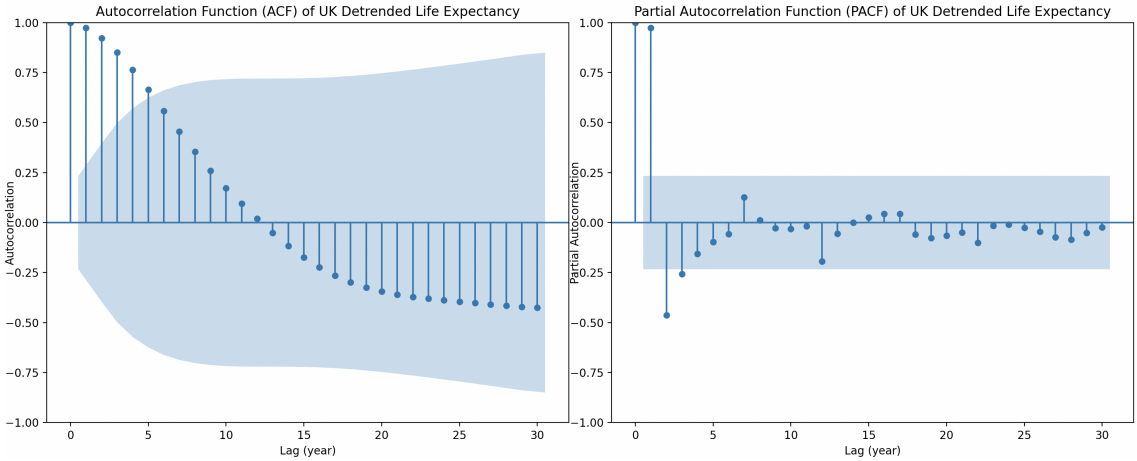

Figure 7: ACF and PACF plot of the detrended UK life expectancy data between 1950 and 2023.

It is worth mentioning that our 5% significance level within the ACF plot is larger than within other plots due to having less data. This is due to the fact that this data is yearly and our other data is quarterly. This data shows no clear signs of stationarity so we can continue in modelling the detrended data. From the ACF plot, we can see a slow decline over the lags and the PACF exhibits a cut-off at lag 2 or 3. This indicates that there may be an AR component of our model with order 2 or 3. Therefore, an ARIMA(2,0,0) or ARIMA(3,0,0) model may be a good fit for our data and we can use AIC analysis to find the better of the two. AIC values of -268.2 and -266.2 are found for the ARIMA(2,0,0) and AR(3,0,0) models respectively. This indicates that the ARIMA(2,0,0) model may be a better fit but only marginally. Furthermore, due to the fact that we don't want to over-fit or cause this model to be too complex, the lower-order term is better. Therefore, we will assume an ARIMA(2,0,0) model for our life expectancy data.

4.1.4 Youth Unemployment Rate

Finally, we must also model youth unemployment rate data to an ARIMA(p,d,q) model. Collecting this data from Macrotrends [11] and plotting will allow us to analyse the initial time series:

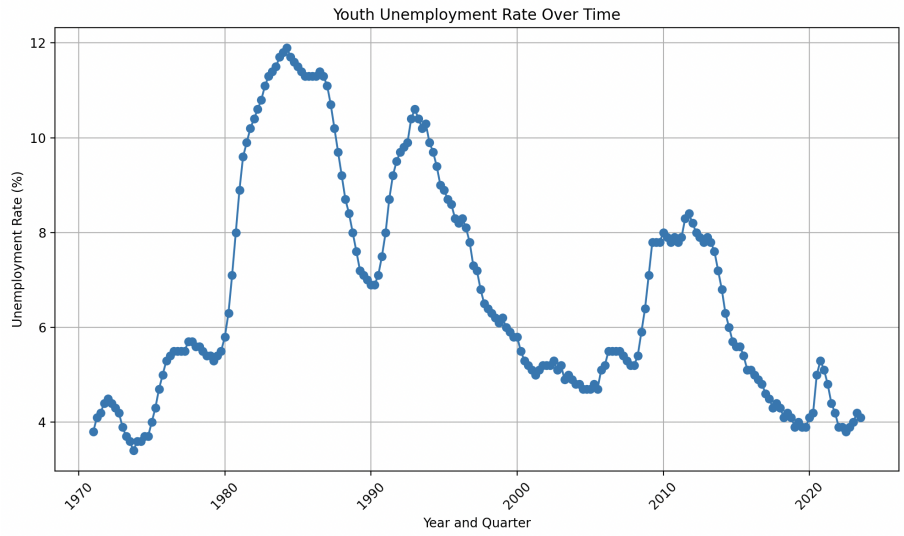

Figure 8: Youth unemployment rate data in the UK between 1971 and 2023.

This graph overall shows several peaks and troughs of the unemployment rate throughout the years with high volatility. This indicates the potential of the data being non-stationary due to the likely potential of a non-constant variance. This plot shows no clear upward or downward trend. This indicates there may be no reason to remove the trend from this plot as this will have no effect if the trend is approximately linear. Now it is appropriate to perform an Augmented Dickey-Fuller test on our data to test for stationarity. A null hypothesis H_0 := the youth unemployment rate data is non-stationary and an alternative hypothesis H_1 := the youth unemployment rate data is stationary is taken with a significance level of 5%. A test statistic of -2.325793 is found and compared to a 5% critical value of -2.875538, we find that we cannot reject the null hypothesis indicating the data may not be stationary. A p-value of 0.163784 is found which is higher than our significance level of 0.05 meaning we may have to perform a transformation on this data. After implementing Algorithm 1 11, it can be found that differencing this data achieves stationarity at a 5% significance level. The plot of the differenced data is as shown:

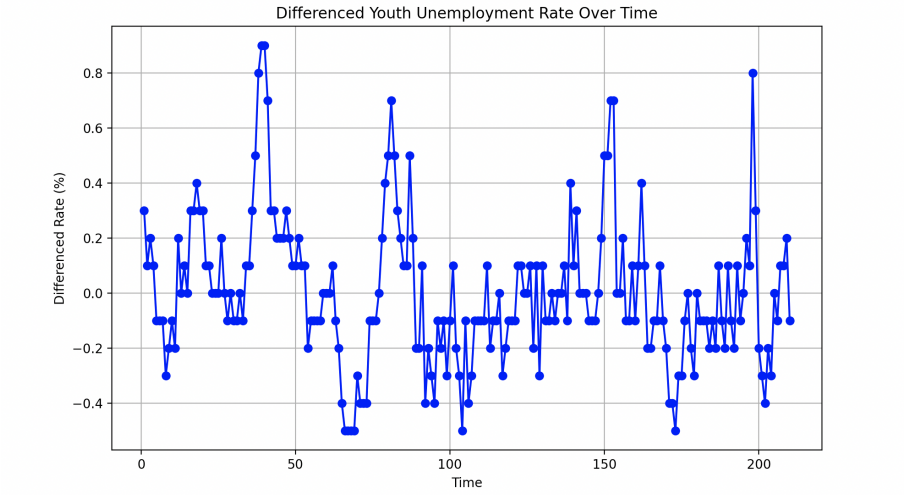

Figure 9: Differenced youth unemployment rate data in the UK between 1971 and 2023.

We can now continue to analyse this stationary data and model them to an ARIMA(p, d, q) process. An ACF and PACF plot is now appropriate:

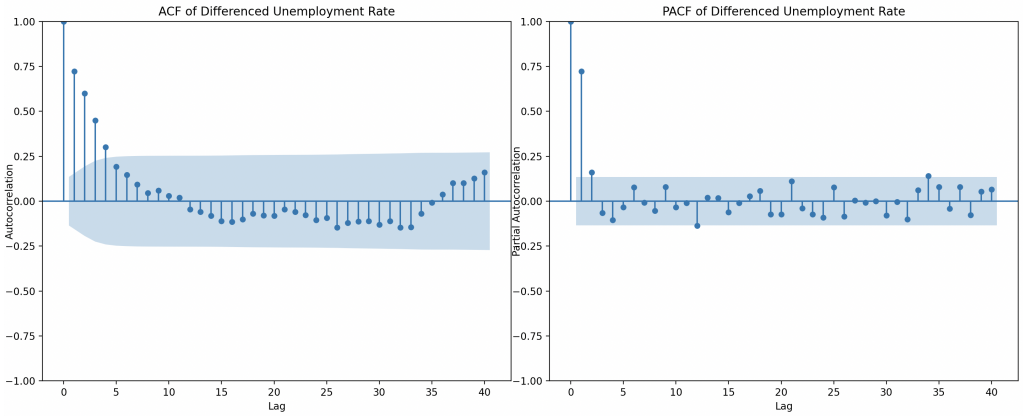

Figure 10: ACF and PACF plot of the differenced detrended youth unemployment rate data.

The ACF plot shows a gradual cut-off and we fall within the confidence interval at lag 5. This is an indication to consider an AR process. Considering the PACF, we find it cuts off after lag 1 or 2, indicating an AR(1) or AR(2) model would be fitting. This leaves the potential model fits as ARIMA(2,1,0) and ARIMA(1,1,0). Using AIC analysis we can find the better fit of the two. After calculating AIC values of -260.9 and -112.2 for our ARIMA(2,1,0) and ARIMA(1,1,0) models respectively, we can infer that an ARIMA(2,1,0) model may fit better. We will assume an ARIMA(2,1,0) model for our data.

4.1.5 Model Credibility

To test the credibility of our ARIMA models we can use a technique called walk-forward validation. This allows us to assess the performance of our model. We begin by splitting the time-series observed dataset into training data and testing data. We can then train our chosen model on the training data and see how correctly it predicts our testing data. We will split the data at approximately a 4:1 ratio. This means we train our model on the first $\frac{4}{5}$ of the observed data using our chosen model in the above chapter and see how accurately it forecasts the final $\frac{1}{5}$ of the observed data. We will use the Root Mean Squared Error (RMSE) to measure the error in our forecasting. RMSE is given by:

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(\hat{y}_i - y_i)^2}, \tag{7}$$

where n is the number of observations, $\hat{y}_i$ is the predicted value and y_i is the observed value. A RMSE of 10% of the average value within the dataset will indicate a good fit. This algorithm explains this process:

Algorithm 2 Walk-Forward Validation for ARIMA Model

1: Split the dataset into training and testing sets
2: Initialise the training dataset D_{train} and testing dataset D_{test}
3: Define the ARIMA model order (p, d, q)
4: Initialise an empty list for predictions P and for actual values A
5: **for** each point t in D_{test} **do**
6: Fit the ARIMA model on D_{train}
7: Forecast the next point $\hat{y}_{t+1}$ using the fitted model
8: Append forecast $\hat{y}_{t+1}$ to P
9: Append actual observation y_{t+1} to A
10: Add y_{t+1} to D_{train} for the next iteration
11: **end for**
12: Calculate RMSE using predictions P and actual values A

Firstly, considering GDP, we find a RMSE value of 16203.6 with an average of 204326.6 implying the RMSE is under 10% of our average value indicating a sufficient fit. Following, now we consider life expectancy. Applying the above algorithm, 2, we find an RMSE of 0.051 with an average value of 75.6. This, once again, indicates a sufficient model fit. Finally, considering youth unemployment rate, we find an RMSE of 0.204 with an average of 6.7 indicating we have found a good model for our data. Now we have appropriately modelled and validated our data we can perform the Monte Carlo simulations.

4.2 Scenario Generation using Monte Carlo Simulations

4.2.1 Overview

A Monte Carlo simulation is a computational algorithm which creates various paths based on randomly sampling a chosen aspect such as past volatility, residuals or any other plausible result determined from the data. These random samples are used recursively to create a path. This relationship can be described mathematically as $Q_{t+1} = Q_t + \epsilon$ where Q_t is the variable value at time t and ϵ is a randomly selected residual. Furthermore, the residuals can be represented as $\epsilon = q_t - \hat{q}_t$, where q_t is the observed value and $\hat{q}_t$ is the modelled value at time t. This method is used in multiple different circumstances and can be used to assess risk and uncertainty or create a wide array of scenarios as well as many other use cases. These paths can be visualised by plotting them in an easily understandable time series form.

Monte Carlo simulations will allow us to simulate a wide array of economic outcomes. This simulation of GDP, youth unemployment rate and life expectancy will be based on the residuals between our fitted ARIMA(p, d, q) model and the observed data. These are random fluctuations which are not explained by our model. A residual will be selected at random and will be sequentially added onto the starting point, hence creating a path. This is a good method as it provides a large range of possible outcomes. Furthermore, this ensures that we are not assuming normality or linearity in the residuals allowing us to capture any complex pattern within the data. We must note that this method assumes past volatility will indicate future volatility. Algorithm 2, 3, highlights how this is done.

Algorithm 3 Monte Carlo Simulation Generation

```
procedure GenerateMCSimulations(data, order, last_value, years, num_simulations)
    model ← fit ARIMA model to data with order
    residuals ← extract residuals from model
    paths ← initialise array of size years × num_simulations
    for i ← 1 to num_simulations do
        path ← array of size years with first value last_value
        for j ← 2 to years do
            residual ← sample from residuals
            path[j] ← path[j − 1] + residual
        end for
        paths[:, i] ← path
    end for
    plot the simulated paths starting from last_known_date
end procedure
data ← load life expectancy data
order ← specify the order of the ARIMA model
last_value ← the last known value of the time series
years ← the number of years to simulate
num_simulations ← the number of simulation paths
GenerateMCSimulations(data, order, last_value, years, num_simulations)
```

These Monte Carlo simulations will allow us to train the machine-learning model to distribute government budget with a wide array of economic outcomes while aiming to optimise our variables with respect to funding in the correlated sector. Section 5.1 highlights how this is done.

4.2.2 Application to Economic Indicators

Gross Domestic Product (GDP) GDP, although showing consistent signs of growth from the 1960s, can vary. Recent events such as the COVID-19 pandemic caused high levels of volatility within this indicator. There are a plethora of potential events that could cause similar volatility within our economy so we must consider a wide array. Monte Carlo simulations allow us to do this. Here this simulation shows 100 different paths generated using the methods described above. It is important to note that when training our machine learning model a much larger number of paths will be generated allowing more comprehensive training, we show only 100 paths for clarity.

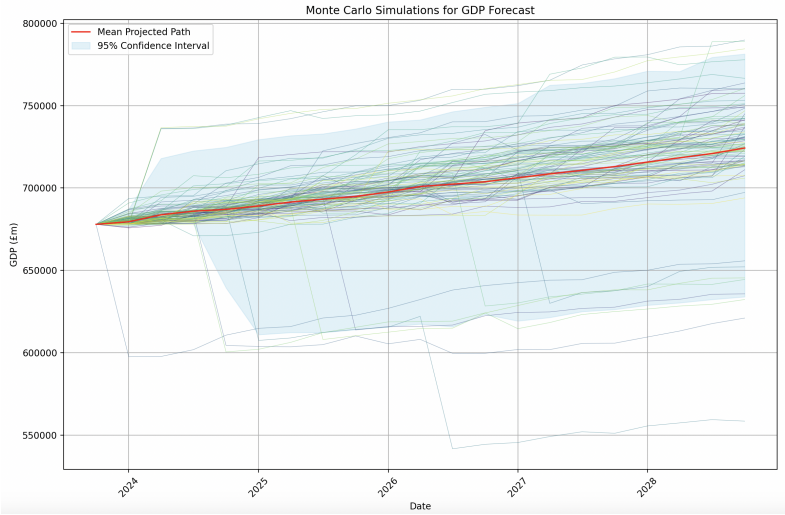

Figure 11: Monte Carlo simulation of GDP.

This data plotted above allows us to consider a large array of outcomes for GDP that occur with a non-negligible probability. As we can see the mean path, plotted in red, indicates steady GDP growth which seems the most likely outcome.

Youth Unemployment Rate The UK youth unemployment rate has shown consistent volatility throughout the years. This volatility emphasises the necessity for government preparation. This preparation can be considered as funding the education sector and ensuring there is no over-funding within other sectors causing a lack of resources. This Monte Carlo simulation will allow us to consider cases where we experience extreme volatility within this variable and also when it is stable.

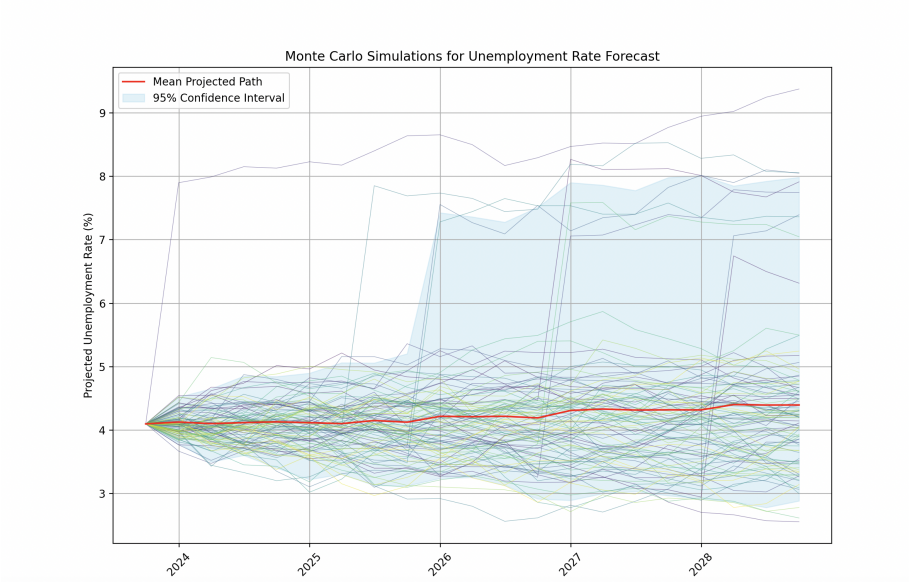

Figure 12: Monte Carlo simulation of youth unemployment rate.

These simulations provide a multitude of scenarios that youth unemployment could face. As shown by the 95% confidence interval there is a general upside expected. This seems reasonable from the time series plot of youth unemployment rate and we can see we are at a low point if we consider historical data.

Life Expectancy A Monte Carlo simulation of life expectancy will allow us to consider various circumstances where life expectancy will differ. Due to the nature of the life expectancy time series plot, which shows a fairly consistent incline in life expectancy throughout time, we expect to also see a fairly consistent set of paths for the Monte Carlo simulations with a relatively tight confidence interval. This plot can be found below:

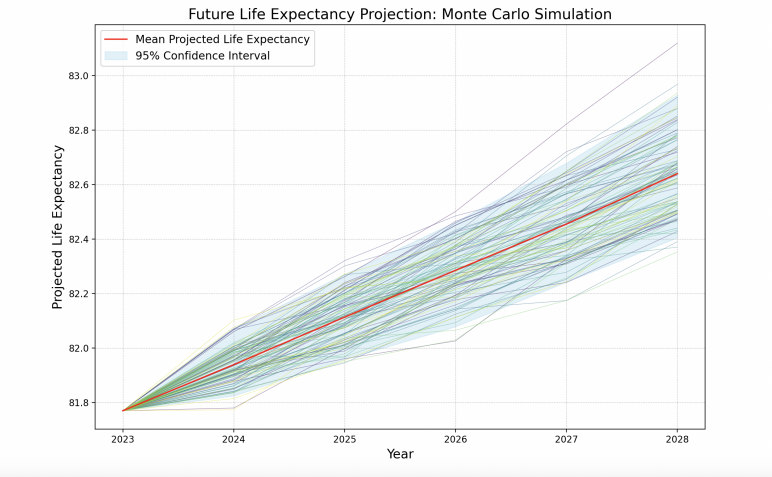

Figure 13: Monte Carlo simulation of life expectancy.

As expected, a very tight 95% confidence interval compared to the other variables is identified. One could consider that these simulations aren't providing an extremely wide array of outcomes for our life expectancy variable. However, it is important to note once again that this Monte Carlo simulation only displays paths which are of non-negligible probability. This indicates that this simulation does in fact provide a near exhaustive array of potential outcomes.

These simulations collectively have provided us with sufficient data to now train our machine-learning model to distribute government budget effectively while trying to optimise these variables with respect to funding in the correlated sector.

5 Model Setup

5.1 Overview

The intricate challenge of distributing government budget while facing uncertainty within variables is a problem that government bodies around the world face daily. Methods such as machine learning, and in particular reinforcement learning, provide a potentially effective solution. The first definition of machine learning was given by Arthur Samuel [12] which says, "Machine Learning is the field of study that gives computers the ability to learn without being explicitly programmed". Furthermore, M. Wu et al. [13] gives us a suitable definition for deep reinforcement learning (RL) - Reinforcement learning is a machine learning technique that learns an optimal decision policy based on defined rewards and observations of an environment.

To provide more insight into reinforcement learning it would be appropriate to include an example of how this technique can be used in a basic scenario. Consider a game of chess, we will define some variables informally. The **environment** is the chessboard along with all its pieces. The **state** is the current arrangement of pieces of the chessboard; The **action** is all legal moves on the chess board and the **reward** is the feedback given from the environment, i.e. positive reward for moves that improve the player's position and negative reward for moves that worsen the position. Using this data we have the **agent**; the agent exists within the environment. The agent is in a state at all times and it aims to complete actions which maximise reward. At first, the agent would move randomly and each time would receive feedback in the form of a reward. From this, it could tell whether this is a good or bad move. For example, in a simple form, taking a piece could be a positive reward, losing a piece could be a negative reward and checkmating could be a large positive reward. Over many games, the agent will learn what moves to make in which positions to optimise the reward and begins to form a strategy for the game, this is what we refer to as the **policy**. This would be effective as, unlike humans, the agent could play millions of games in a short space of time.

5.2 Model Design

Within reinforcement learning, there are many different techniques used to implement this. For this project, Proximal Policy Optimisation (PPO) was best suited. This method works by iteratively improving the policy, π, to optimally distribute government budget. By receiving feedback from the environment, the PPO algorithm tunes the policy to effectively distribute budget when under uncertain conditions. The feedback system which is represented as a reward function is designed so that the agent's policy, will be to maximise this function. The aim is to maximise GDP and life expectancy while also minimising youth unemployment rate with respect to funding the correlated sector. This reward function can be set up as so:

$$R = G(f_i) + L(f_h) - Y(f_e) - P, \tag{8}$$

where R is the reward, G is GDP as shown in Equation 1; L is life expectancy as shown in Equation 2, Y is youth employment rate as shown in Equation 3 and f_i, f_h, f_e is funding within the infrastructure, healthcare and education sectors respectively. We also have the P term involved in this expression, this can be considered the penalty term. The penalty term will be made up of three components. To represent this we first need to define B_{tot} as the total amount of budget available and B_{alloc} as the amount of allocated budget, then, the penalty cost, P, is defined as:

$$P = \begin{cases} 2 \cdot (B_{\text{alloc}} - B_{\text{total}}) & \text{if } B_{\text{alloc}} > B_{\text{total}}, \\ (0.9 \cdot B_{\text{total}} - B_{\text{alloc}}) & \text{if } B_{\text{alloc}} < 0.9 \cdot B_{\text{total}}, \\ 0 & \text{otherwise.} \end{cases} \tag{9}$$

The first case infringes a penalty of factor 2 for exceeding budgetary decisions. This indicates to the agent that this is not allowed. The second case takes into account if the agent does not distribute the budget to at least 90%. This is of a lower factor than the first one as it is not essential but important. We use this to ensure the model does not under-distribute budget. The third term signifies no penalty induced. We also have to consider the normalisation of our variables due to the large difference in them. For example, a typical value of G is approximately $69,000$, comparing this to a typical L value of 81.8 we can see a noticeable difference. This data needs to be normalised as if we considered optimising R without normalising the variables, the agent would realise over time that the best way to maximise this function would be to maximise G and would essentially consider G to have higher importance when this is not what the model implies. Now we consider a set of data X, we normalise it using Min-Max Scaling as so:

$$X_{\text{norm}} = \frac{X - X_{\text{min}}}{X_{\text{max}} - X_{\text{min}}}. \tag{10}$$

Therefore the agent's goal is to maximise the function:

$$R = G_{\text{norm}}(f_i) + L_{\text{norm}}(f_h) - Y_{\text{norm}}(f_e) - P. \tag{11}$$

The environment the agent exists in is represented by our Monte Carlo simulations and the given budgetary allocation. Within this environment, our agent can perform actions. The actions the agent can perform is to change the budgetary allowance in order to fulfil our policy - maximising the reward function. Also, the state at time t, s_t, of our system is the current GDP, life expectancy and youth unemployment rate as well as the remaining budget. The relationship between the agent and the environment can be represented in a diagram:

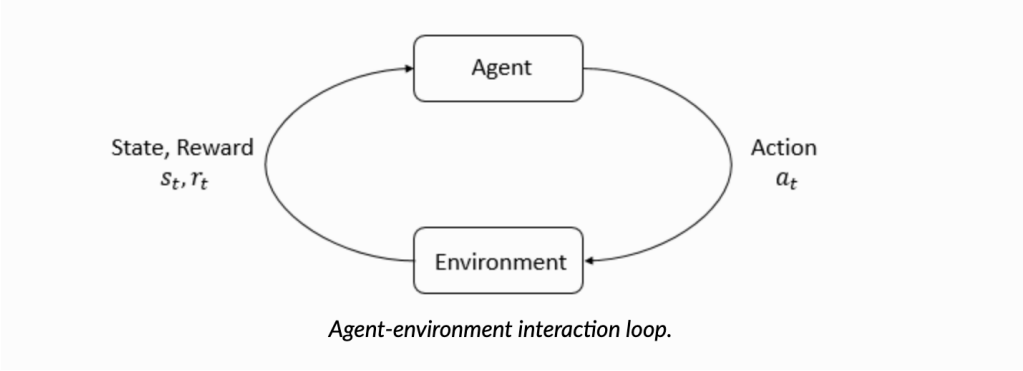

Figure 14: Diagram to represent the relationship between the agent and the environment given by OpenAI's course on reinforcement learning [14].

As we can see here the agent begins in a state and acts on the environment at time t with action a_t. The environment returns the new state, s_t and the reward, r_t. The cycle continues until the process is finished. The repetition of this cycle creates a trajectory, τ, which is made up of a sequence of states and actions, we can represent this as $\tau = (s_0, a_0, s_1, a_1, \dots)$. It would be useful to note that our trajectories, τ, obey the Markov property, this is that the system is memoryless or as given by D.Elavarasan et al. [15], "the likelihood of changing to a specific state is reliant exclusively on the present state". If we zoom into one particular part of a hypothetical trajectory we can see the process within the environment.

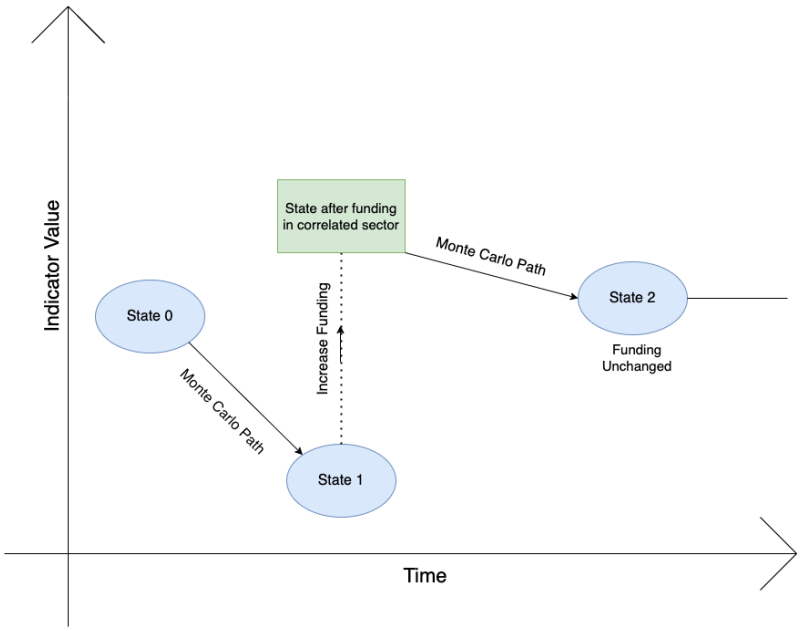

Figure 15: Example of how the environment will work within the Monte Carlo simulation.

In the above diagram, we consider a trained agent in an environment intending to maximise a variable. Beginning in state 0 we can follow this process to state 1 which is guided by the Monte Carlo simulations performed earlier. The agent can now react to being in this new state, which is lower, by increasing the funding. As we can see this increase in funding instantly increases the indicator value - this assumption is made to simplify our model. This process repeats while the agent refines the policy over thousands of simulations.

It is essential for the agent to understand the value of each state to justify actions, to consider this we use the expected return. The recursive relationship between the current state and the expected future reward can be modelled by the Bellman Equations, also known as the Value functions. Firstly, we must consider a discounting factor, γ. This factor is key in reinforcement learning. We take the parameter $0 \leq \gamma \leq 1$ to quantify how much future reward is valued compared to immediate reward. Therefore, the agent can be trained to value future expected rewards rather than just taking into account immediate reward. We can use this to alter the quantity of the expected reward k steps into the future by taking the discounted value as γ^k. Using this technique we determine the optimal balance between immediate reward and future reward in our model allowing us to ensure that the agent does not disproportionally prioritise actions, which would lead to high immediate rewards but diminished future rewards. We consider γ to be a 'hyperparameter' such that a higher value of γ indicates a higher focus on future rewards. A hyperparameter is a model parameter which controls the learning process. Each hyperparameter will need tuning to ensure an optimal model. The Bellman Equations are key within deep reinforcement learning to the agent's learning process. They allow the agent to make decisions to optimally distribute government resources.

Firstly, we can consider the State-Value function, denoted as $V^\pi(s)$. This is essentially the expected return when the agent follows a policy, π, when in state s which is from the set of all possible states, S. This is defined as:

$$V^\pi(s) = \mathbb{E}_\pi[R_{t+1} + \gamma V^\pi(S_{t+1}) \mid S_t = s] \tag{12}$$

This equation exhibits that the value of a state under policy is the expected value of the immediate reward, R_{t+1}, plus the discounted value of the following state, S_{t+1}. Relating this to our project, we find that this equation predicts future budgetary performance based on current economic indicators. Following, the Action-Value function, denoted as $Q^\pi(s, a)$

can be considered to be the expected return after an action, a, which is from the set of possible actions, A, performed in state s while under the policy π. In our case, a is the percentage budget change in each sector. This can be defined as:

$$Q^{\pi}(s,a) = \mathbb{E}_{\pi}[R_{t+1} + \gamma Q^{\pi}(S_{t+1}, A_{t+1}) \mid S_t = s, A_t = a] \tag{13}$$

This allows the agent to calculate the expected potential reward of several different actions and informs on how the budget should be allocated to optimise indicator outcomes. The Optimal State-Value function $V^*(s)$ allows the agent to consider the maximum expected return from any state while taking into account the best policy. We can define this as:

$$V^*(s) = \max_a \mathbb{E}[R_{t+1} + \gamma V^*(S_{t+1}) \mid S_t = s] \tag{14}$$

In relation to this project, this can be understood to be the best possible outcome for our indicator variables given the current state. Finally, consider the Optimal Action-Value function $Q^*(s,a)$. This function outputs the expected return after taking action a while in state s. This can be defined as:

$$Q^*(s,a) = \mathbb{E}[R_{t+1} + \gamma \max_a Q^*(S_{t+1}, a) \mid S_t = s, A_t = a] \tag{15}$$

This will allow the agent to calculate which combination of budget allocations will result in the optimal expected indicator values in future states. These equations together allow the agent to calculate the result of its actions and iteratively update its strategy. These iterative improvements are called policy updates. Furthermore, the usage of PPO only allows policy updates which are within a "trust region". This trust region can be defined as the bounded region in which we allow the policy to be updated. This is used to prevent the policy from being updated too drastically which can cause catastrophic forgetting. Catastrophic forgetting is a phenomenon in which the agent forgets all previous knowledge which has been learnt due to making an overly aggressive policy update. We can define this trust region mathematically by "clipping" the ratio of probabilities of selecting an action under a new policy to the current or an old policy. The clipping mechanism constrains the ratio of the probabilities of selecting an action under the new policy to the probability under the old policies which is defined as:

$$r_t(\theta) = \frac{\pi_{\theta}(a_t \mid s_t)}{\pi_{\theta_{\text{old}}}(a_t \mid s_t)}, \tag{16}$$

where $r_t(\theta)$ denotes the ratio of the new policy's probability of taking action a_t given its currently at state s_t to that of the old policy. We then "clip" this function to ensure the policy is kept within a predetermined bound. The clipping function is applied as follows:

$$\text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon), \tag{17}$$

where ϵ is a hyperparameter. This ensures the ratio stays in the interval $[1-\epsilon, 1+\epsilon]$. If, for example, ϵ is set to 0.2, this means we apply the constraint $0.8 \leq r_t(\theta) \leq 1.2$. So for example, if we found $r_t(\theta) = 1.4$ this value would be clipped to 1.2 altering the extent to which the policy is updated. With the agent continuously updating the policy towards higher rewards and the use of clipping we ensure that the agent develops a robust policy for distributing the government budget.

We now introduce α, the learning rate. This quantifies how much we adjust the model's parameters in order to reach a maximum. PPO uses a technique called gradient ascent. This is an iterative process which maximises the reward function by repeatedly moving towards the function maxima. This process works by initialising a policy model. This includes choosing the model parameters, including γ, the discount factor; ϵ the clipping parameter and the learning rate, α. A larger learning rate may cause a faster convergence

however it is also more likely to 'overshoot' the maximum the plot below demonstrates this:

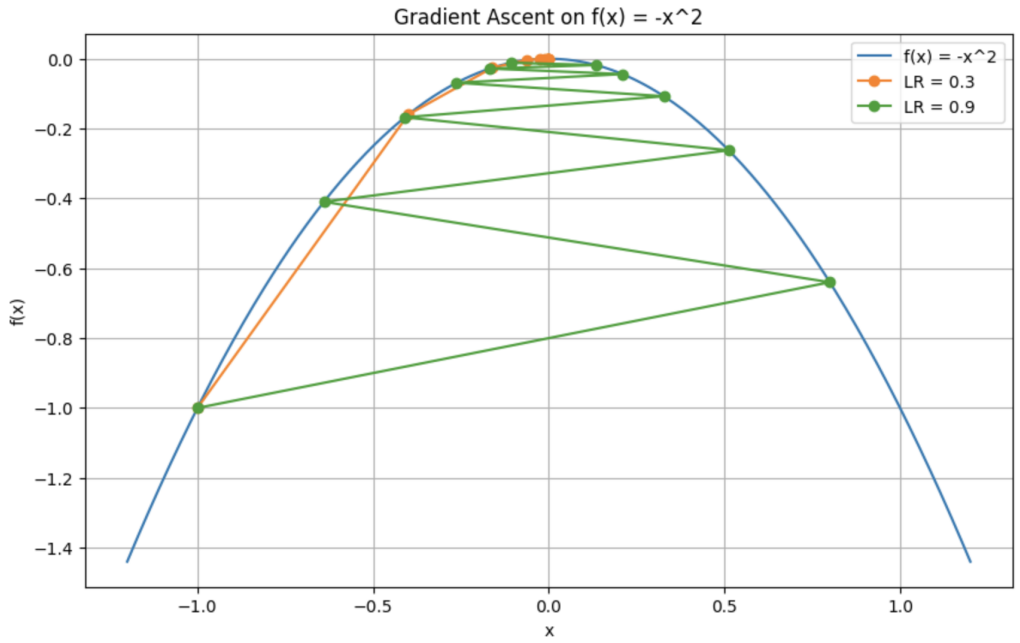

Figure 16: Demonstration of different learning rates on reaching a maximum using a gradient ascent method. Note - LR here means learning rate.

It is clear to see that the higher learning rate takes more iterations to reach an optimal value. Also, looking closely, it goes past the maximum and obtains a suboptimal value. On the other hand, the lower learning rate, represented by the orange line, finds a higher maximum in fewer iterations. However, it is important to note that the key aspect of this is to highlight how the higher learning rate goes past the maximum. On a differently shaped curve, it is usually the case that the higher learning rate leads to a faster convergence.

After this, we take steps to reach different states and at each state the agent will update the policy and parameter, in turn, gradually increasing the reward to a maximum. Some challenges include the agent may converge to a local maximum rather than a global one. This will cause convergence to a suboptimal policy. To solve this we can use multiple random initialisation points and find which yields the maximum reward. Another parameter to consider is the batch size, r. This is the number of training examples used within one iteration of the model update. A smaller batch size will generally result in a faster convergence. However, since we are using a smaller sample size we generally see a higher variance. This means that we generally prefer a higher batch size but computational demand also needs to be taken into account. Increasing the batch size too much will require high levels of computational power so it is important to find a balance. When we consider the number of episodes, n, which represents the number of complete processes from start to finish, there are a few factors which need to be considered. As we increase n, the model is exposed to more situations within the training phase. This in general is positive but increasing this too high can waste computational resources while also posing the threat of making the policy too complex and overfitting it. These hyperparameters can be inputted into the model as shown by a simple algorithm for our PPO model [4]:

Algorithm 4 Simplified Proximal Policy Optimisation (PPO) for Budget Allocation

Initialise policy π with random parameters θ
Initialise environment with economic data
Define hyperparameters: α, γ, ϵ, r, n
repeat
    Collect data by executing policy π in the environment
    Compute expected returns and advantages
    Optimise policy π using gradient ascent and clipping
    Update policy if improvement is observed
until Convergence or maximum iterations reached

5.3 Training Process

5.3.1 Parameter Tuning Process

It is important to emphasise the model's reliance on parameters in machine learning. The process behind tuning parameters involves systematically and progressively adjusting parameters to eventually fine-tune your model. As we want this model to relate as closely to real life as possible without causing over-complexity, it will be useful to consider government spending in these sectors historically. Therefore we will use the budget from previous years' allocation as a guideline. Within infrastructure, the Office of National Statistics [16] noted there was funding of £21.525 billion in this sector over the whole year. Within healthcare, the Office of National Statistics [17] estimated the total funding into this sector to be £214.4 billion in 2018. Finally, within the 2017/18 academic year, the Institute for Fiscal Studies [18] reported the UK government approximately £90 billion in spending within education. These values will guide us to set the budget appropriately to £300 billion per year. This is lower than the invested amount in 2018 into these sectors of £325 billion since we want to encourage the agent to behave conservatively. It is worth noting that the year 2018 was used as it is recent and was unaffected by COVID-19 which caused fluctuations in funding. This could cause implications in the model as these values don't follow general trends.

Following this variable selection, we must now consider hyperparameters. We include here the learning rate (α), discount factor (γ), batch size (r), number of episodes (n) and preemptively set the clipping factor, ϵ, to 1.2. Using a random search algorithm, 5, we can quickly decipher which set of parameters outputs the best mean reward.

Algorithm 5 Random Search for Hyperparameter Tuning in Reinforcement Learning

Define the environment for the RL model.
Specify the hyperparameter space: learning rates, batch sizes, number_episodes, discount factor.
Initialise the number of iterations for the search, N.
Initialise variables for tracking the best reward and hyperparameters.
for $i = 1$ to N **do**
    Sample a set of hyperparameters from the specified space.
    Train the RL model using the sampled hyperparameters.
    Evaluate the model's performance with a set number of episodes.
    if current model's performance $>$ best recorded reward **then**
        Update the best reward and best hyperparameters.
        Output the performance of the current model.
    end if
end for
Return the best set of hyperparameters and the corresponding reward.

Using the parameters $\alpha = \{1 \times 10^{-3}, 1 \times 10^{-4}, 1 \times 10^{-5}\}$, $\gamma = \{0.9, 0.95, 0.99\}$, $r =$

$\{64, 128, 256\}$ and $n = \{10, 20, 30\}$ we can determine the best combination of these for our model. These combinations provide a broad range of potentially optimal values without it being too computationally intense. This plot allows us to explore the result of these combinations.

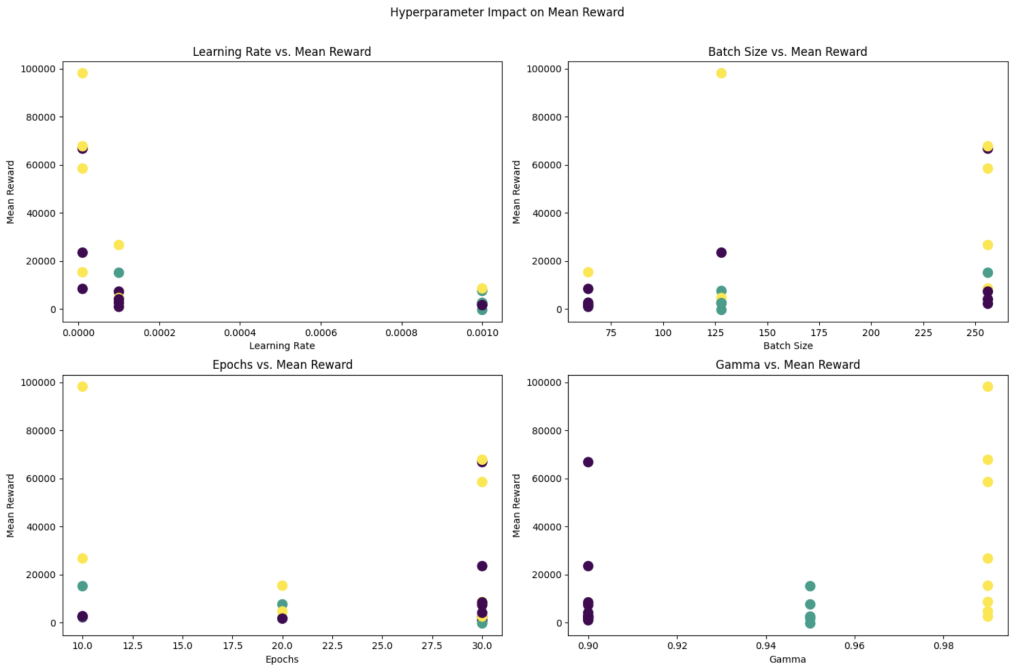

Figure 17: Reward values plotted for various combinations of hyperparameters.

Here we can see clearly that a smaller learning rate correlates with a higher reward. We see for the number of episodes, 10 yields the highest reward. This links back to the potential that anything above this could be overfitting the model. We also see the model prefers a slightly higher discount factor as we see higher reward values for 0.99 than 0.95 and 0.90. This implies that in this model there is more value in planning for potential future reward than instantaneous reward. Finally, there is a preferred batch number of 128, although we can see that on average 256 outputs a higher reward. This could be due to variance in the model, but since they are similar, we will choose 128 to avoid overcomplexity. The highest reward output is given by the hyperparameters $[\alpha, \gamma, r, n] = [1 \times 10^{-5}, 0.99, 128, 10]$ respectively. Now we have effectively tuned the model we input these hyperparameters and find a mean reward of 132131.5 with a standard deviation of 4065.7. In comparison, this is much higher than other hyperparameter combinations which resulted in mean reward values as low as 442 with a standard deviation of 572. This indicates some combinations of hyperparameters were producing negative reward values which is an indication of a very ineffective system. We must also note that due to normalisation these results are comparative but not a quantitative measure of the increase.

5.4 Model Analysis

The evaluation of the policy developed by our agent involves delving into how the agent 'thinks' in various states. To do this we can consider the distribution of the action value at different states. This has been normalised from $[-1, 1]$ with -1 and 1 respectively representing minimum and maximum emphasis of funding in the correlated sector. This can be done by taking the trained agent, putting it in various scenarios and recording the action taken by the agent. Consider all indicators at a 'baseline' state. This means that the indicator variables are at values which are neither considered high nor low. Note that in the following bar charts, the values for each sector are overlaid rather than stacked on top of one another.

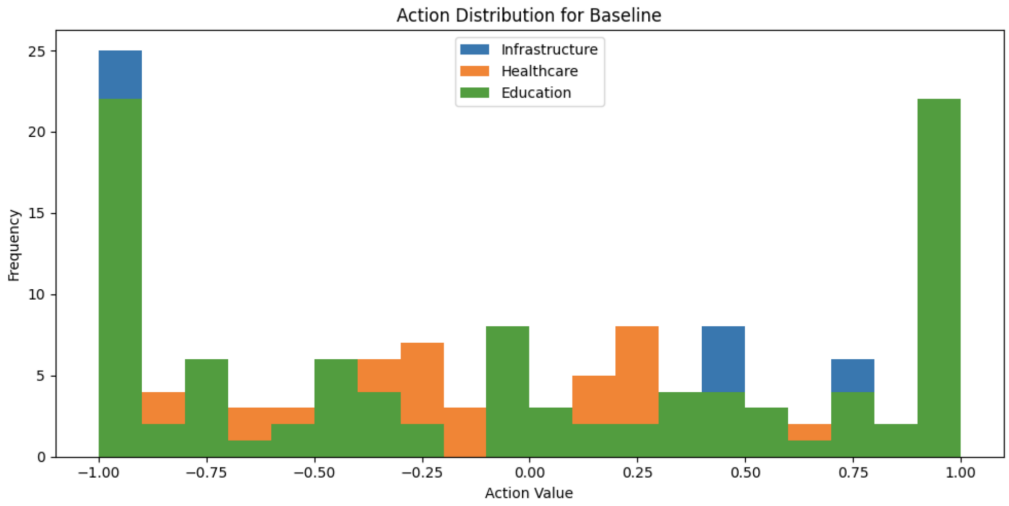

Figure 18: The distribution of action values at a baseline state for GDP, youth unemployment rate and life expectancy.

Here in the baseline scenario, we can see the agent takes an extreme allocation strategy towards the education sector, shown by the high bars at both extreme action values. This means that the agent is likely to invest heavily or completely withhold funding from this sector. In terms of the infrastructure sector, we see the agent tends to withhold funding completely, demonstrated by the high value at the leftmost bar. However, this is not always the case as we can see that occasionally the agent tends to increase funding in this sector as represented by the blue bars on the positive action-value side of the plot. Now, considering the healthcare sector, we can see in general the agent tends to take actions which on average, but not always, hold back funding in this sector. Overall, we can see the agent tends to act conservatively when at a baseline state. We can imply the agent is trying to save resources in this case for future negative situations where more funding is required. It is also important to notice that the agent is biasing decisions towards education signified by the amount of green area we see in this plot. Although this could simply just be part of the policy, there is a potential it is caused by bias in the system. Now we consider a scenario where we have high GDP, high life expectancy and a low youth unemployment rate. This is where all the economic indicators are in a positive state.

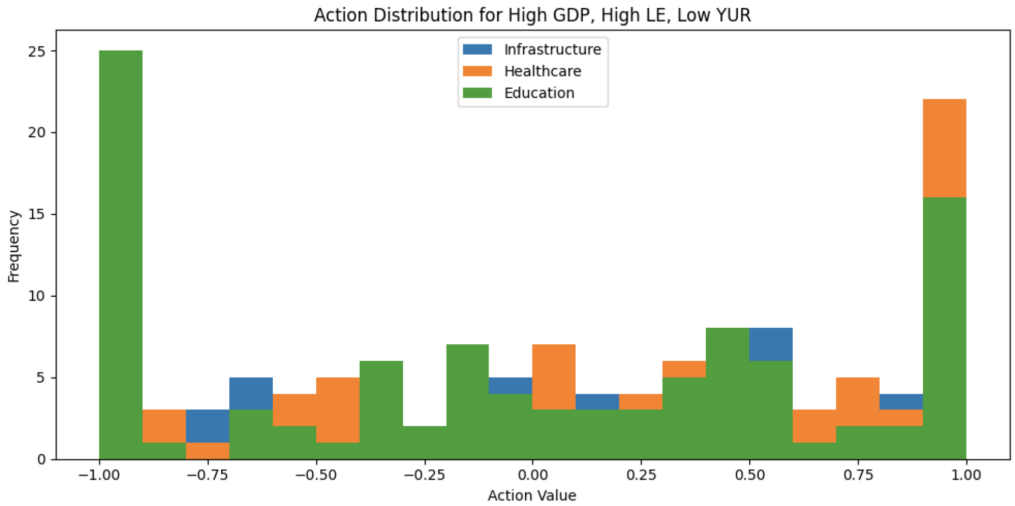

Figure 19: The distribution of action values at a high GDP, low youth unemployment rate and high life expectancy values.

This plot shows that the healthcare sector is favoured in terms of funding when all

indicators are in a positive position, this is highlighted by the rightmost orange bar. This could indicate the model preferring a lower youth unemployment rate to a higher GDP or youth unemployment rate. We can also imply the potential of the agent feeling the need to prepare for youth unemployment rates increasing rather than GDP or life expectancy decreasing. We see a balance in the decisions for infrastructure and education implying no major preference on the correlated indicators. We again see a bias in education-related budgetary decisions. Now, consider a low GDP, low life expectancy and a high youth unemployment rate scenario. This is a scenario where all indicator variables are in a sub-par state.

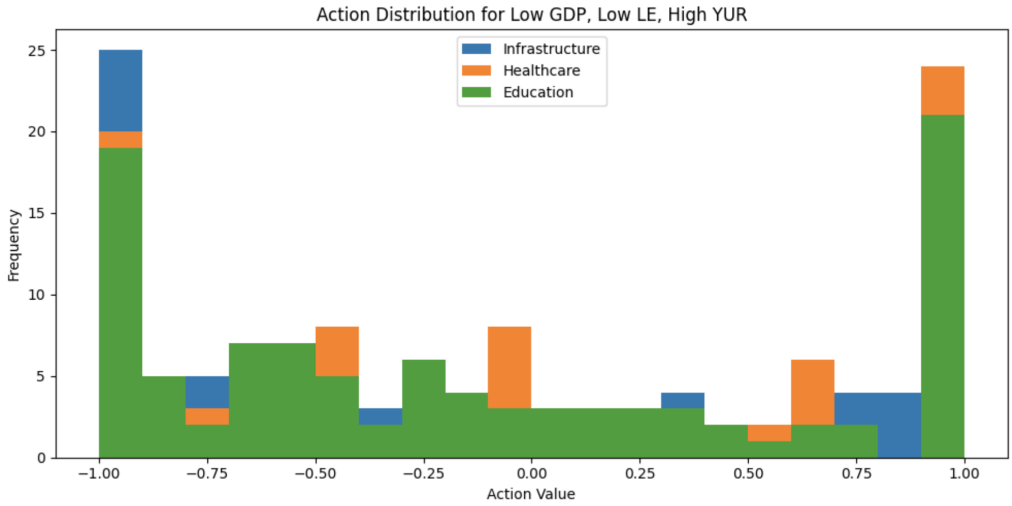

Figure 20: The distribution of action values at a low GDP, high youth unemployment rate and low life expectancy value.

As we can see from this plot, education and healthcare are heavily favoured when all indicators are in a deficient state, highlighted by the overwhelming rightmost bars on the chart. However, this is balanced out by the large leftmost bars representing education and healthcare. Also, interestingly, despite GDP being in a dissatisfactory state, the agent developed a policy which tends to neglect funding in infrastructure for the most part. This could be due to factors such as having a much higher bias towards funding in education or healthcare or the agent learnt that GDP tends to naturally recover more often than not. Both these cases together offer a reasonable and potentially accurate explanation. However, despite the fact that these assumptions are fair and have reasonable backing it is hard to truly determine the reasoning behind the policies implemented by machine learning, it is important to note that these are just likely suggestions.

Overall, the model appears to accurately recognise different demands placed on it by the economy. It tends to adapt sufficiently and prepare rigorously. This aspect can largely be put down to the high discount factor within our model. The bias of education throughout these scenarios could potentially represent a strategic method to maximise the reward. Once again, it is important to note that it is difficult to tell how much this is linked to bias within the model or other underlying factors where it is, in fact, necessary to place more emphasis on the education sector's budgetary decisions.

6 Discussion

An effective deep reinforcement learning model has now been made which distributes government budget to optimise the given indicator variables while facing economic uncertainty. This model is both reactive and proactive which is ideal when we consider the potential of a volatile economy. The reactive aspect of the model was developed through

the extensive training provided whereas the proactive attribute comes from the parameter tuning as well, particularly the tuning of the discount factor, γ. Noticeably, the model tended to prioritise decisions towards altering the funding within the education sector. This could be an indication of this sector being crucial to the overall health of the UK economy according to our reward function which did not place any additional weight on the youth unemployment rate variable.

Considering our solution, it is important to note that although the reward obtained is high and indicates an effective system, there is a strong potential for slightly more effective results which could be derived using a wider range of hyperparameter testing. The further tuning of these parameters could create an even better model. However, this would be computationally intense and require industrial equipment. It is also important to note that this particular hyperparameter tuning is only effective for UK data, other countries' data would need to be modelled to an appropriate ARIMA model and the hyperparameters would once again have to be fine-tuned. Despite this, we have shown that machine learning is an effective approach to distributing government resources. The use of ARIMA modelling and Monte Carlo simulations created an effective method of forecasting a wide range of potential future economic scenarios. This, in turn, allowed the training of the model to be exhaustive and realistic. This offers a look into the potential of machine learning to aid policy-making to inform government bodies and enhance their decisions to provide effective solutions. However, it is important to consider the many assumptions that were made within this project. We considered the effects of funding to be instantaneous, while in reality, this would not be the case. The effects would occur at a later date, over time and also likely not at a linear rate. Further analysis of this project could attempt to model the rate at which the effects of funding occur in the correlated sector. An improved model could also take into account other influences on our economic indicators. For example, GDP could be linked to other factors such as investment in research and development which can cause GDP growth. Furthermore, the implementation of a model of this type would also require consideration regarding limiting the growth of our economic indicators due to some detrimental effects it can have. Again considering GDP, if the GDP growth rate is too high this could cause high levels of inflation which would need to be contained. Also, to improve the applicability of our machine learning model we could consider further investigation into the ARIMA modelling of our data. A more accurate model would lead to more accurate path generation within our Monte Carlo simulations, creating a more realistic training environment. This may better prepare the model to handle real-life situations more effectively.

7 Acknowledgements

I would like to thank Prof. Onno Bokhove for his guidance and support throughout this project. The expertise and insights provided have been invaluable in the development of this project. I am grateful to have worked under his supervision and for his rigorous feedback throughout.

8 References

[1] A.L. Samuel. "Some Studies in Machine Learning Using the Game of Checkers". In: *IBM Journal of Research and Development* 44 (1959), pp. 206–226.

[2] F. Rosenblatt. "The Perceptron: A Probabilistic Model for Information Storage and Organization in The Brain". In: *Psychological Review* 65.6 (1958), pp. 386–408.

[3] G. E. Hinton, Y. Bengio, and Y. LeCun. "Deep Learning". In: *Nature* 521 (2015), pp. 436–444.

[4] J Manyika et al. “Big data: The next frontier for innovation, competition, and productivity”. In: (2011).

[5] N Mehmetaj and N Xhindi. “The Impact of Increased Education Spending on Youth Unemployment Rates”. In: *Journal of Educational Economics* 10.12 (2021), pp. 293–305.

[6] T Watt, A Charlesworth, and B Gershlick. “The Relationship Between Health Spending and Life Expectancy: An Empirical Analysis”. In: *Journal of Health Economics* 39.4 (2021), pp. 621–632.

[7] M Brueckner. “Evaluating the Effects of Infrastructure Spending on Economic Growth”. In: *Journal of Infrastructure Economics* 14.11 (2021), pp. 543–556.

[8] TrainDataHub. *How to Interpret ACF and PACF plots for identifying AR, MA, ARMA, or ARIMA models*. https://medium.com/@ooemma83/how-to-interpret-acf-and-pacf-plots-for-identifying-ar-ma-arma-or-arima-models-498717e815b6. [Accessed 15 March 2024]. 2019.

[9] Office for National Statistics. *GDP: Gross Domestic Product, volume index, seasonally adjusted*. https://www.ons.gov.uk/economy/grossdomesticproductgdp/timeseries/ybez/pn2. [Accessed 12 March 2024]. 2024.

[10] Office for National Statistics. *United Kingdom Life Expectancy*. https://www.macrotrends.net/global-metrics/countries/GBR/united-kingdom/life-expectancy. [Accessed 12 March 2024]. 2024.

[11] Macrotrends. *Unemployment: LMS - Labour Market Statistics*. https://www.ons.gov.uk/employmentandlabourmarket/peoplenotinwork/unemployment/timeseries/mgwy/lms. [Accessed 12 March 2024]. 2024.

[12] A. L. Samuel. “Some Studies in Machine Learning Using the Game of Checkers”. In: *IBM Journal of Research and Development* 3(3) (1959), pp. 210–229.

[13] M. Wu et al. “Real-time optimization of a chemical plant with continuous flow reactors via reinforcement learning”. In: *33rd European Symposium on Computer Aided Process Engineering* 52 (2023), pp. 457–462.

[14] OpenAI. *Agent-environment interaction loop*. https://spinningup.openai.com/en/latest/spinningup/rl_intro.html. Accessed on 29th March 2024. 2024.

[15] D. Elavarasan et al. “Forecasting yield by integrating agrarian factors and machine learning models: A survey”. In: *Computers and Electronics in Agriculture* 155 (2018), pp. 257–282.

[16] Office for National Statistics. *Developing New Measures of Infrastructure Investment*. https://www.ons.gov.uk/economy/economicoutputandproductivity/productivitymeasures/articles/developingnewmeasuresofinfrastructureinvestment/may2023. [Accessed 15 March 2024]. 2023.

[17] Office for National Statistics. *UK Health Accounts: 2018*. [Accessed 15 March 2024]. 2018. URL: https://www.ons.gov.uk/peoplepopulationandcommunity/healthandsocialcare/healthcaresystem/bulletins/ukhealthaccounts/2018.

[18] Institute for Fiscal Studies. *2018 Annual Report on Education Spending in England*. [Accessed 15 March 2024]. 2018. URL: https://ifs.org.uk/publications/2018-annual-report-education-spending-england.

A Appendix

A.1 Monte Carlo Simulation Generation

```
1
2  import numpy as np
3  import pandas as pd
4  from statsmodels.tsa.arima.model import ARIMA
5
6  folder_path = # FILE DIRECTORY HERE
7  GDP_file_path = #GDP FILE PATH HERE
8  unemployment_file_path = #YUR FILE PATH HERE
9  le_file_path = #LE FILE PATH HERE
10
11
12 gdp_data = pd.read_csv(GDP_file_path, names=['Quarter', 'GDP ( m )'])
13
14 years = 5
15 num_simulations = 10
16 def quarter_to_datetime(quarter_str):
17     year, qtr = quarter_str.split(' Q')
18     month = (int(qtr) - 1) * 3 + 1
19     return pd.to_datetime(f'{year}-{month:02d}-01')
20 gdp_data['Quarter'] = gdp_data['Quarter'].apply(quarter_to_datetime)
21 gdp_data.set_index('Quarter', inplace=True)
22 gdp_data['GDP ( m )'] = pd.to_numeric(gdp_data['GDP ( m )'], errors='coerce
       ')
23
24
25 def convert_time_to_date(time_str):
26     year, quarter = time_str.split(' Q')
27     month = (int(quarter) - 1) * 3 + 1
28     return pd.to_datetime(f'{year}-{month:02d}-01')
29 unemployment_data = pd.read_csv(unemployment_file_path, skiprows=61, nrows
       =273-62, names=['Time', 'Unemployment Rate (%)'])
30 unemployment_data['Date'] = pd.to_datetime(unemployment_data['Time'].apply(
       convert_time_to_date))
31 unemployment_data['Unemployment Rate (%)'] = pd.to_numeric(
       unemployment_data['Unemployment Rate (%)'], errors='coerce')
32 unemployment_data.set_index('Date', inplace=True)
33
34 def process_life_exp_data(file_path):
35     data = pd.read_csv(file_path, skiprows=4, header=None, usecols=[0, 1])
36     data.columns = ['Year', 'LifeExpectancy']
37     data['Year'] = pd.to_datetime(data['Year'])
38     data['Year'] = data['Year'].dt.year
39     data.set_index('Year', inplace=True)
40     return data
41
42 data1 = process_life_exp_data(le_file_path)
43 # fit models
44 gdp_model = ARIMA(gdp_data['GDP ( m )'], order=(0,1,0))
45 gdp_model_fit = gdp_model.fit()
46
47 yur_model = ARIMA(unemployment_data['Unemployment Rate (%)'], order=(2,1,0)
       )
48 yur_model_fit = yur_model.fit()
49
50 # extract residuals
51 gdp_residuals = gdp_model_fit.resid
52 yur_residuals = yur_model_fit.resid
53 years = 5
54 quarters = years * 4  # Assuming 4 quarters per year
55 num_simulations = 1000
56
57
58 simulated_gdp_paths = np.zeros((quarters, num_simulations))
59 simulated_yur_paths = np.zeros((quarters, num_simulations))
60 # last known values
```

```
last_gdp_value = gdp_data['GDP ( m )'].iloc[-1]
last_yur_value = unemployment_data['Unemployment Rate (%)'].iloc[-1]

# monte carlo simulations for GDP
for i in range(num_simulations):
    random_residuals_gdp = np.random.choice(gdp_residuals, size=quarters)
    future_values_gdp = [last_gdp_value]
    for residual in random_residuals_gdp:
        future_value_gdp = future_values_gdp[-1] + residual
        future_values_gdp.append(future_value_gdp)
    simulated_gdp_paths[:, i] = future_values_gdp[1:]

# monte carlo simulations for YUR
for i in range(num_simulations):
    random_residuals_yur = np.random.choice(yur_residuals, size=quarters)
    future_values_yur = [last_yur_value]
    for residual in random_residuals_yur:
        future_value_yur = future_values_yur[-1] + residual
        future_values_yur.append(future_value_yur)
    simulated_yur_paths[:, i] = future_values_yur[1:]


life_exp_data = process_life_exp_data(le_file_path)


life_exp_model = ARIMA(life_exp_data['LifeExpectancy'], order=(2,0,0))
life_exp_model_fit = life_exp_model.fit()

# extract life expectancy residuals
life_exp_residuals = life_exp_model_fit.resid

# define simulation parameters
years = 5
num_simulations = 1000
simulated_life_exp_paths = np.zeros((years, num_simulations))

# last known value of life expectancy
last_life_exp_value = life_exp_data['LifeExpectancy'].iloc[-1]

# monte Carlo simulations for life expectancy
for i in range(num_simulations):
    random_residuals_life_exp = np.random.choice(life_exp_residuals, size=
        years)
    future_values_life_exp = [last_life_exp_value]
    for residual in random_residuals_life_exp:
        future_value_life_exp = future_values_life_exp[-1] + residual
        future_values_life_exp.append(future_value_life_exp)
    simulated_life_exp_paths[:, i] = future_values_life_exp[1:]
```

A.2 PPO Implementation

```
import gym
from gym import spaces
import numpy as np
from stable_baselines3 import PPO
from stable_baselines3.common.vec_env import DummyVecEnv
from stable_baselines3.common.evaluation import evaluate_policy
from sklearn.preprocessing import MinMaxScaler

class BudgetAllocationEnv(gym.Env):
    metadata = {'render.modes': ['human']}

    def __init__(self, simulated_gdp_paths, simulated_le_paths,
        simulated_yur_paths, sigma_G, sigma_L, sigma_Y, initial_B):
```

```
        super(BudgetAllocationEnv, self).__init__()
        self.action_space = spaces.Box(low=np.array([-1.0, -1.0, -1.0]),
                                       high=np.array([1.0, 1.0, 1.0]),
                                       dtype=np.float32)
        self.observation_space = spaces.Box(low=np.array([0, 0, 0, 0]),
                                            high=np.array([1, 1, 1, 1]),  # normalized values
                                            dtype=np.float32)


        self.sigma_G = sigma_G  #
        self.sigma_L = sigma_L
        self.sigma_Y = sigma_Y
        self.initial_B = initial_B

        self.simulated_gdp_paths = simulated_gdp_paths
        self.simulated_le_paths = simulated_le_paths
        self.simulated_yur_paths = simulated_yur_paths

        self.gdp_min, self.gdp_max = self.simulated_gdp_paths.min(), self.simulated_gdp_paths.max()
        self.le_min, self.le_max = self.simulated_le_paths.min(), self.simulated_le_paths.max()
        self.yur_min, self.yur_max = self.simulated_yur_paths.min(), self.simulated_yur_paths.max()

        self.scaler = MinMaxScaler()

        # scale to the min and max bounds
        self.scaler.fit([[self.gdp_min, self.le_min, self.yur_min],
                         [self.gdp_max, self.le_max, self.yur_max]])

    def reset(self):
        self.current_step = 0
        self.current_gdp_path = np.random.choice(range(self.simulated_gdp_paths.shape[1]))
        self.current_le_path = np.random.choice(range(self.simulated_le_paths.shape[1]))
        self.current_yur_path = np.random.choice(range(self.simulated_yur_paths.shape[1]))
        self.G = self.simulated_gdp_paths[0, self.current_gdp_path]
        self.L = self.simulated_le_paths[0, self.current_le_path]
        self.Y = self.simulated_yur_paths[0, self.current_yur_path]
        self.B = 300000000000*5  # initial budget

        scaled_values = self.scaler.transform([[self.G, self.L, self.Y]])
        self.G, self.L, self.Y = scaled_values[0]

        return np.array([self.G, self.L, self.Y, self.B / self.B], dtype=np.float32)  # normalize budget

    def step(self, action):
        # convert the action to budget allocation amounts
        f_i, f_h, f_e = self.convert_action_to_funding(action)

        # calculate the stochastic elements
        epsilon_G = np.random.normal(0, math.sqrt(gdp_data['GDP ( m )'].std()))
        epsilon_L = epsilon_L = np.random.normal(0, math.sqrt(data1['LifeExpectancy'].std()))
        epsilon_Y = np.random.normal(0, math.sqrt(unemployment_data['Unemployment Rate (%)'].std()))#np.random.normal(0, self.sigma_Y)

        # calculate the new indicators
        self.G += 4.17e-3 * f_i / self.B + epsilon_G
```

```
        self.L += 35 * f_h / self.B + epsilon_L
        self.Y += -10.81 * f_e / self.B + epsilon_Y

        # update budget
        self.B -= (f_i + f_h + f_e)

        # normalize the state values to be between 0 and 1
        scaled_state = self.scaler.transform([[self.G, self.L, self.Y]])
        self.G, self.L, self.Y = scaled_state[0]

        # calculate reward
        reward = self.calculate_reward(self.G, self.L, self.Y, f_i, f_h,
            f_e, self.B)


        done = self.B <= 0 or self.current_step >= self.simulated_gdp_paths
            .shape[0] - 1
        self.current_step += 1
        next_state = np.array([self.G, self.L, self.Y, self.B / self.
            initial_B], dtype=np.float32)

        return next_state, reward, done, {}

    def convert_action_to_funding(self, action):

        max_budget_pct_change = 0.1  # 10% of the total budget


        f_i = (action[0] * max_budget_pct_change + 1) * (self.B / 3)  #
            infrastructure
        f_h = (action[1] * max_budget_pct_change + 1) * (self.B / 3)  #
            healthcare
        f_e = (action[2] * max_budget_pct_change + 1) * (self.B / 3)  #
            education


        return f_i, f_h, f_e

    def calculate_reward(self, G_scaled, L_scaled, Y_scaled, f_i, f_h, f_e,
         B):
        # reverse scaling
        G, L, Y = self.scaler.inverse_transform([[G_scaled, L_scaled,
            Y_scaled]])[0]


        # apply impact of the budget allocations on the indicators
        G_impact = 4.17e-3 * f_i / B
        L_impact = 35 * f_h / B
        Y_impact = -10.81 * f_e / B

        # include stochastic elements
        epsilon_G = np.random.normal(0, math.sqrt(gdp_data['GDP ( m )'].std
            ()))
        epsilon_L = np.random.normal(0, math.sqrt(data1['LifeExpectancy'].
            std()))
        epsilon_Y = np.random.normal(0, math.sqrt(unemployment_data['
            Unemployment Rate (%)'].std()))


        G_real = G_scaled + G_impact + epsilon_G
        L_real = L_scaled + L_impact + epsilon_L
        Y_real = Y_scaled + Y_impact + epsilon_Y




        # calculate the reward
```

```
        reward =  G_real + L_real - Y_real

        # penalize for budget over-allocation or underutilization
        total_budget_allocated = f_i + f_h + f_e
        if total_budget_allocated > B:
            reward -= 2e-12 * (total_budget_allocated - B)  # Penalty for
                overspending
        elif total_budget_allocated < 0.9 * B:
            reward -= 1e-12*(0.9 * B - total_budget_allocated)  # Penalty
                for underutilization

        return reward



    def render(self, mode='human'):
        pass
import math

sigma_G = math.sqrt(gdp_data['GDP ( m )'].std())
sigma_L = math.sqrt(data1['LifeExpectancy'].std())
sigma_Y = math.sqrt(unemployment_data['Unemployment Rate (%)'].std())
initial_B = 300000000000 *5

env = BudgetAllocationEnv(simulated_gdp_paths, simulated_life_exp_paths,
    simulated_yur_paths, sigma_G, sigma_L, sigma_Y, initial_B)
env = DummyVecEnv([lambda: env])

# initialize the model with policy and environment
model = PPO("MlpPolicy", env, verbose=1, learning_rate=0.00001, n_steps
    =2048, batch_size=128, gamma = 0.99)

# train model
model.learn(total_timesteps=20000)

# evaluate the policy
mean_reward, std_reward = evaluate_policy(model.policy, env,
    n_eval_episodes=10)
print(f"Mean reward: {mean_reward}, Std Reward: {std_reward}")

model.save("ppo_budget_allocation")
model = PPO.load("ppo_budget_allocation")


\
```

Any additional data or code required, such as code for plots, is available upon request.

UNIVERSITY OF LEEDS

You must sign this (digitally signing with your name is acceptable) and include it with each piece of work you submit.

I am aware that the University defines plagiarism as presenting someone else's work, in whole or in part, as your own. Work means any intellectual output, and typically includes text, data, images, sound or performance.

I promise that in the attached submission I have not presented anyone else's work, in whole or in part, as my own and I have not colluded with others in the preparation of this work. Where I have taken advantage of the work of others, I have given full acknowledgement. I have not resubmitted my own work or part thereof without specific written permission to do so from the University staff concerned when any of this work has been or is being submitted for marks or credits even if in a different module or for a different qualification or completed prior to entry to the University. I have read and understood the University's published rules on plagiarism and also any more detailed rules specified at School or module level. I know that if I commit plagiarism I can be expelled from the University and that it is my responsibility to be aware of the University's regulations on plagiarism and their importance.

I re-confirm my consent to the University copying and distributing any or all of my work in any form and using third parties (who may be based outside the UK) to monitor breaches of regulations, to verify whether my work contains plagiarised material, and for quality assurance purposes.

I confirm that I have declared all mitigating circumstances that may be relevant to the assessment of this piece of work and that I wish to have taken into account. I am aware of the University's policy on mitigation and the School's procedures for the submission of statements and evidence of mitigation. I am aware of the penalties imposed for the late submission of coursework.

Student Signature	Nathan Scarrott	Date	03/04/2024
Student Name	Nathan Scarrott	Student Number	201522393