ANALYSIS OF THE UNEMPLOYMENT RATE

2 / 11 / 2023

Charlie Liu 520296576
Ting Yu Liu 520285488
Zishan Huang 520049949
Hongbin Zou 520384943

1. Introduction

To begin with, the best out-of-sample forecasting Root Mean Squared Error (RMSE) of our group is the **Neutral Network**, having an RMSE of **0.1728**. The unemployment rate, with its profound influence on the macroeconomy, consistently draws broad social concerns. Consequently, projecting future trends of the unemployment rate is crucially important across various sectors of society. In this report, various models are used to forecast the unemployment rate, each of which exhibit their distinct advantages. For instance, the ARIMA models demonstrate both AR and MA models' characteristics with differencing, while the Seasonal ARIMA model accounts for seasonality. Subsequently, we explored both the Simple Exponential Smoothing and Trend Corrected Exponential Smoothing models to determine whether a comparatively simplified approach would yield superior results. Finally, we delved into Neural Network Autoregression and Recurrent Neural Networks (RNN), including various variants like Long Short-Term Memory (LSTM) networks and Gated Recurrent Unit (GRU) networks.

2. Data pre-processing and exploratory data analysis (EDA)

To begin with, the data in 1978 was removed from the in-sample dataset. The reason is that the year 1978 is too distant from the current period, implying its data is potentially less pertinent for the modelling efforts. Additionally, it was found that the data in 1978 was incomplete, compared to other years in the dataset. Based on this analysis, the data for the analysis has been chosen from 1979 to 2017, and data in 1978 was considered not relevant.

During the preliminary data preprocessing, both the in-sample and out-of-sample datasets had no missing values or repeated records, showing that the data was ready for the next steps of analysis and modelling. To help with time-series studies, the "Date" columns in both datasets were converted to the DateTime format. This step made it easier to index by time and was important for the next stages of data visualisation. After this, the "Date" column was used as the main index to help look at trends, patterns, and other time-based features in more detailed studies.

To make sure the data was of great quality, we used the Interquartile Range (IQR) method to find any outliers in the in-sample dataset. Even though two outliers were found, we chose to use all the data for the next analysis. This choice was made because the IQR method is strict, and the outliers were not too different from the rest to be seen as wrong data.

After this, we made a statistical summary for the in-sample dataset. The results are displayed below.

Table 2.1

count	468.0000
mean	6.7817
std	1.9833
min	3.4992

25%	5.3125
50%	6.1866
75%	8.0900
max	12.6958

The data had values between 3.50% and 12.70%, showing big changes in unemployment. These changes might be because of things like changes in the economy, new policies, or big events. The close values of the mean and median tell us that the unemployment rates are spread out evenly. One thing to point out is the bigger gap between the 75th percentile and the highest value than between the 25th percentile and the lowest. This means that unemployment has a rapid ascent compared to a descent. The standard deviation of the data shows that the unemployment rate changes over time, but these changes are not too extreme and mostly stay around the average.

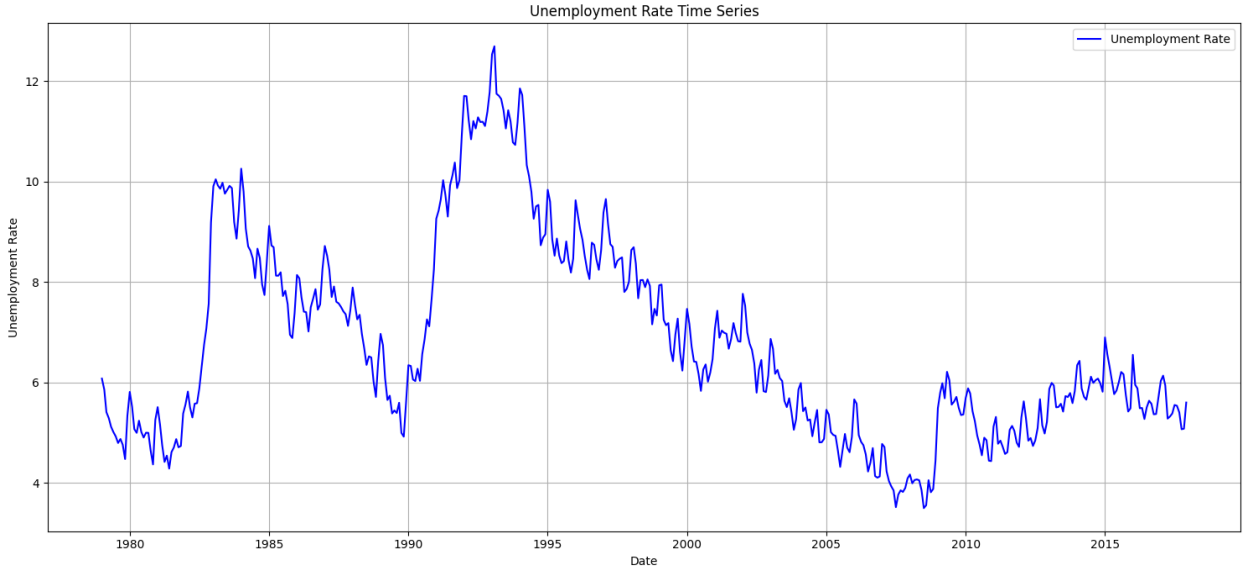

Figure 2.1

The visualisation of the unemployment rate's time series reveals the economic changes over the years. When we look closely at this series, there are clear fluctuations in the data, especially a peak in the early 1990s, followed by a downward trend. Another notable change is seen around 2008-2009, likely influenced by the global financial crisis of that time. This series effectively captures the regular changes in the unemployment rate over the time period mentioned.

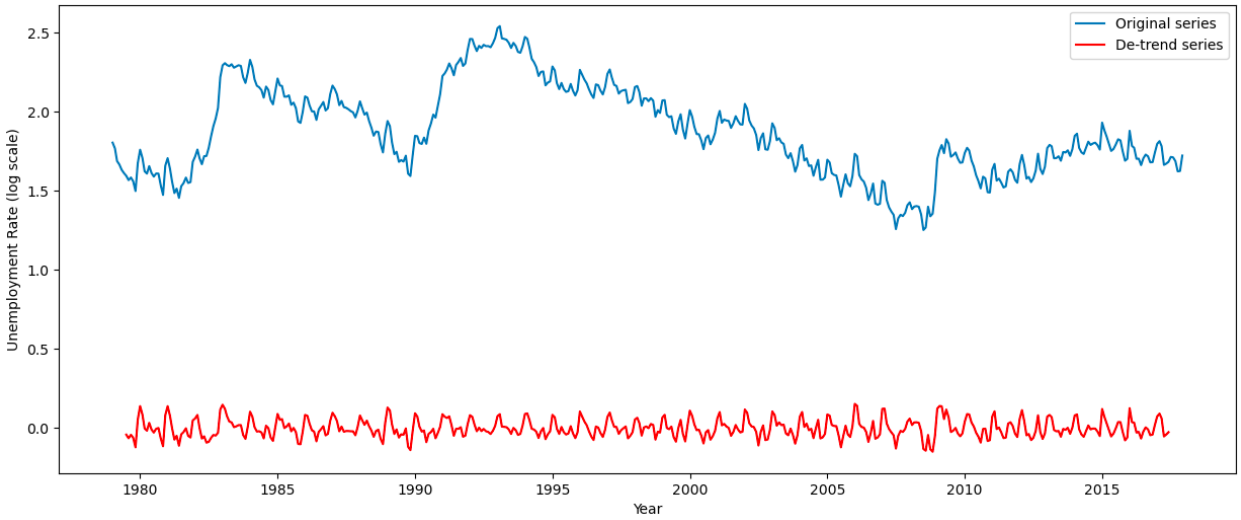

Figure 2.2

Upon observing the graph above, we see two distinct time series: the logarithmic transformation of the unemployment rate and its de-trended version. The de-trended series, shown in red, highlights the seasonality after removing the long-term trend. Its closeness to the zero line indicates the successful removal of the consistent long-term trend. By looking at this series, we can identify seasonality fluctuation of short-term changes in unemployment that might be hidden by the general trend.

Figure 2.3

The first plot above displayed the same time series of logarithmic transformations of the unemployment rate as we explained before. Delving into the next plot, it shows how unemployment has been trending over

time. By removing seasonality changes, this line gives us insights into the broader tendency of unemployment moves. Looking more closely, the next picture showcases unemployment's seasonality. These annual changes might be because of factors such as macroeconomic changes and potential policy impacts. Lastly, the last image displays the unemployment data but without the annual seasonality. Removing seasonality lets us figure out the main reasons for the trend of unemployment. This can be useful when trying to distinguish the effects of external factors.

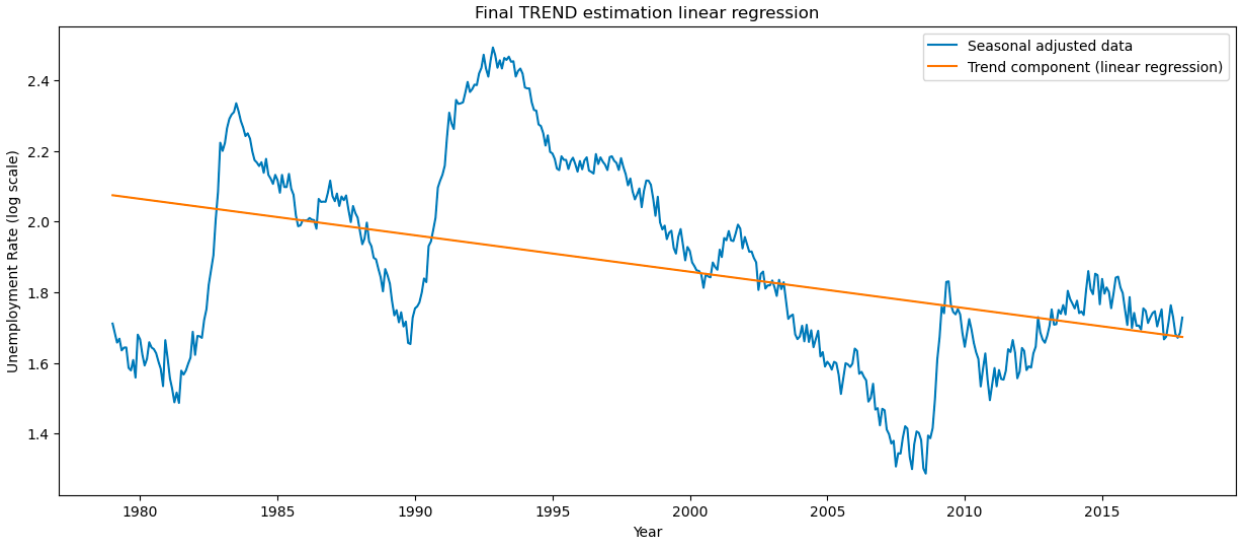

Figure 2.4

Upon examining the visualisation above, the short-term fluctuation still can be observed, implying the cyclic employment landscape. Simultaneously, an orange linear trend line simplifies the overall trend, indicating a steady downward tendency in unemployment over the years. This plot of the seasonal adjusted data and the linear regression trend line presents a comprehensive view of the unemployment pattern, revealing both the cyclic shifts and the general tendency which is necessary for further analysis.

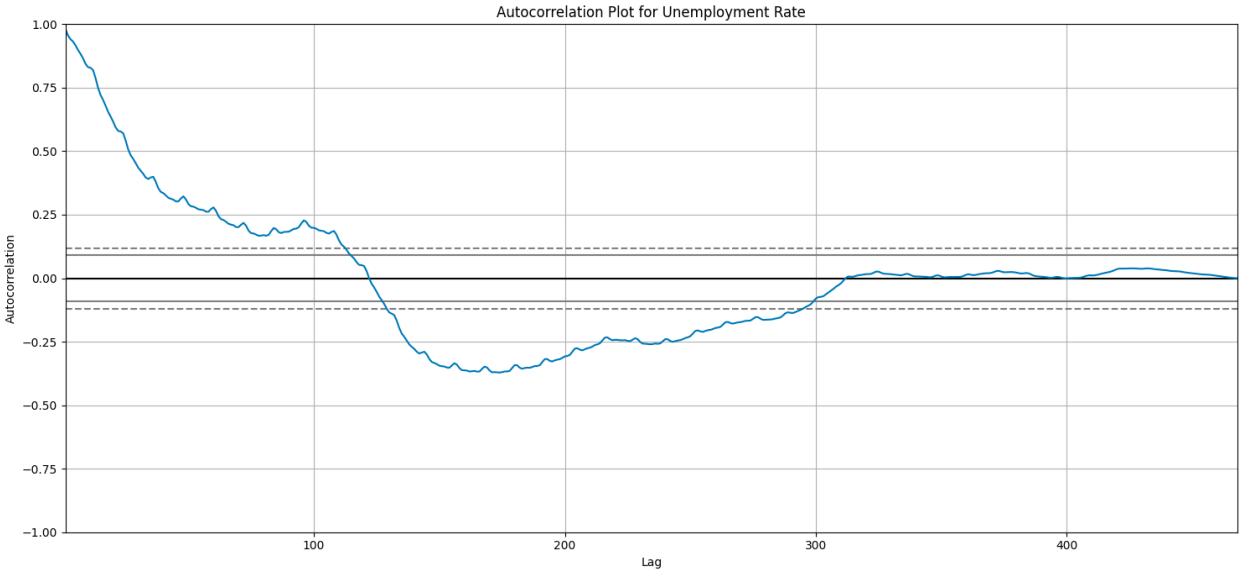

Figure 2.5

The autocorrelation plot illustrates the dataset's underlying correlation structure. Starting from lag 0, there is a significant positive relationship, indicating that the data points are closely tied from one moment to the

next. As we move further to the right, the autocorrelation experiences a sharp decline and then stabilises around the midpoint of the graph. This stabilisation reveals that as time progresses, the influence of past data on future data diminishes.
An observation is a wavy pattern observed around the middle and extending towards the right side of the plot. This oscillation could suggest periodic fluctuations in the unemployment rate over extended durations. Moreover, after a phase where the autocorrelation values become negative, there's a slight uptick towards 0, implying recurring patterns of the unemployment rate that happen in the short term. This information is vital for understanding the cyclical nature of unemployment over time.

3. Methodology and forecasting results

3.1. ARIMA model

Before fitting the ARIMA model, also known as Box-Jenkins method, it is necessary to visualise the sample Autocorrelation function (ACF) and Partial Autocorrelation function (PACF) plots to examine the stationarity since it requires weak stationarity that the mean, variance and covariance functions do not vary over time. As can seen in Figure 3.1 and 3.2, the sample ACF and PACF of the time series doesn't cut off or die down quickly, indicating the non-stationarity feature of the unemployment dataset.

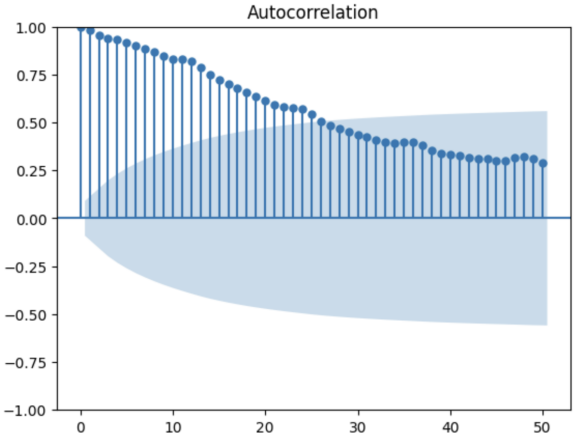

Figure 3.1

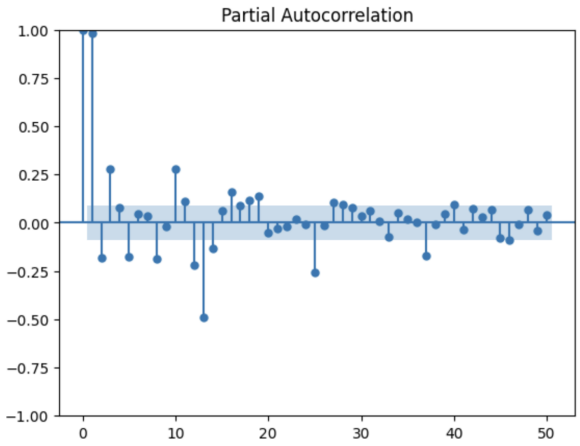

Figure 3.2

The Dickey-Fuller test also suggests the same result, the low p-value (<0.05) suggests that we fail to reject the null hypothesis, and thereby the series is not stationary. Similarly, the test statistic, −1.796153, is greater than all the critical values, 10%, 5%, and 1%, meaning we cannot reject the null hypothesis at any of the usual significance levels. As shown in Figure 3.3, the mean is fluctuating while the standard deviation stays stable. This outcome signifies the importance of transforming the data before fitting ARIMA models.

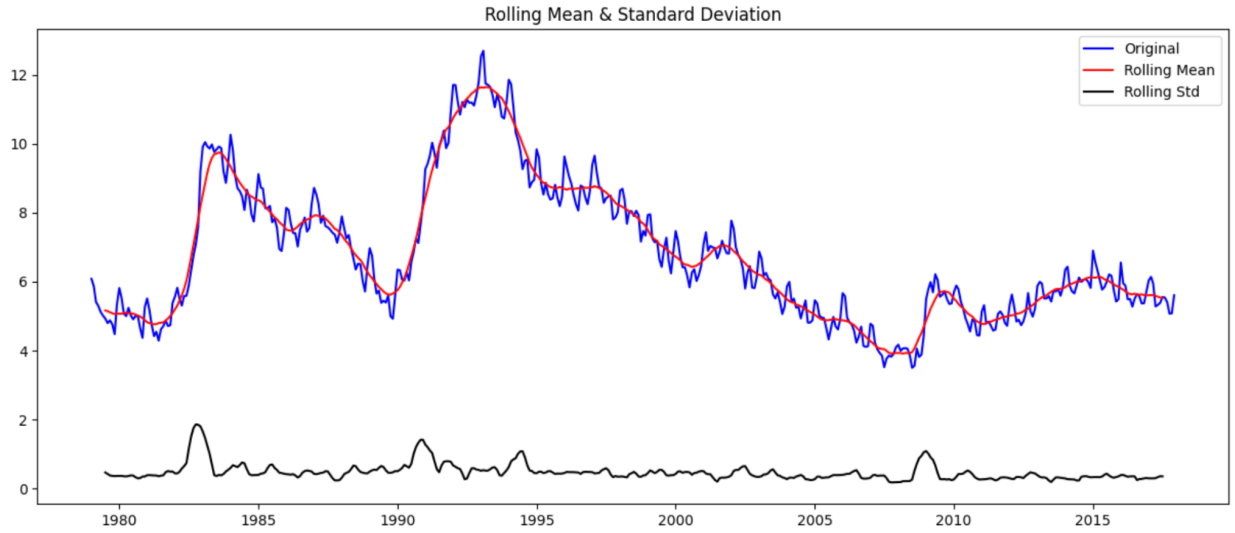

Figure 3.3

Consequently, we applied a differencing method to achieve stationarity. By subtracting the value of the series at time t and t-1, the mean of the time series will be stabilised and the trend and seasonality will be smoothed or eliminated. After differencing the data, as shown in Figure 3.3, the overall trend seems to be removed quite nicely and the series fluctuates at around the mean of zero. Furthermore, the variance is comparably more constant over time, suggesting a certain level of homoscedasticity. Visually speaking, the time series appears to be much more stationary than its raw counterpart.

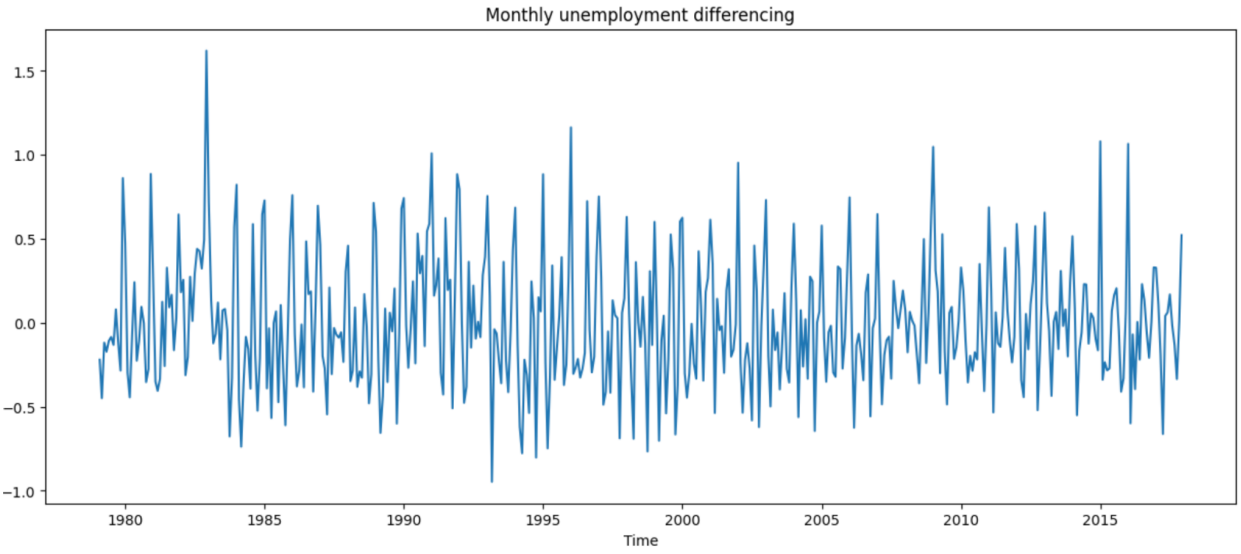

Figure 3.4

The Dickey-Fuller Test implies the identical narrative. The test statistic, −5.9149, is more negative than all the critical values at the 1%, 5%, and 10% significance levels, rejecting the null hypothesis and concluding that the time series is stationary. This successful transformation is more evident in Figure 2.4, with the mean and standard deviation varying slightly over time. Now, it is safe to say that the dataset is prepared to fit the ARIMA models.

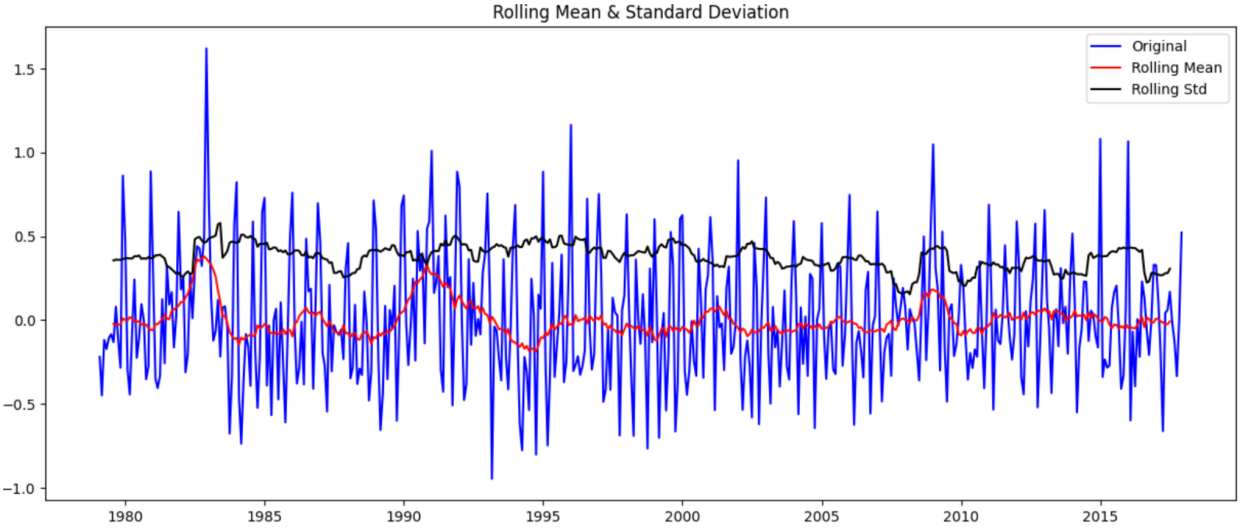

Figure 3.5

AutoRegressive Integrated Moving Average (ARIMA) model is the integration of both the AR(AutoRegressive) and the MA(Moving Average) models, taking the number of differences required to make the time series stationary into consideration. The AR(p) component indicates that the evolving variable of interest is regressed on its own lagged values, whereas the MA(q) component delineates the fact that the regression error is a linear combination of error terms whose values occurred contemporaneously and at various times in the past.

To minimise the Akaike Information Criterion (AIC) value, we performed a grid search across various p and q values for the AR(p), MA(q), and ARIMA(p,q) models. The best results are as follows (Including their forecast errors):

Table 3.1

Model	**AIC Value**	**RMSE Value (24-step ahead)**
ARIMA(10,1,0)	299.1693	0.5679
ARIMA(0,1,8)	298.3652	0.4275
ARIMA(5, 1, 2)	248.6163	0.4955

Figure 3.6

To ensure the ARIMA(5,1,2) yields the best RMSE result, we re-examined the ACF and PACF plot of the differenced data and identified significant seasonal spikes. Consequently, we applied seasonal differencing to mitigate seasonal factors. Despite this adjustment, seasonality remained evident within the data, prompting us to employ a Seasonal ARIMA model for more precise modelling.

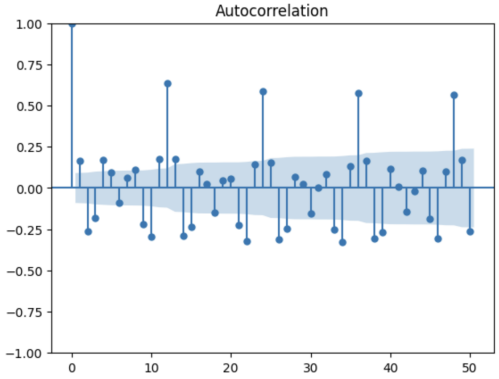

Figure 3.7

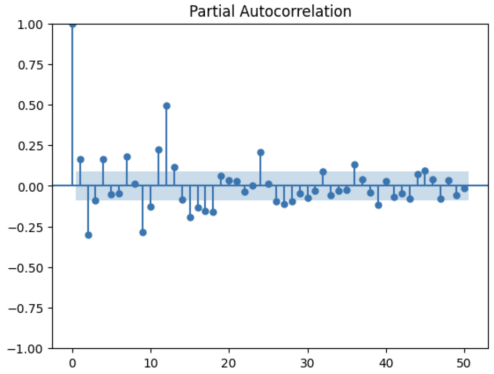

Figure 3.8

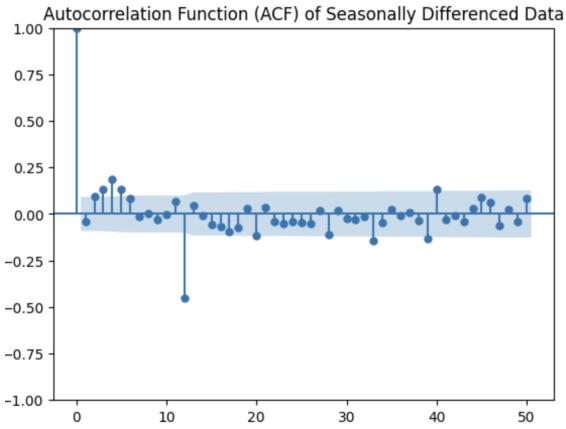

Figure 3.9

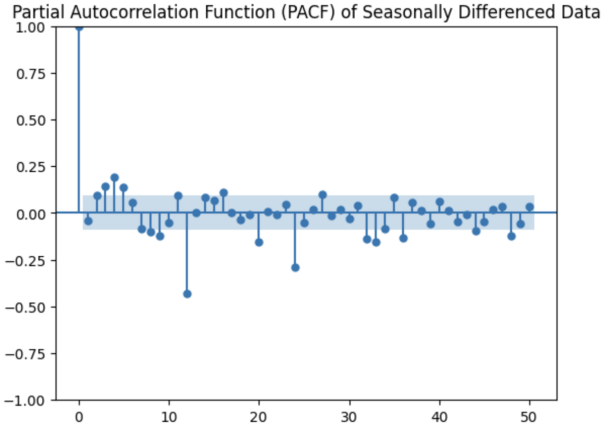

Figure 3.10

The negative autocorrelation terms suggest a preference for using MA terms. At the non-seasonal level, both the sample ACF and PACF exhibit pronounced spikes at lags 4, 5, and 6, with a noticeable cutoff beyond lag 6. In contrast, at the seasonal level, the ACF shows a singular spike at lag 12 before it cuts off. Following these observations and after evaluating the forecasting error using out-of-sample data, the best model was determined to be SARIMAX(5,1,1)(0,1,0,12), which achieved a RMSE of 0.2609. This finding implies that, though we theoretically favour such SARIMAX(0,1,4)(0,1,1,12) incorporating seasonal MA terms, the optimal fit ultimately hinges on the forecasting performance with the help of the out-of-sample data.

Figure 3.11

3.2. TCES

The idea behind Exponential Smoothing is that the forecasts produced using exponential smoothing methods are weighted averages of past observations, with the weights decaying exponentially as the observations get older. In other words, the more recent the observation the higher the associated weight.
The simplest of the exponential smoothing methods is naturally called simple exponential smoothing (SES). This method is suitable for forecasting data with no clear trend or seasonal pattern. The level is the only component of this model, and the corresponding smoothing parameter is α. We can also introduce the trend component bt, the corresponding smoothing parameter is β. Then we got the Trend corrected exponential smoothing method (TCES). We can tune the parameters α and β* to optimise the model.

Furthermore, we can introduce the seasonal component st, with the corresponding smoothing parameters α and β and γ. We got the Holt-Winters' additive method (HWA) and the Holt-Winters' multiplicative method (HWM). The same as the TCES. We can tune the parameters alpha and beta to optimise the model.
We tested all kinds of Exponential smoothing models, and finally, we've chosen the TCES. Because this model got the lowest RMSE.
Holt extended simple exponential smoothing to allow the forecasting of data with a trend. This method involves a forecast equation and two smoothing equations (one for the level and one for the trend):

$$\begin{aligned} \text{Forecast equation} \quad & \hat{y}_{t+h|t} = \ell_t + hb_t \\ \text{Level equation} \quad & \ell_t = \alpha y_t + (1-\alpha)(\ell_{t-1} + b_{t-1}) \\ \text{Trend equation} \quad & b_t = \beta^*(\ell_t - \ell_{t-1}) + (1-\beta^*)b_{t-1}, \end{aligned}$$

Equation 3.1

where ℓt denotes an estimate of the level of the series at time t, bt denotes an estimate of the trend (slope) of the series at time t, α is the smoothing parameter for the level, 0≤α≤1, and β is the smoothing parameter for the trend, 0≤β≤1.

As with simple exponential smoothing, the level equation here shows that it is a weighted average of observation yt and the one-step-ahead training forecast for time t, here given by lt−1 + bt−1. The trend equation shows that bt is a weighted average of the estimated trend at time t based on lt − lt-1 and bt−1, the previous estimate of the trend.

All the soothing models follow these two steps:

1)fit the model manually or using an imported function,

2)tune the smoothing parameter α or β or γ.

Since we need to tune the smoothing parameters α and β. We use the Holt function from the statsmodels library.

We tune the smoothing parameter using a method called grid search. The idea behind this is just to find the best values for alpha and beta by systematically trying different combinations and selecting the ones that result in the lowest error.

We can do a grid search over a range of possible alpha values to select the alpha that produces the smallest RMSE between the forecast values produced by Holt and the corresponding true data. The standard process to perform grid search for α and β is as follows:

Step 1: Define an array of N · N possible values of α: 0≤α1,...,αN≤10≤α1, ..., αN≤1, so as β,

Step 2: For each combination of α and β, obtain the corresponding forecast using holt ()

Then we compute the corresponding RMSE

Step 3: Select the smoothing weight α*, β* that minimises the RMSE.

In practice,

1. We start by defining a range of values for alpha and beta that you want to explore. These values range between 0 and 1.

2. Use loops to iterate through all combinations of alpha and beta values from your predefined range.

For each combination of alpha and beta, fit Holt's linear method model to the training data. You'll get a set of forecasts for the validation set. We take the range from 0.01 to 1, and the stepsize is 0.01. So, there should be nearly 10000 times calculation.

Evaluate the performance of the model for each combination using a suitable metric. Choose the metric that is most appropriate for your specific forecasting problem. In this case, Root Mean Squared Error (RMSE) is:

```
Best Alpha: 0.08
Best Beta: 0.060000000000000005
Best RMSE: 0.25595288925476684
```

Figure 3.12

we can see the best alpha is 0.08, the best beta is 0.06 and the best RMSE is 0.2559.

We randomly tried several alphas and betas using pandas, and no one was better than 0.2559.

We also tried to implement TCES manually. However, we found that using the manual method 0.08 is not the best alpha. As long as we keep increasing the alpha, the RMSE will keep decreasing.

There might be some assumptions that are not met. So we stay with the statsmodels.

From the result we can see that the smoothing parameter is nearly zero and far from one. It tells us the forecasts should not heavily rely on recent observations. The older numbers still have their power.

For the other smoothing methods:

SES	0.441
HW-add	1.265
HW-mul	0.428

Figure 3.13

The best RMSE of SES is 0.441, and the best RMSE of HWA/HWH is 0.428. So we chose the TCES model.

3.3. Neutral Network Models

Artificial neural networks used in this project, including standard Neutral Networks (NNs) and Recurrent Neural Networks (RNNs). In NNs, data flows in one direction without loops, while RNNs have a recurrent connection, allowing information to flow in loops within the network. RNNs can capture complex patterns and seasonality in time series data. They can learn and remember long-term dependencies and use this information to make predictions. Therefore, RNNs should theoretically perform better while analysing time-series data. (*What Are Recurrent Neural Networks? | IBM*, n.d.)

Furthermore, RNNs include standard RNNs and many variances, such as Long Short-Term Memory (LSTM) networks and Gated Recurrent Unit (GRU) networks. LSTMs incorporate a more complex architecture with gating mechanisms that allow them to capture long-term dependencies effectively. The unemployment rate data exhibits complex, long-term dependencies and patterns that extend over several years, LSTMs are more suitable due to their ability to capture such dependencies. Dynamic forecasting enables the model to adapt to changing patterns or seasonality in the data. The model's predictions can influence future predictions, allowing it to react to shifts in the data distribution. Therefore, Dynamic forecasting is built based on LSTM so that it can provide a more realistic view of how your model would perform in practice. (Mittal, 2022)

3.3.1. Model selection

Based on the above analysis, standard NN and LSTM are used to make predictions of the unemployment rate. In the beginning, models with similar hyperparameters were built both in NN and LSTM. The one that produces a lower RMSE will be selected for further hyperparameter tuning. Defines time window as 12, which is used for creating sequences of data, and then scales the data within the range [0, 1]. In this stage, the hyperparameter is set as follows:

Number of layers and neurons: 1 layer with 10 neurons.

In this stage, try a simple model to select one type of model for further exploration in the following stage.

Activation function: Rectified Linear Unit (ReLU)

Common activation functions include ReLU (Rectified Linear Unit), Sigmoid, and Hyperbolic Tangent (Tanh). Sigmoid activation functions are commonly used in the output layer of binary classification problems, while Tanh is especially used when outputs are centred around zero. Therefore, ReLU, which is defined as f(x) = max (0, x), is chosen for this stage.

Shuffling: False

Time series models rely on the sequential patterns within the data. Shuffling the data disrupts these patterns and can lead to a loss of information, so ‘shuffle = False’ is set while fitting the model.

Batch size: 5

Larger batch sizes can lead to faster training because they utilise parallelism more effectively, but the training process might be stuck in local minima. Smaller batch sizes can introduce more noise in the training process and escape local minima to reach global minima, so it offers better convergence and stability. Therefore, a smaller batch size is set.

Optimizer: Adam (Adaptive Moment Estimation)

Optimizer includes Momentum-based Optimizers and Adaptive Learning Rate Optimizers. Momentum-based Optimizers (e.g., Momentum, Nesterov Accelerated Gradient) incorporate momentum to accelerate convergence and reduce oscillations. However, they require more tuning of hyperparameters, and the performance is sensitive to the choice of hyperparameters, and the optimal value may vary across different tasks.

Adaptive Learning Rate Optimizers (e.g., RMSprop, Adam, Adagrad) automatically adjust the learning rates for each parameter based on the historical behaviour of the gradients. They adapt the learning rate to the specific requirements of each parameter during training.

Adam combines the benefits of both AdaGrad and RMSprop, and it helps to solve the initial bias in the moving averages of the gradients. (Zhang, 2021)

Next, NN and LSTM go through the training data 3000 times and then make predictions using validation data. RMSE is calculated to measure the model performance.

Table 3.2

	Training RMSE	Validation RMSE
NN	1.1169	0.1602
LSTM	1.1861	0.2321
LSTM-dynamic		0.2633

According to the above results, training RMSE for both models is higher than Validation RMSE, which means the models are not overfitting. NN has a lower RMSE, so it has better performance compared to LSTM. LSTMs might not be able to leverage their full potential in this dataset, which is not complex enough to show the effectiveness of high-complexity models like LSTM. In machine learning, complexity is not always advantageous. Simpler models are preferred in situations where the problem is not inherently complex. In this way, standard NN is chosen for the next stage.

3.3.2. Hyperparameter tuning

Try 1, 2, 3 hidden layers and 10, 20, 30 neurons on each stage for standard NN models, when keeping other hyperparameters the same as the above stage.

The following table shows the results of this process:

Table 3.3 RMSE with different number of layers and neurons

Neurons/Layers	**1**	**2**	**3**
10	0.1807	0.2067	0.2440
20	0.2347	0.2834	0.1716
30	0.2293	0.2030	0.3475

The results show that the NN model that has 3 hidden layers with 20 neurons on each layer has the best performance. Next, perform hyperparameter tuning to improve the performance of the 3-layer standard NN

model with 20 neurons on each layer. Define a function to suggest hyperparameter by using Optuna. Specifically, suggests hyperparameters as followed:

- learning_rate is a floating-point value between 0.01 and 0.1.
- batch_size is an integer between 5 and 30.
- num_epochs is an integer between 50 and 500.

After that, create an Optuna study with a specified direction to minimise the RMSE, and get the best hyperparameters.

3.3.3. Prediction

The best model is built using the best parameters from the above stages. It makes predictions based on out-of-sample data and calculates the RMSE to measure the final performance. RMSE for the final prediction is 0.1728.

The following chart shows the out-of-sample forecast performs well.

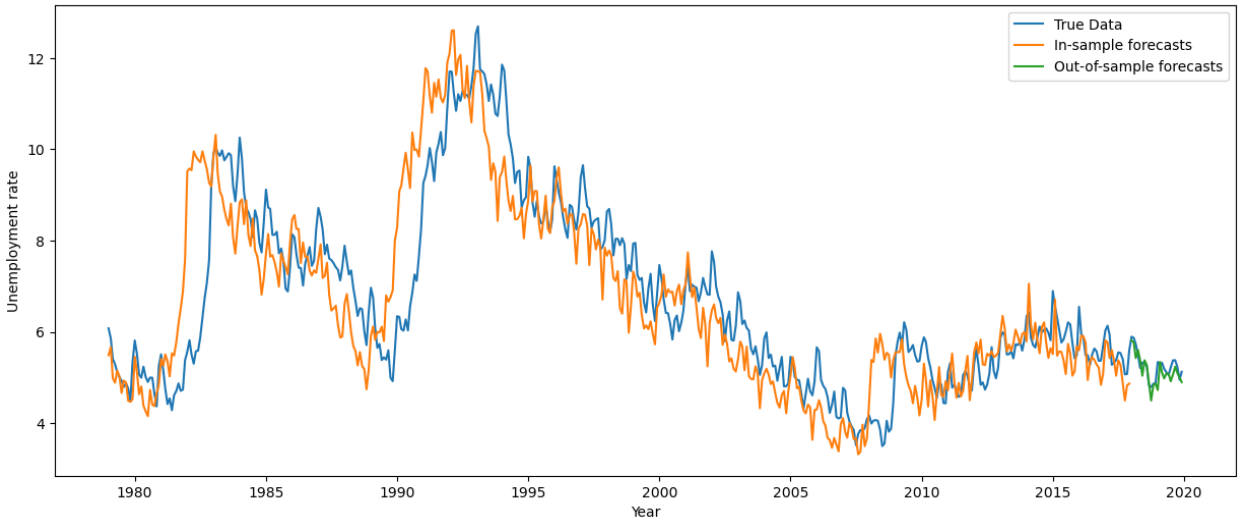

Figure 3.14

4. Final analysis, conclusion, limitations and future steps

In conclusion, the Seasonal ARIMA model performs well in identifying and predicting patterns, particularly in the presence of strong seasonality within our dataset. Nonetheless, our findings indicate a misalignment between the optimal parameters and the theoretical ACF and PACF plot analysis. Moreover, this model does not yield the most accurate forecasts compared to the other models we assessed.

Trend Corrected Exponential Smoothing is our next selected model that includes trend adjustments, making it potentially valuable for predicting the unemployment rate. Evaluated by RMSE, the performance of TCES is not optimal. The limitation of this model is that it fails to incorporate the pronounced seasonality, and when facing unforeseen impact, despite the consideration of trend, the model has more pivotal factors to anticipate, such as the geopolitical events and policy changes.

The Neural Network (NN) model showcased the best results, particularly in forecasting well to future data, as evidenced by its low RMSE. This indicates that NNs have the capacity to model the complex nature of the unemployment rate effectively. However, it tends to require more computational power for training which would put much pressure on executing the code. Looking ahead, the future improvement of the NN

model should concentrate on improving both performance and code efficiency to achieve more accurate unemployment rate forecasts.

Reference

What are Recurrent Neural Networks? | IBM. (n.d.). https://www.ibm.com/topics/recurrent-neural-networks

Mittal, A. (2022, August 11). Understanding RNN and LSTM - Aditi Mittal - Medium. *Medium*. https://aditi-mittal.medium.com/understanding-rnn-and-lstm-f7cdf6dfc14e

Zhang, J. (2021, December 13). Optimisation Algorithm — Adaptive Moment Estimation (ADAM). *Medium*. https://towardsdatascience.com/optimisation-algorithm-adaptive-moment-estimation-adam-92144d75e232

Appendix

```
15/15 [==============================] - 0s 401us/step
Layers: 1, Neurons: 10
Test Data RMSE original scale: 0.1807
15/15 [==============================] - 0s 410us/step
Layers: 1, Neurons: 20
Test Data RMSE original scale: 0.2347
15/15 [==============================] - 0s 409us/step
Layers: 1, Neurons: 30
Test Data RMSE original scale: 0.2293
15/15 [==============================] - 0s 408us/step
Layers: 2, Neurons: 10
Test Data RMSE original scale: 0.2067
15/15 [==============================] - 0s 353us/step
Layers: 2, Neurons: 20
Test Data RMSE original scale: 0.2834
15/15 [==============================] - 0s 381us/step
Layers: 2, Neurons: 30
Test Data RMSE original scale: 0.2030
15/15 [==============================] - 0s 364us/step
Layers: 3, Neurons: 10
Test Data RMSE original scale: 0.2440
15/15 [==============================] - 0s 378us/step
Layers: 3, Neurons: 20
Test Data RMSE original scale: 0.1716
15/15 [==============================] - 0s 352us/step
Layers: 3, Neurons: 30
Test Data RMSE original scale: 0.3475
```