Pandemic Prediction Using Graph Neural Networks

Royston Rex Fernandez

A DISSERTATION

Submitted to

The University of Liverpool

in partial fulfilment of the requirements
for the degree of
MASTER OF SCIENCE

Student Declaration

I confirm that I have read and understood the University's Academic Integrity Policy.

I confirm that I have acted honestly, ethically and professionally in conduct leading to assessment for the programme of study.

I confirm that I have not copied material from another source nor committed plagiarism nor fabricated data when completing the attached piece of work. I confirm that I have not previously presented the work or part thereof for assessment for another University of Liverpool module. I confirm that I have not copied material from another source, nor colluded with any other student in the preparation and production of this work.

I confirm that I have not incorporated into this assignment material that has been submitted by me or any other person in support of a successful application for a degree of this or any other university or degree-awarding body.

SIGNATURE [illegible]
DATE September 24, 2024

Acknowledgements

I would like to express my heartfelt gratitude to my supervisors, Dr Simona Capponi and Ms Estelle Varloot, for their constant support, valuable insights, and expert guidance throughout this project. Their advice and encouragement have been instrumental in shaping the direction and successful completion of this dissertation.

I am also thankful to my academic mentors and fellow students at the University of Liverpool for their helpful discussions and shared knowledge, which significantly enhanced my learning. Special recognition goes to the technical team for their assistance with the computational resources that were vital to the project's success.

I am deeply grateful to my family and friends for their unwavering support, patience, and encouragement during the lengthy hours of research and writing.

Lastly, I would like to express my gratitude to everyone who contributed, both directly and indirectly, to the completion of this dissertation. Your support made this work possible.

Pandemic Prediction Using Graph Neural Networks

Abstract

This dissertation explores the use of Graph Neural Networks (GNNs) to model the spread of COVID-19, integrating human mobility and epidemiological data to predict virus transmission across regions. By combining GNNs with mobility and epidemiological data, this research provides a more robust and scalable method for predicting virus transmission. Using mobility data and demographic factors, a GNN model was developed and trained using PyTorch Geometric, with a final Mean Absolute Error (MAE) of 0.27, demonstrating its effectiveness in this context. The model's ability to predict virus spread offers a valuable tool for public health officials to track and mitigate the transmission of infectious diseases. This research provides a foundation for broader applications in other regions and disease contexts, highlighting the relevance of advanced graph-based modelling techniques in epidemiology.

Statement of Ethical Compliance

This project falls under Category B0: Anonymous data about humans from open sources and no use of human participants in any activity. I confirm that I have adhered to the ethical guidelines provided by the University, ensuring that all data presented are genuine, properly sourced, and used in compliance with GDPR and the University's Policy on Research Ethics.

Contents

1 Introduction **1**
1.1 Scope . . . 1
1.2 Problem Statement . . . 2
1.3 Approach . . . 2
1.4 Outcome . . . 3

2 Background **4**

3 Design **6**
3.1 Original design . . . 6
3.2 Refined Design and Methodology . . . 6

4 Implementation **9**
4.1 The Dataset . . . 9
4.2 Data Pre-processing . . . 9
4.3 Exploratory Data Analysis . . . 11
4.3.1 Time Series Analysis . . . 11
4.3.2 Feature Correlation . . . 12
4.4 Graph Structures . . . 13
4.4.1 Components of a Graph . . . 13
4.4.2 PyTorch Geometric . . . 14
4.5 Graph Neural Network . . . 14
4.5.1 Model Architecture . . . 15
4.5.2 Model Training and Validation . . . 17

5 Testing and Evaluation **19**

6 Conclusion **21**
6.1 Future Work . . . 21

7 BCS Project Criteria & Self-Reflection **22**
7.1 Self Evaluation . . . 22

A Code **26**
A.1 Model Architecture . . . 26
A.2 Essential Libraries . . . 27

B Original design document **29**
B.1 Project Description . . . 29
B.2 Aims and Requirements . . . 29
B.2.1 Essential Requirements . . . 30
B.3 Development Summary . . . 30

B.4 Data Sources . . . 31
B.5 Testing and Evaluation . . . 31
B.6 Project Ethics and Human Participants . . . 31
B.7 BCS project criteria . . . 31
B.8 UI/UX Mock-up . . . 32
B.9 Project Plan . . . 32
B.10 Risks & Contingency Plans . . . 33

List of Figures

3.1 System Architecture Diagram for GNN Model Pipeline 7

4.1 Infection dataset after preprocessing 10
4.2 Mobility dataset after preprocessing 10
4.3 Descriptive statistics of infection dataset 11
4.4 Descriptive statistics of mobility dataset 11
4.5 Trend of infection and death rates over time 12
4.6 Feature correlation matrix 13
4.7 Graph Structure 15
4.8 Model Learning Curve 18

5.1 Final Model Prediction 19

B.1 Gantt Chart 33

List of Tables

B.1 The risks and contingencies associated with the project 33

Chapter 1

Introduction

This project aims to develop a Graph Neural Network (GNN) model to predict the pandemic infection rates, particularly COVID-19, in the counties of the United States (US). This involves integrating comprehensive infection rate data with human mobility data to form graph structures for a given day, and then feed this information to the GNN model for prediction. The GNN model will be designed to adapt to changing infection patterns over time and seamlessly incorporate newly generated data. Additionally, the project will produce intuitive visualizations, as well as detailed reports on the data being analysed in order to predicted infection trends and highlight potential outbreak.

1.1 Scope

This project is designed to be a versatile and scalable framework for predicting pandemic infection rates using GNNs. While the current focus is on predicting COVID-19 infection rates across the counties of the United States, the methodology and model architecture are intended to be adaptable to a wide range of infectious diseases and geographic regions. The scope of this project includes:

1. Geographical Flexibility: Although the initial application of the model is within the United States, the design allows for easy adaptation to other countries or regions, provided that similar data—such as infection rates and human mobility—is available.

2. Applicability to Other Epidemics: The GNN model is not limited to COVID-19; it can be applied to any epidemic or pandemic situation where infection data and mobility patterns are relevant. This makes the project highly relevant for future public health emergencies.

3. Data Requirements: The project relies on comprehensive datasets, including daily COVID-19 infection rates and human mobility data, which will be processed into graph structures. The availability and quality of such data in other regions or future scenarios could impact the model's accuracy, but the underlying framework remains robust.

4. Temporal Adaptation: The GNN model will be designed to accommodate new data over time, allowing it to update predictions as new infection patterns emerge. This ensures that the model remains relevant throughout the course of an epidemic or pandemic.

5. Output and Deliverables: The project will produce not only a functional GNN model but also detailed visualisations, reports, and documentation that outline the research methodology, model performance, and potential areas for further improvement.

The final deliverable will be a comprehensive tool that can be used by public health officials and researchers to predict infection trends and respond to potential outbreaks proactively.

1.2 Problem Statement

The COVID-19 pandemic has highlighted the critical need for accurate, real-time predictions of infection rates to inform public health responses and mitigate the spread of the virus (Kapoor et al. 2020). Traditional epidemiological models often struggle to incorporate complex, dynamic data such as human mobility patterns, which are crucial for understanding and predicting the spread of infectious diseases. Additionally, these models may not easily adapt to rapidly changing infection patterns, which can result in outdated or inaccurate predictions.

Given the highly interconnected nature of counties within the United States, where population movement between regions significantly impacts infection rates, there is a pressing need for a more sophisticated predictive model (Arenas et al. 2020). This model must not only leverage diverse data sources, such as daily infection rates and human mobility data, but also adapt to temporal changes and incorporate newly generated data seamlessly.

The core problem this project seeks to address is the development of a predictive tool capable of accurately forecasting COVID-19 infection rates across different counties in the United States. The primary challenges include:

1. Data Integration: Effectively combining infection rate data with human mobility data to create meaningful graph structures that represent the spread of the virus.

2. Model Adaptability: Designing GNN that can adapt to changing infection patterns over time and maintain prediction accuracy as new data becomes available.

3. Scalability: Ensuring the model can be generalised and applied to other regions or pandemics, potentially extending its utility beyond the scope of COVID-19.

4. Visualisation and Reporting: Creating intuitive visualisations and detailed reports that can help stakeholders, such as public health officials, understand and act on the predicted trends.

1.3 Approach

The project begins with data acquisition and preprocessing, focusing on collecting essential datasets, including COVID-19 infection rates and human mobility data from reliable sources. These datasets will be integrated into a dynamic graph structure where counties are represented as nodes, and the mobility patterns between them form the edges. This graph will capture the spatial relationships between different regions, allowing the model to learn how movement between counties influences the spread of the virus. Preprocessing steps such as data cleaning, normalisation, and transformation will ensure the data is properly formatted and ready for input into the model.

The core of the project involves designing and developing a GNN tailored to this task. The model will be specifically structured to capture both spatial and temporal dependencies within the data. The GNN will be trained using cross-validation techniques and optimised through hyperparameter tuning to ensure it performs well across various scenarios. Regularisation methods will be employed to prevent overfitting and to maintain robustness when applied to new data.

Evaluation and validation are crucial steps in this approach. The model's performance will be assessed using metrics such as Mean Squared Error (MSE) and Mean Absolute Error (MAE), which will provide a clear understanding of its predictive accuracy. Validation techniques, including time series cross-validation and scenario analysis, will be used to test the model's ability to generalise to unseen data and to adapt to different conditions. This rigorous evaluation process will ensure that the model is not only accurate but also reliable in various real-world applications.

Finally, the project is designed with future work and scalability in mind. The model's architecture and methodology are flexible enough to be adapted to other regions or infectious diseases beyond COVID-19. This scalability ensures that the project can have a lasting impact, providing a valuable tool for public health responses in future pandemics. Continuous improvement will also be emphasised, with the model designed to incorporate new data and advanced techniques as they become available, ensuring its ongoing relevance and effectiveness.

1.4 Outcome

The outcome of this project will be a sophisticated GNN model capable of accurately predicting COVID-19 infection rates across U.S. counties. This model will not only provide precise forecasts but also adapt dynamically to new data, maintaining its relevance over time. The project will result in a powerful tool for public health officials, enabling them to visualise infection trends, identify potential hotspots, and make informed decisions to mitigate the spread of the virus. Additionally, the comprehensive documentation, visualisations, and reports generated will facilitate the model's adoption and adaptation for other regions or future pandemics, making a significant contribution to public health management and preparedness.

Chapter 2

Background

The COVID-19 pandemic has spurred significant research into the role of population movement and mobility patterns in the transmission of the virus. GNNs have emerged as powerful tools for modelling these complex dynamics, as demonstrated in several key studies.

Panagopoulos et al. in 2021 explored the impact of population movement on the spread of COVID-19 by using GNNs, where regions within a country were represented as nodes and the mobility data as edge weights (Panagopoulos, Nikolentzos, and Vazirgiannis 2021). This approach allowed them to model the spatial interactions between different regions effectively, highlighting how movement patterns could influence the spread of the virus.

Building on this concept, Kapoor et al. in 2020 employed a forecasting approach using GNNs to distinguish their model from traditional time series approaches (Kapoor et al. 2020). By leveraging a large-scale spatio-temporal graph, they were able to incorporate both spatial and temporal information, which was crucial for accurately modelling the complex dynamics of COVID-19 spread at the U.S. county level. Their findings underscored the effectiveness of combining mobility data with graph-based deep learning techniques to enhance our understanding of the virus's transmission and evolution.

Son et al. in 2022 introduced the Temporal Multiresolution Graph Neural Networks (TMGNN) framework, which advances the field by learning to construct multiscale and multi-resolution graph structures while integrating time-series signals (Son Hy et al. 2022). This approach captures temporal changes in dynamic graphs, allowing for the extraction of both local and global information. Their research emphasised the importance of these multiscale structures in comprehensively understanding the dynamics of a global pandemic, providing insights into how different levels of granularity can impact predictive accuracy.

In addition to the methodological advancements in GNNs, Kang et al. in 2020 contributed a valuable multiscale dynamic human mobility flow dataset, tracking the movement of millions of anonymous mobile phone users across the United States during the pandemic (Kang et al. 2020). This dataset, available at various geographic levels, demonstrated a high correlation with other data sources, reinforcing its reliability. It has been instrumental in monitoring epidemic spread, informing public health policies, and analysing changes in human behaviour during the crisis.

Arenas et al. in 2020 developed an age-stratified, mobility-based meta-population model to address this crisis. Their model incorporates key elements of COVID-19 transmission, including the interaction between individuals, the impact on different demographic groups, and human mobility patterns (Arenas et al. 2020). By employing a microscopic Markov chain approach, they formalised the epidemic dynamics and evaluated the effects of non-pharmacological interventions like social distancing and confinement. Their model successfully forecasted the incidence and fatalities across municipalities in Spain, providing valuable insights for policymakers on the effectiveness of containment measures. This approach highlights the importance of integrating demographic and mobility data to accurately track and manage the epidemic's progression.

Further expanding on the application of GNNs in dynamic settings, Gravina et al. in 2024 conducted a comprehensive survey of recent advancements in deep learning for dynamic graphs (Gravina and Bacciu 2024). Their work highlights the maturation of deep graph networks (DGNs) and the ongoing challenges in applying these models to real-world systems that evolve over time. They provide a thorough overview of current state-of-the-art techniques for learning temporal and spatial information in dynamic graphs, establishing a baseline for evaluating new architectures. They also emphasise the importance of extending techniques developed for static graphs to the temporal domain, addressing challenges such as over-smoothing, over-squashing, and robustness to adversarial attacks. Their work serves as a critical reference point for future research in dynamic graph representation learning, particularly in the context of pandemics, where understanding the evolution of interconnected entities over time is crucial.

These studies collectively highlight the potential of GNNs and mobility data in modelling and predicting the spread of infectious diseases like COVID-19. They provide a strong foundation for further exploration and application of graph-based techniques in public health, particularly in the context of dynamic and complex pandemics.

Chapter 3

Design

This section outlines the evolution of the design, highlighting key differences between the initial and updated designs and detailing the new design approach and methodologies.

3.1 Original design

The original design document, as detailed in Appendix B, has undergone minor revisions while maintaining the project's core principles. The updated project description now includes specific details regarding the model's design and methodology, whereas the previous version provided a more general overview of the project's goals and deliverables. In terms of aims and requirements, the revised document offers a deeper focus on the design process and model architecture. The previous document mainly outlined the overall project goals and requirements, without delving into the specifics of the design. The essential requirements section in the updated version presents a more comprehensive account of the design components and methodologies, as opposed to the high-level requirements described earlier. Furthermore, the development summary in this updated document gives a more structured explanation of the development process, including detailed descriptions of the model architecture and training methodologies, which contrasts with the broader overview provided in the original document.

3.2 Refined Design and Methodology

The system designed for this project involves several key components, each contributing to the overall functionality of the pandemic prediction model. The system architecture as seen on figure 3.1 has various components such as:

1. Data Acquisition and Preprocessing: This module handles the collection and initial processing of data from various sources.

2. Graph Construction: This component transforms the processed data into a graph structure suitable for analysis with GNNs. Using PyTorch Geometric, the system constructs graphs where nodes represent counties and edges represent mobility flows between them.

3. Model Architecture: The core of the system is a GCN model built using PyTorch and PyTorch Geometric. The model consists of multiple 'GCNConv' layers, with batch normalization and dropout applied to prevent overfitting

and improve generalisation. The 'GCNConv' layers in the GNN use convolutional operations to aggregate node features from local neighbourhoods.

4. Training and Validation: The system includes a training module to optimise the model's performance and a validation module to assess its accuracy. The training process uses standard techniques and employs gradient descent with the AdamW optimizer, and the walk-forward validation approach to evaluate the model over different time windows.

5. Testing and Evaluation: After training, the model is tested on unseen data to evaluate its predictive power. Performance metrics such as Mean Absolute Error (MAE) and Mean Squared Error (MSE) are calculated to measure the model's accuracy.

6. Visualisation: The final component involves visualising the results and model predictions using libraries like Matplotlib and seaborn. This includes plotting learning curves, prediction vs. actual graphs, and other relevant visual representations. The entire list of libraries used can be found in Appendix A.2

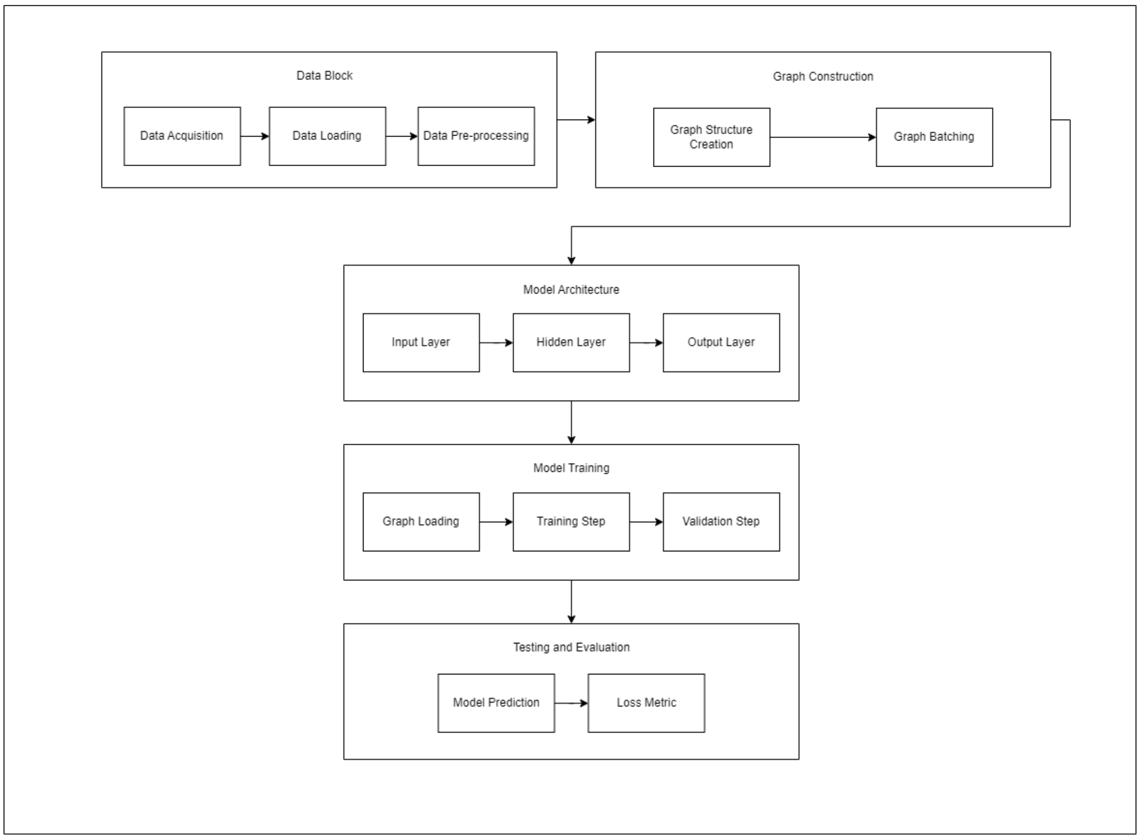


Figure 3.1: System Architecture Diagram for GNN Model Pipeline

Since this project is primarily focused on the development and evaluation of a model rather than user interaction, there is no traditional user interface. However, the system includes an interface for Jupyter notebook. It provides commands for executing data processing, model training, and evaluation scripts within Jupyter Notebook using block execution. This interface allows for the seamless running of scripts and commands directly in Jupyter Notebook cells, facilitating efficient data manipulation, model configuration, training, and performance evaluation. Users can manage workflows through an interactive environment, making adjustments and running different sections of the code with ease to streamline the development and testing process.

The implementation section explains, in detail, the practical steps and processes involved in developing and deploying the GNN model, including data handling, exploratory data analysis, model training, and performance evaluation.

Chapter 4

Implementation

4.1 The Dataset

The model design is built upon two key datasets used to analyse COVID-19 infection trends. The first dataset is the infection data from the COVID-19 Data Repository by the Center for Systems Science and Engineering (CSSE) at Johns Hopkins University (Dong, Du, and Gardner 2020). This repository serves as the source for the 2019 Novel Coronavirus Visual Dashboard, which reports COVID-19 case data aligned with daily updates from the Chinese CDC and the WHO for regions within and outside mainland China. The dataset effectively captures the timing of the first reported cases in new countries or regions. For this project, the 2 files containing time series data on confirmed cases and death rates for U.S. counties were used.

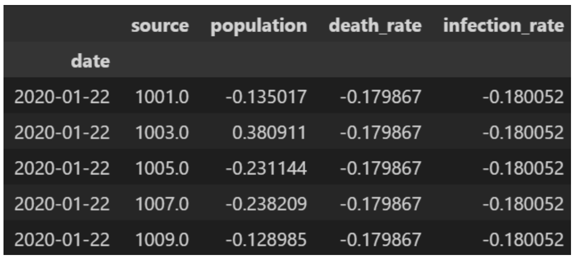
The second dataset is the Multiscale Dynamic Human Mobility Flow Dataset in the U.S. during the COVID-19 Epidemic, provided by the GeoDS Lab at the University of Wisconsin-Madison (Kang et al. 2020). This dataset is based on the analysis of millions of anonymous mobile phone users' visits to various locations, collected by SafeGraph. It provides daily and weekly origin-to-destination (O-D) population flows at the census tract, county, and state levels. The mobility dataset is highly correlated with other publicly available data, affirming its reliability. For this project, the daily mobility flow data between U.S. counties starting from March 1st, 2020, was used.

To manage the large number of mobility data files, a Python script from STAN: Spatio-Temporal Attention Network for Pandemic Prediction was employed to download and merge individual CSV files from the dataset into a single file for analysis (Gao et al. 2021). This streamlined the processing of mobility data, enabling efficient loading and integration with the infection dataset for model training.

4.2 Data Pre-processing

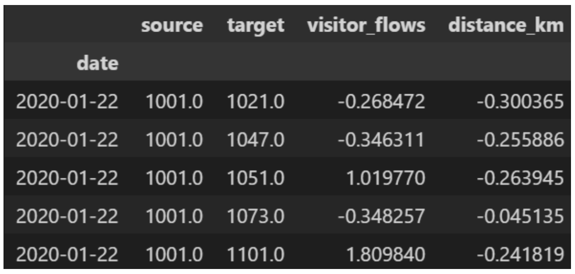
Given the large volume of CSV files in the mobility dataset, Dask was utilized to process these files efficiently within the Jupyter environment. Dask is a Python library designed for parallel and distributed computing, allowing for scalable data handling (Rocklin et al. 2015). By employing Dask, it became possible to process the large dataset in parallel, which significantly improved the performance when loading, merging, and analysing the mobility data without overwhelming memory resources. This approach ensured efficient handling of the numerous files involved.

Several preprocessing steps were applied to the mobility dataset to optimize its usability for the project. Columns containing only identifiers for counties and areas



	source	population	death_rate	infection_rate
date				
2020-01-22	1001.0	-0.135017	-0.179867	-0.180052
2020-01-22	1003.0	0.380911	-0.179867	-0.180052
2020-01-22	1005.0	-0.231144	-0.179867	-0.180052
2020-01-22	1007.0	-0.238209	-0.179867	-0.180052
2020-01-22	1009.0	-0.128985	-0.179867	-0.180052

Figure 4.1: Infection dataset after preprocessing



	source	target	visitor_flows	distance_km
date				
2020-01-22	1001.0	1021.0	-0.268472	-0.300365
2020-01-22	1001.0	1047.0	-0.346311	-0.255886
2020-01-22	1001.0	1051.0	1.019770	-0.263945
2020-01-22	1001.0	1073.0	-0.348257	-0.045135
2020-01-22	1001.0	1101.0	1.809840	-0.241819

Figure 4.2: Mobility dataset after preprocessing

were removed, and relevant columns were renamed to reflect the source and target counties.

An assumption was made regarding visitor flows between counties: only flows greater than 100 visitors were considered significant enough to influence mobility patterns. This threshold was set to enhance both computational efficiency and memory usage. Additionally, intra-county mobility (movement within the same county) was ignored, as the focus of the project was on inter-county regional trends rather than internal county movement.

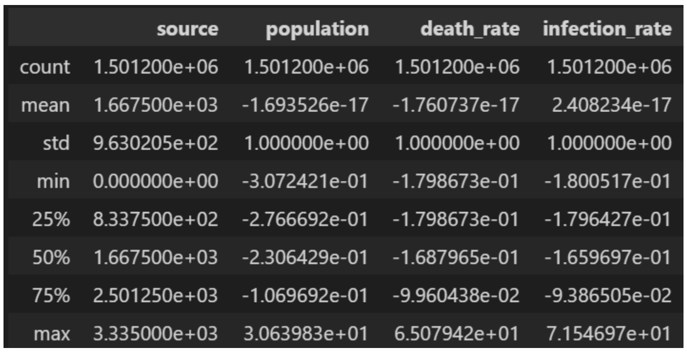
Since the dataset included spatial information (latitude and longitude of counties), the geodesic function from the 'geopy' library was used to calculate the distance between counties in kilometres. This distance information was added to the dataset. The data was further cleaned by removing invalid or irrelevant rows, ensuring that the dataset was prepared for subsequent analysis and modelling.

Unlike the mobility data, the infection data consisted of only two CSV files: one for confirmed cases and one for confirmed deaths, making the use of Dask unnecessary. Instead, Pandas was sufficient for processing the dataset. The infection data was messy and inconsistent, particularly with the date formats. The columns containing dates were cleaned and renamed to follow a standardized date format. As the dataset was in a time series format, with daily infection rates presented in columns, the data needed to be unpivoted. This process transformed the dataset into a standard format, where some columns served as identifier variables (e.g., counties), while others were measured variables (e.g., infection rates). Missing county identifiers were reconstructed from the last five digits of the UID column. Once the data was cleaned, the confirmed cases and death records were merged, and column names were updated accordingly. Figure 4.1 and figure 4.2 shows the two datasets (infection and mobility data) after all the data-preprocessing steps.

To align the counties from the infection data with the source and target columns in the mobility data, a mapping was created by assigning new indices starting from 0 for all counties. This mapping was then applied to the source and target columns in the mobility dataset, as well as to the source column in the infection data. This ensured consistency between the two datasets, enabling accurate graph-based analysis where the infection data and mobility flow could be integrated and referenced by a common county identifier.

4.3 Exploratory Data Analysis

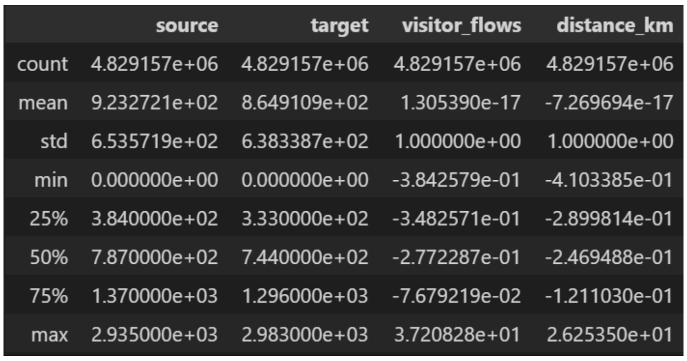
To gain a deeper understanding of the data, Exploratory Data Analysis (EDA) was performed to determine the optimal approach for manipulating the data sources. EDA is crucial for uncovering patterns and identifying anomalies in the dataset (Mukhiya and Ahmed 2020). The 'pandas.describe()' method was used to generate descriptive statistics, providing insights into the distribution, central tendency, and spread of the data. Figure 4.3 and figure 4.4 shows the descriptive statistics of both the datasets. This step helped to assess the dataset's structure and guided decisions for subsequent data cleaning and transformation. Both datasets (mobility and infection) were normalized using the 'StandardScaler' method from the 'scikit-learn' library, which standardizes features by removing the mean and scaling to unit variance. This step is critical, as many machine learning models expect input data to have a distribution with 0 mean and unit variance to perform optimally.



	source	population	death_rate	infection_rate
count	1.501200e+06	1.501200e+06	1.501200e+06	1.501200e+06
mean	1.667500e+03	-1.693526e-17	-1.760737e-17	2.408234e-17
std	9.630205e+02	1.000000e+00	1.000000e+00	1.000000e+00
min	0.000000e+00	-3.072421e-01	-1.798673e-01	-1.800517e-01
25%	8.337500e+02	-2.766692e-01	-1.798673e-01	-1.796427e-01
50%	1.667500e+03	-2.306429e-01	-1.687965e-01	-1.659697e-01
75%	2.501250e+03	-1.069692e-01	-9.960438e-02	-9.386505e-02
max	3.335000e+03	3.063983e+01	6.507942e+01	7.154697e+01

Figure 4.3: Descriptive statistics of infection dataset

Finally, the date columns were set as the index in both datasets to group information by date, a key step in preparing the data for graph creation.



	source	target	visitor_flows	distance_km
count	4.829157e+06	4.829157e+06	4.829157e+06	4.829157e+06
mean	9.232721e+02	8.649109e+02	1.305390e-17	-7.269694e-17
std	6.535719e+02	6.383387e+02	1.000000e+00	1.000000e+00
min	0.000000e+00	0.000000e+00	-3.842579e-01	-4.103385e-01
25%	3.840000e+02	3.330000e+02	-3.482571e-01	-2.899814e-01
50%	7.870000e+02	7.440000e+02	-2.772287e-01	-2.469488e-01
75%	1.370000e+03	1.296000e+03	-7.679219e-02	-1.211030e-01
max	2.935000e+03	2.983000e+03	3.720828e+01	2.625350e+01

Figure 4.4: Descriptive statistics of mobility dataset

4.3.1 Time Series Analysis

To understand the trends in the data, time series analysis was conducted by grouping the datasets by date and plotting the trends over time. The resulting graph, as shown in the figure 4.5, illustrates the gradual rise in COVID-19 infection and death rates. Initially, as the pandemic began, the graph remained relatively flat, but an exponential rise in both infection and death rates was observed over time.

An interesting observation was the decline in the death trend below the infection trend. This crossover occurred around the time when vaccination efforts began in the United States on December 14, 2020 (Thomas, Weiland, and LaFraniere 2020). While this timing suggests that vaccination could be a contributing factor to the decline in death rates, it cannot be considered a strong correlation, as vaccinating a large population would take time to produce a noticeable reduction in death rates. Nonetheless, vaccination remains a plausible candidate to explain this decline.

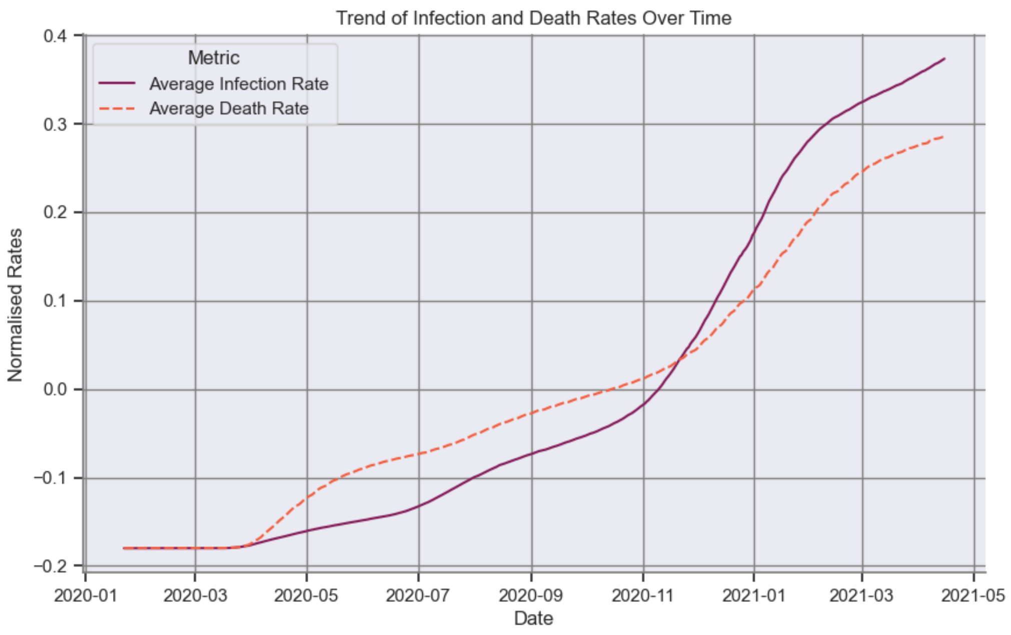


Figure 4.5: Trend of infection and death rates over time

4.3.2 Feature Correlation

Feature correlation analysis is crucial for understanding the relationships between variables within a dataset. It examines how variables influence one another and identifies whether their fluctuations occur simultaneously (Ashraf 2023). A high correlation coefficient indicates that two variables tend to increase or decrease together, suggesting a strong relationship. On the other hand, a low correlation coefficient implies that the variables do not exhibit a strong connection, with their fluctuations being largely independent of each other.

This analysis is particularly important when investigating complex systems, such as predicting pandemic infection rates, where multiple factors like mobility and infection data are at play. Identifying correlated features can help refine models by focusing on the most influential variables and eliminating redundant or less impactful ones. For example, understanding the correlation between human mobility and infection spread can shed light on how population movement contributes to the dynamics of virus transmission.

The correlation analysis was performed using the pandas 'corr()' method, which computes pairwise correlations between all variables, excluding any null values. The result is a correlation matrix, a powerful tool for visualizing how variables are related. This matrix can be further explored by plotting it, as shown in the figure 4.6, allowing the identification of key relationships that can inform feature selection and modelling decisions. High correlation between certain variables can suggest multi-

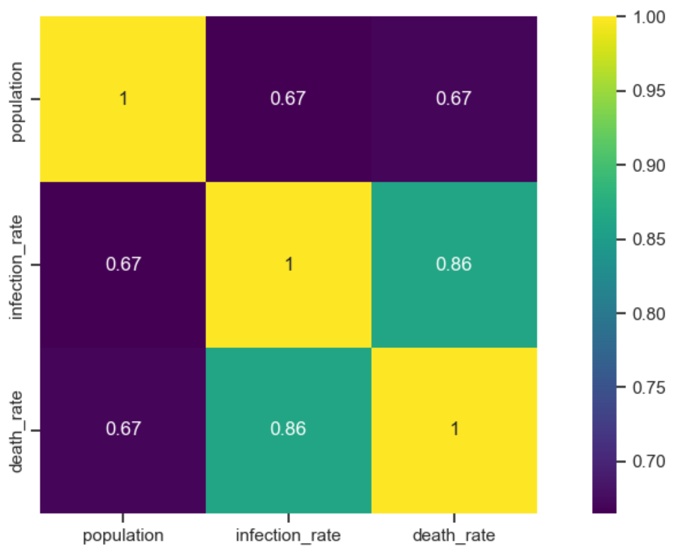


Figure 4.6: Feature correlation matrix

collinearity, while low correlations may prompt further investigation into potential hidden relationships or data quality issues.

The correlation between infection and death rates was comparatively high, reflecting a close relationship between the rise in infection numbers and the subsequent increase in death rates. This strong correlation emphasizes the direct impact of rising infection levels on mortality during the pandemic (Baud et al. 2020). However, population showed a low correlation with other features, which realistically seems inconsistent. This discrepancy could potentially be explained by poor data quality or inaccuracies in population reporting.

4.4 Graph Structures

A graph structure is a fundamental mathematical concept used to model relationships between entities, where objects are represented as nodes (or vertices), and the relationships between them are depicted as edges (or links) (Jaiswal 2021). Graphs can be used to represent a wide range of real-world systems, such as social networks, transportation routes, biological systems, and in the case of this project, infection spread and human mobility during a pandemic.

4.4.1 Components of a Graph

1. Nodes (Vertices): Represent individual entities or objects. In a social network, for example, nodes could represent people, and in the context of this project for pandemic modelling, they could represent geographic regions such as counties.

2. Edges (Links): Represent the relationships or interactions between nodes. In the mobility dataset, an edge might indicate movement between two counties, while in a social network, an edge might represent a connection or interaction between two people.

3. Weights: In some graph structures, edges have weights, representing the strength or magnitude of the connection. For example, in a mobility graph, the weight could represent the number of people travelling between two locations.

4. Directed and Undirected: A directed graph has edges with a specific direction, indicating a one-way relationship. In this project, a directed edge will represent the flow of people from one county to another. An undirected graph has edges that represent a mutual or two-way relationship, such as a friendship in a social network.

5. Static and Dynamic Graphs: A static graph has a fixed structure, where the nodes and edges do not change over time. A dynamic graph evolves, with nodes or edges added or removed as time progresses, useful for modelling systems like epidemic spread where interactions change over time.

4.4.2 PyTorch Geometric

To capture the spatio-temporal information from the dataset, the data is converted into a graph structure using PyTorch Geometric (PyG). PyG is a powerful library built on PyTorch that facilitates the implementation and training of Graph Neural Networks (GNNs) for structured data applications (Fey and Lenssen 2019).

PyG offers a range of methods specifically designed for deep learning on graphs and other irregular structures, commonly referred to as geometric deep learning. It includes implementations of various algorithms from published research papers. The library supports efficient mini-batch processing, enabling operations on both many small graphs and single large graphs. It is optimized for high performance, offering multi-GPU support and compatibility with advanced PyTorch features

Graphs are constructed for each day by fetching data from January 2022 to December 31, 2020. This date range is selected to balance computational efficiency, memory usage, and data size, optimizing the training process. In these graphs, nodes represent counties, with features including 'population,' 'infection rate,' and 'death rate'. The edges between nodes are determined by the 'source' and 'target' pairs from the mobility dataset, with edge features such as 'distance' and 'visitor flows' serving as edge weights.

These constructed graphs, along with their corresponding target values (infection rates), are stored in a dictionary format and saved in batches. This structure allows the neural network to perform regression tasks efficiently, leveraging both the spatial and temporal aspects of the data for accurate predictions.

Figure 4.7 illustrates the graph structure for December 31, 2020, showcasing the various nodes (counties) and their connections (mobility). Some nodes appear without connections due to thresholding applied during data preprocessing to enhance memory usage and computational efficiency. Despite this, the graph effectively captures the dataset's dynamics, where high mobility between counties correlates with a faster spread of the pandemic.

4.5 Graph Neural Network

Graph Neural Networks (GNNs) are a class of neural networks designed to process and analyse data represented in graph structures. Unlike traditional neural networks, GNNs leverage the connectivity and relationships between nodes in a graph to make predictions or extract insights (Kapoor et al. 2020). They operate by iteratively updating node representations based on the features of their neighbours and the graph's structure, allowing them to capture complex dependencies and patterns in the data. GNNs are particularly effective for tasks involving relational data, such

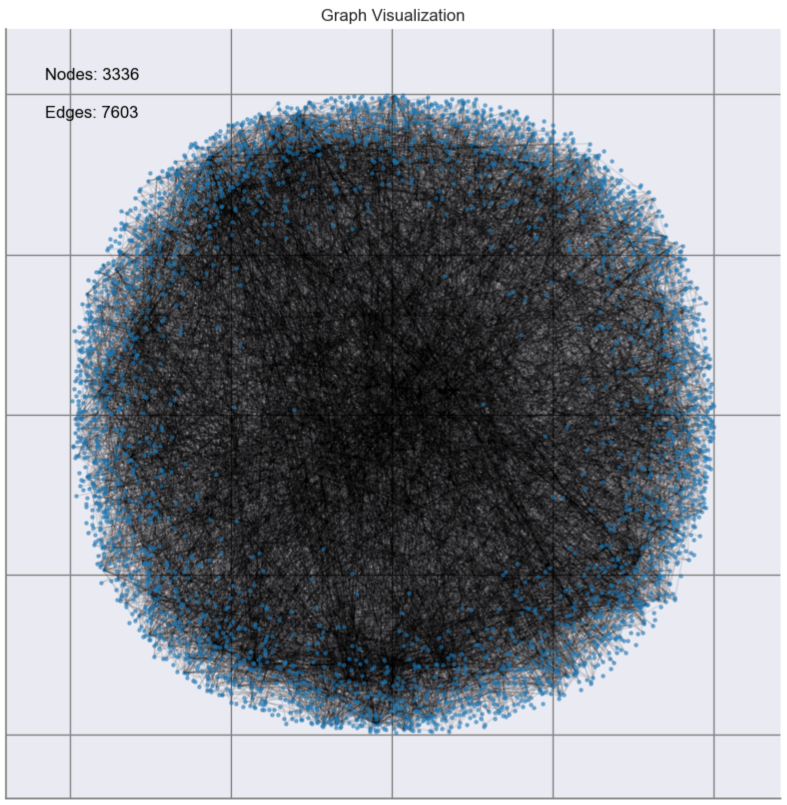


Figure 4.7: Graph Structure

as social network analysis, molecular chemistry, and traffic forecasting, where the interplay between entities is crucial (Ferludin et al. 2023).

Graph Convolutional Networks (GCNs) utilise graph convolutional layers to perform semi-supervised classification on graph-structured data. GCN convolutional operations aggregate information from a node's local neighbourhood to update its feature representation (Kipf and Welling 2016). This process involves applying a learned weight matrix to the node features, followed by a non-linear activation function. The core idea is to propagate labels from labelled nodes to unlabelled nodes through their connections, leveraging both node features and graph structure. This approach allows GCNs to capture complex patterns in graph data and improve classification and regression performance, even with limited labelled samples.

To implement this, the GCNConv layer from PyTorch Geometric is employed. The GCNConv layer performs the graph convolution by aggregating features from a node's neighbours and combining them with the node's own features. This aggregation is followed by a transformation using a learnable weight matrix and a non-linear activation function. Additionally, the layer applies normalisation to adjust for variations in node connectivity, ensuring a balanced contribution from all nodes regardless of their degree. Multiple GCNConv layers can be stacked to capture more complex relationships and dependencies within the graph, enhancing the model's ability to learn from and make predictions on graph-structured data.

4.5.1 Model Architecture

The architecture of the GNN model is designed to effectively learn from graph-structured data by employing a series of Graph Convolutional Network (GCN) layers. The architecture of the GNN model is detailed in Appendix A.1. The model employs a series of GCN layers to learn from graph-structured data. The design includes three GCN layers, each followed by batch normalisation, activation functions, and dropout for regularisation. The necessary libraries used in the project can be found in Appendix A.2. The model is structured as follows:

1. Initialisation:
   (a) First GCN Layer: The ‘GCNConv‘ layer initializes with ‘in_channels‘ as the input feature dimension and ‘hidden_channels1‘ as the number of output features. This layer is responsible for the initial transformation of node features based on their local neighbourhoods.
   (b) Batch Normalisation: Following the first GCN layer, ‘BatchNorm‘ normalises the node features, helping to stabilize and accelerate training by reducing internal covariate shift.
2. Second GCN Layer:
   (a) GCNConv Layer: The second ‘GCNConv‘ layer processes the output from the first GCN layer, transforming ‘hidden_channels1‘ features into ‘hidden_channels2‘ features. This additional layer allows the model to learn more complex patterns in the graph data.
   (b) Batch Normalisation: A second ‘BatchNorm‘ layer normalises the output of the second GCN layer, ensuring consistent scaling of the features before activation.
3. Third GCN Layer:
   (a) GCNConv Layer: The third ‘GCNConv‘ layer maps the features from ‘hidden_channels2‘ to ‘out_channels‘, providing the final feature representations for each node. This layer outputs the predictions for each node based on the learned graph structure.
   (b) Batch Normalisation: The final ‘BatchNorm‘ layer normalises the features after the third GCN layer, preparing them for the output.
4. Activation and Dropout:
   (a) Activation Function: A ’ReLU’ activation function is applied after each GCN layer (except the final one), introducing non-linearity to the model and allowing it to learn more complex relationships in the data.
   (b) Dropout: A dropout layer with a specified ‘dropout_rate‘ is used after the activation function in the first two GCN layers. Dropout helps to prevent overfitting by randomly setting a fraction of the input units to zero during training.

The model architecture is given below:

```
GNN(
    (conv1): GCNConv(3, 64)
    (batch_norm1): BatchNorm(64)
    (conv2): GCNConv(64, 32)
    (batch_norm2): BatchNorm(32)
    (conv3): GCNConv(32, 1)
    (batch_norm3): BatchNorm(1)
    (relu): ReLU()
    (dropout): Dropout(p=0.5, inplace=False))
```

The forward pass of the model involves applying these layers sequentially, with node features being transformed, normalised, activated, and regularised at each step. The output of the model is a tensor with shape '[num_nodes, out_channels]', representing the final feature embeddings for each node in the graph. The instantiated model, 'gcn_model', is configured with specific dimensions for input features, hidden layers, and output features, which are adjusted according to the specific needs of the task at hand.

4.5.2 Model Training and Validation

Training Phase

The training process involves the 'train()' function, where the model learns patterns in the data by adjusting its internal parameters to minimise the prediction error. For each batch of data from the training set, the model makes predictions by performing a forward pass through the GNN. During this step, the input data, including node and edge features, are passed through multiple Graph Convolutional layers.

The predicted outputs are then compared to the actual target values, and the difference between them is calculated using the Mean Squared Error (MSE) as the loss function. This loss value indicates how far the model's predictions are from the true values. The 'AdamW' optimiser then updates the model's weights in the direction that reduces this error. After each batch, the optimiser adjusts the weights of the network to improve the model's performance gradually.

Throughout this process, techniques like dropout and batch normalisation are applied to regularise the model and prevent overfitting. After each batch, the average training loss is computed, providing insight into how well the model is learning.

Validation Phase

The 'validate()' function handles the evaluation of the model on unseen validation data, offering a way to assess the model's generalisation capabilities. During validation, the model is not trained further, meaning its parameters are not updated. Instead, it performs a forward pass on the validation data, making predictions which are then compared to the actual target values using the same MSE loss function as in training.

The purpose of validation is to ensure that the model does not overfit to the training data and can generalise to new, unseen data. After the predictions are made, the loss is calculated to determine how accurate the model is on the validation set. Monitoring validation loss is crucial for detecting when the model is overfitting—if the validation loss increases while the training loss continues to decrease, this is a sign that the model is not generalising well.

Both the training and validation phases are repeated across multiple time windows in the walk-forward validation approach, ensuring the model is trained on a rolling set of data and evaluated on progressively newer data points. Walk-forward validation is particularly useful for time series data, where the objective is to predict future values based on past observations (Yoon 2021). This method splits the data into training and validation windows, allowing the model to train on historical data while being evaluated on subsequent data, closely mimicking a real-time forecasting environment. During each validation step, the training window includes a fixed set of data points, while the validation window moves forward incrementally over time. This ensures that the model is evaluated on more recent data as it progresses, improving the model's ability to make predictions for future time steps.

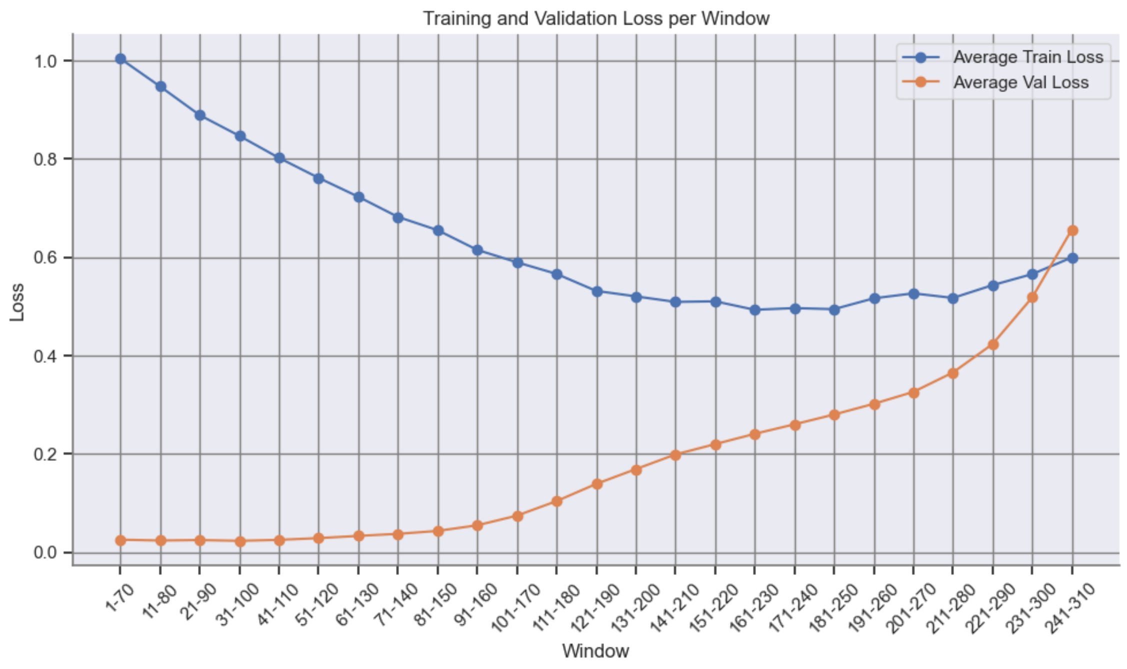


Figure 4.8: Model Learning Curve

As observed in the learning curve graph generated during training (figure 4.8), the learning curve graph generated during training, the model demonstrates proper learning behaviour, as indicated by the decreasing training loss. This reflects the model's ability to adjust its weights and biases effectively. However, the validation curve shows an upward trend, particularly towards the latter stages, suggesting signs of overfitting. This indicates that additional steps, such as more hyperparameter tuning or improving the data quality, may be necessary to enhance the model's performance and generalisation.

Chapter 5

Testing and Evaluation

The testing of the model's predictive performance was conducted using a test dataset that had been excluded from the original data prior to training and validation. The testing procedure closely mirrors the validation process, with only a forward pass being performed and no backward pass or weight updates. The predicted outputs were then compared against the actual targets (infection rates), and the loss values were computed to assess model performance.

For this model, the test results showed:

1. Mean Squared Error (MSE): 1.1185

2. Mean Absolute Error (MAE): 0.2784

MSE and MAE are two common metrics used to evaluate regression models. MAE measures the average magnitude of errors between predicted and actual values, giving a straightforward interpretation of how far off predictions are on average. MSE, on the other hand, squares the errors before averaging, which penalises larger errors more heavily than smaller ones. This makes MSE more sensitive to outliers. While both are useful, MSE can highlight models with larger mistakes, whereas MAE gives a more direct sense of the average error magnitude.

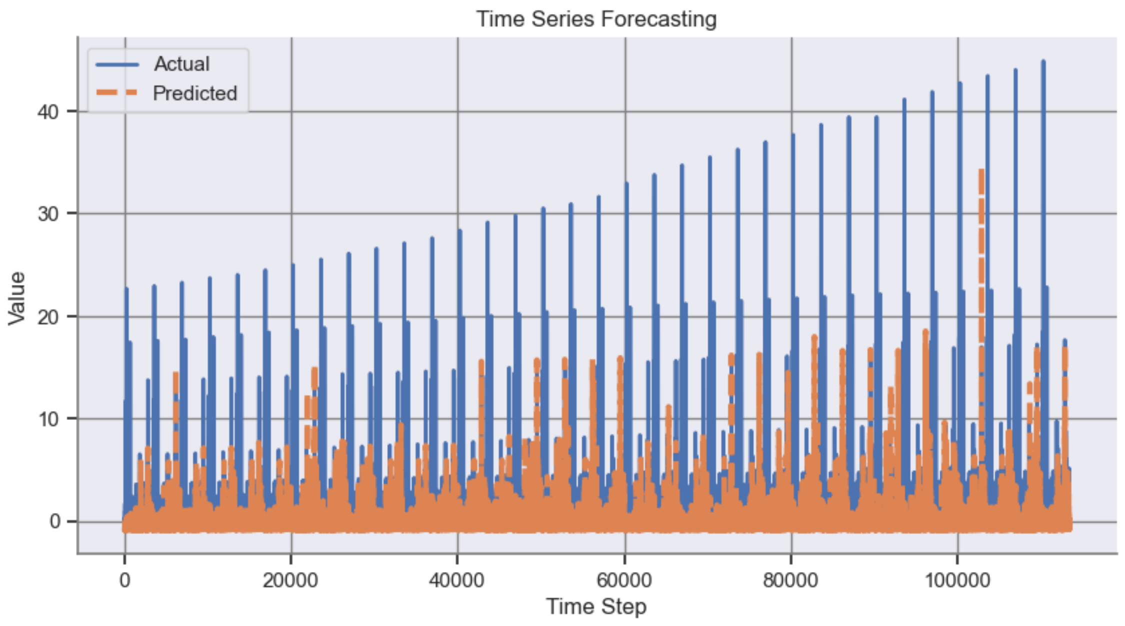


Figure 5.1: Final Model Prediction

Finally, the predicted results were plotted alongside the actual infection rates to visualise the model's efficiency, as seen on figure 5.1. The graph indicates that, while the model offers some predictive capability, it is far from perfect. There remains room for improvement through further hyperparameter tuning and enhanced data preprocessing.

Chapter 6

Conclusion

This project successfully developed a Graph Neural Network (GNN) model that integrated mobility data with human mobility information to model the spread of COVID-19 across various regions. Despite challenges such as integrating large datasets, handling technical complexities, and overcoming software limitations, the model met its core objectives. The innovative combination of mobility data and epidemiological factors allowed for accurate predictions of virus spread, demonstrating the effectiveness of GNNs in handling complex, spatio-temporal data. The project also highlighted the importance of choosing the right frameworks and implementing efficient data processing techniques for large-scale datasets. Moreover, this approach has the potential to serve as a powerful tool for healthcare officials to track and prevent widespread infections, offering critical insights into the dynamics of virus transmission. Although further improvements in model performance and scalability are possible, the project provides a solid foundation for future research and practical applications in public health modelling.

6.1 Future Work

The GNN framework developed in this project has significant potential for broader applications beyond the scope of COVID-19. Future work can focus on extending the model to cover larger geographical regions or even cross-country analyses, allowing policymakers to track and predict disease spread across different borders. This approach can also be adapted for other infectious diseases, such as influenza or dengue fever, by modifying the input data to reflect disease-specific transmission dynamics and demographic factors. Additionally, future work could focus on enhancing the scalability and real-time prediction capabilities of the model, making it more responsive to rapidly changing situations. Improved hyperparameter tuning, alternative model architectures, and incorporating more detailed demographic data could further refine the accuracy and utility of this design in various public health applications.

Chapter 7

BCS Project Criteria & Self-Reflection

The project successfully meets the six outcomes expected by the Chartered Institute for IT as follows:

- Application of Practical and Analytical Skills: This project demonstrates the application of skills gained during the degree programme, including data preprocessing, model training, and the use of advanced machine learning algorithms. These aspects are detailed in the Implementations section (chapter 4).
- Innovation and Creativity: By employing Graph Neural Networks to predict COVID-19 infection rates based on human mobility data, the project applies an innovative approach to pandemic modelling. The novel integration of mobility patterns with epidemiological data is discussed in the Literature Review and Model Design sections (chapters 2 and 3).
- Synthesis of Information and Evaluation: The project synthesises various data sources, methodologies, and techniques to provide a high-quality solution for infection rate prediction. An evaluation of this solution, including the model's performance, is covered in the Dataset section (section 4.1) and the Testing and Evaluation chapter (chapter 5).
- Meeting a Real Need: COVID-19's significant public health impact highlights the need for accurate and scalable prediction models. This project addresses this need, providing a tool for predicting infection rates and outbreak hotspots, as discussed in the Background and Objectives sections (chapter 2 and section 1.2).
- Self-Management of a Significant Piece of Work: The completion of this project required effective time management, organization, and independent research. While the project was delivered successfully, certain challenges were encountered, such as the need for additional data processing due to the complexities of working with mobility datasets. Adjusting the scope of work and managing unexpected delays in data acquisition were crucial learning points.

7.1 Self Evaluation

Managing the project involved balancing technical complexity with practical constraints, and while the project achieved its core objectives, there were several areas for improvement. One of the main challenges was the integration of various data sources, which required extensive pre-processing and cleaning. The mobility dataset, in particular, posed a significant issue due to its sheer size (24 GB), which could not be efficiently handled in Jupyter notebooks. This led to the identification and use of 'Dask', a library for parallel computing, to manage the dataset more effectively. Further thresholding techniques were applied to reduce the number of data points without compromising the model's learning capability, streamlining the data processing phase.

An area for improvement is feature engineering. The inclusion of additional epidemiological features, such as vaccination data, healthcare facilities, and proximity to hospitals, could have enriched the model's predictions. Furthermore, more domain knowledge on factors influencing virus spread would have helped in selecting better features.

Another challenge arose with the initial model architecture, which proved to be suboptimal. Early hyperparameter tuning only produced an average-performing model, and due to time constraints, further tuning and exploration of alternative models were not feasible. As a result, the project proceeded with this model, although there was clear potential for further optimization, particularly in terms of scalability and real-time prediction capabilities.

Additionally, the choice of software frameworks was another hurdle. Initially, TensorFlow was selected for implementing the GNN. However, the online documentation for TensorFlow's graph modelling was insufficient for understanding the specific requirements of the project. This resulted in significant delays, as it became difficult to realise the graph-based model. After considerable struggle, the decision was made to switch to PyTorch Geometric, which offered extensive and well-documented resources. This transition was a turning point, enabling the graph structure to be implemented successfully within two days.

Despite these challenges, a key strength of the project was the innovative approach taken to combine mobility data with epidemiological models. This demonstrated creativity and technical skill, particularly in leveraging GNNs to model complex dynamics. However, the time spent resolving data and framework-related issues limited the exploration of alternative models and optimizations for the GNN architecture. In future, a more iterative approach to model development and evaluation, along with earlier identification of suitable tools, could help address such limitations earlier in the process.

Bibliography

Arenas, Alex et al. (2020). “Modeling the spatiotemporal epidemic spreading of COVID-19 and the impact of mobility and social distancing interventions”. In: *Physical Review X* 10.4, p. 041055.

Ashraf, Abdallah (2023). *Correlation in machine learning—All you need to know.* URL: `https://medium.com/@abdallahashraf90x/all-you-need-to-know-about-correlation-for-machine-learning-e249fec292e9` (visited on 09/12/2024).

Baud, David et al. (2020). “Real estimates of mortality following COVID-19 infection”. In: *The Lancet infectious diseases* 20.7, p. 773.

Dong, Ensheng, Hongru Du, and Lauren Gardner (2020). “An interactive web-based dashboard to track COVID-19 in real time”. In: *The Lancet infectious diseases* 20.5, pp. 533–534.

Ferludin, Oleksandr et al. (2023). “TF-GNN: Graph Neural Networks in TensorFlow”. In: *CoRR* abs/2207.03522. URL: `http://arxiv.org/abs/2207.03522`.

Fey, Matthias and Jan Eric Lenssen (2019). “Fast graph representation learning with PyTorch Geometric”. In: *arXiv preprint arXiv:1903.02428.*

Gao, Junyi et al. (2021). “STAN: spatio-temporal attention network for pandemic prediction using real-world evidence”. In: *Journal of the American Medical Informatics Association* 28.4, pp. 733–743.

Gravina, Alessio and Davide Bacciu (2024). “Deep learning for dynamic graphs: models and benchmarks”. In: *IEEE Transactions on Neural Networks and Learning Systems.*

Jaiswal, Dhruvisha (2021). *A brief study on Graph Data Structure and its Applications.* URL: `https://medium.com/@dhruvishakali99/a-brief-study-on-graph-theory-e5dcfba0cc0f` (visited on 09/12/2024).

Kang, Yuhao et al. (2020). “Multiscale dynamic human mobility flow dataset in the US during the COVID-19 epidemic”. In: *Scientific data* 7.1 (390), pp. 1–13.

Kapoor, Amol et al. (2020). “Examining covid-19 forecasting using spatio-temporal graph neural networks”. In: *arXiv preprint arXiv:2007.03113.*

Kipf, Thomas N and Max Welling (2016). “Semi-supervised classification with graph convolutional networks”. In: *arXiv preprint arXiv:1609.02907.*

Mukhiya, Suresh Kumar and Usman Ahmed (2020). *Hands-On Exploratory Data Analysis with Python: Perform EDA techniques to understand, summarize, and investigate your data.* Packt Publishing Ltd.

Panagopoulos, George, Giannis Nikolentzos, and Michalis Vazirgiannis (2021). “Transfer graph neural networks for pandemic forecasting”. In: *Proceedings of the AAAI Conference on Artificial Intelligence.* Vol. 35. 6, pp. 4838–4845.

Rocklin, Matthew et al. (2015). “Dask: Parallel computation with blocked algorithms and task scheduling.” In: *SciPy*, pp. 126–132.

Son Hy, Truong et al. (2022). “Temporal Multiresolution Graph Neural Networks For Epidemic Prediction”. In: *arXiv e-prints*, arXiv–2205.

Thomas, K, N Weiland, and S LaFraniere (2020). *F.D.A. Advisory Panel Gives Green Light to Pfizer Vaccine*. URL: `https://www.nytimes.com/2020/12/10/health/covid-vaccine-pfizer-fda.html` (visited on 09/12/2024).

Yoon, Hyoup-Sang (2021). "Time series data analysis using wavenet and walk forward validation". In: *Journal of the Korea Society for Simulation* 30.4, pp. 1–8.

Appendix A

Code

A.1 Model Architecture

```python
class GNN(nn.Module):
    def __init__(self, in_channels, hidden_channels1,
  hidden_channels2, out_channels, dropout_rate=0.5):
        super(GNN, self).__init__()

        # First GCN Layer
        self.conv1 = GCNConv(in_channels, hidden_channels1)
        self.batch_norm1 = BatchNorm(hidden_channels1)

        # Second GCN Layer (new layer added)
        self.conv2 = GCNConv(hidden_channels1, hidden_channels2)
        self.batch_norm2 = BatchNorm(hidden_channels2)

        # Third GCN Layer
        self.conv3 = GCNConv(hidden_channels2, out_channels)
        self.batch_norm3 = BatchNorm(out_channels)

        # Activation function and Dropout
        self.relu = nn.ReLU()
        self.dropout = nn.Dropout(dropout_rate)

    def forward(self, data):
        x, edge_index = data.x, data.edge_index

        # First GCN layer + BatchNorm + ReLU + Dropout
        x = self.conv1(x, edge_index)
        x = self.batch_norm1(x)
        x = self.relu(x)
        x = self.dropout(x)

        # Second GCN layer + BatchNorm
        x = self.conv2(x, edge_index)
        x = self.batch_norm2(x)
        x = self.relu(x)
        x = self.dropout(x)

        # Third GCN layer + BatchNorm
        x = self.conv3(x, edge_index)
        x = self.batch_norm3(x)

        # Output shape should be [num_nodes, out_channels]
        return x
# Model Initialisation
```

```
gcn_model = GNN(in_channels=3, hidden_channels1=64,
    hidden_channels2=32, out_channels=1)
```

Listing A.1: GNN Model Architecture

A.2 Essential Libraries

```
    # ========== Standard Library Imports ==========
import os  # For interacting with the operating system (file
    handling)
import os.path as osp  # For working with file paths conveniently
import glob  # For retrieving files by matching patterns
import warnings  # To control the display of warning messages
import time  # To measure execution time

# ========== Third-Party Libraries ==========
import pandas as pd  # For data manipulation and analysis
import dask.dataframe as dd  # For parallel computing with large
    datasets
from geopy.distance import geodesic  # For calculating geographic
    distances
from sklearn.preprocessing import StandardScaler  # For
    standardising dataset features
import matplotlib.pyplot as plt  # For data visualisation
import seaborn as sns  # For statistical data visualisation
import networkx as nx  # For creating and visualising network
    graphs
import numpy as np  # For numerical computations
from sklearn.metrics import mean_squared_error, mean_absolute_error
      # For performance metrics

# ========== PyTorch and PyTorch Geometric Imports ==========
import torch  # Core PyTorch library for tensor operations and
    neural networks
import torch.nn as nn  # For creating neural network layers and
    architectures
import torch.optim as optim  # For optimisers (AdamW)
import torch_geometric  # For graph neural network implementations
    in PyTorch
from torch_geometric.data import Data, Batch  # For handling graph
    data
from torch_geometric.loader import DataLoader  # For batching and
    loading graph datasets
from torch_geometric.nn import GCNConv, BatchNorm  # GCN layer and
    BatchNorm for graph neural networks
import torch.nn.functional as F  # For common functions such as
    ReLU, softmax, etc.

# ========== Custom Configurations ==========
# Set custom Seaborn theme for visualisation aesthetics
custom = {
    'xtick.bottom': True,
    'ytick.left': True,
    'grid.color': 'gray',
    'axes.spines.right': False,
    'axes.spines.top': False,
    'axes.edgecolor': 'gray'
}
sns.set_theme(style='darkgrid', rc=custom)
```

```python

# ========== Suppress Warnings ==========
warnings.filterwarnings("ignore")  # To hide unnecessary warnings
    during execution
```

Listing A.2: Import Section

Appendix B

Original design document

B.1 Project Description

The project aims to develop a robust Graph Neural Network (GNN) model to predict COVID-19 infection rates by analysing human mobility patterns. This involves integrating comprehensive infection rate data from reliable sources with human mobility data, such as transportation logs and mobile phone GPS data, along with geographical information to facilitate regional-level predictions. The GNN model will be designed to adapt to changing infection patterns and seamlessly incorporate new data. Additionally, the project will produce intuitive and interactive visualizations, as well as detailed reports, to present predicted infection trends and highlight potential outbreak hotspots. The technical development will prioritize scalability and performance, ensuring the model can handle large volumes of data efficiently and provide real-time or near-real-time predictions. The project is grounded in the latest research and methodologies, with a focus on the practical application of advanced machine learning techniques. Overall, this project will demonstrate an innovative approach to future pandemic prediction, applying advanced analytical skills and creativity to develop a tool that can significantly aid in managing public health responses during pandemics. The deliverables will include not only the GNN model but also comprehensive documentation of the research findings, methodologies, and a critical self-evaluation of the process.

B.2 Aims and Requirements

1. Create a robust machine learning model specifically designed to forecast the number of new COVID-19 infected cases, adaptable to changes in infection patterns, and incorporating new data seamlessly using advanced statistical methods and algorithms.

2. Retrieve comprehensive infection rate data from reliable sources and integrate it with human mobility data, including transportation logs and mobile phone GPS data, along with geographical data to analyse and predict infection rates at different regional levels.

3. Develop intuitive visualizations and detailed reports to present predicted infection trends, highlight key insights and potential outbreak hotspots, and ensure accessibility for public health officials to make informed decisions.

B.2.1 Essential Requirements

1. Data Collection and Integration
   (a) Retrieve infection rate data from reliable sources such as health organizations and governmental databases.
   (b) Incorporate geographical data to analyse and predict infection rates at different regional levels (e.g., cities, counties).
   (c) Gather anonymous human mobility and population data.
   (d) Ensure data is anonymous, pre-processed, cleaned, and integrated for analysis.
2. Predictive Modelling
   (a) Develop and implement a machine learning model capable of predicting new infection cases based on the integrated data.
   (b) Train the model using historical data to optimize prediction accuracy.
   (c) Validate the model's performance using appropriate evaluation metrics.
3. Visualization and Reporting
   (a) Create visualizations to present the predicted infection trends over time.
   (b) Develop reports to display key insights and predictions.
4. Scalability and Performance
   (a) Ensure the predictive model and system can handle large volumes of data efficiently.
   (b) Optimize the performance of the model for real-time or near-real-time predictions.

B.3 Development Summary

The development of this project will begin with selecting the appropriate model architecture. A Graph Neural Network (GNN) will be utilized for a specific country, with nodes representing county-level regions and edges representing mobility data collected during the pandemic/epidemic period. The model will be implemented using Python in a Jupyter Notebook, with either Microsoft Visual Studio Code or Google Colab as the Integrated Development Environment (IDE), depending on the required processing power. Python is chosen for its extensive libraries, such as PyG and TensorFlow, which include built-in GNN packages.

The workflow starts by gathering essential datasets for US COVID-19 data, encompassing infection rates, mortality rates, and recovery rates from credible health departments and governmental databases. Additionally, human mobility data, sourced in openly available data sources, is collected to analyse population movement patterns. Geographic distance data between countries is also obtained to explore potential cross-border impacts on COVID-19 transmission dynamics. Following data collection, thorough preprocessing ensures dataset cleanliness and balance. An initial exploratory data analysis phase provides insights into dataset characteristics before applying machine learning techniques. Subsequently, a Graph

Neural Network (GNN) model with multiple hidden layers is constructed to capture intricate relationships among COVID-19 metrics, mobility patterns, and geographic distances. The model is trained using the integrated datasets to optimize predictive accuracy, facilitating forecasts of COVID-19 outcomes based on learned data dynamics.

B.4 Data Sources

The infection dataset is sourced from the COVID-19 Data Repository by the Centre for Systems Science and Engineering (CSSE) at Johns Hopkins University. This repository serves as a critical open-source resource for tracking and analysing the ongoing COVID-19 pandemic globally. Managed by the CSSE at Johns Hopkins University, it provides comprehensive data on COVID-19 cases, deaths, and vaccination efforts across various geographic scales and time periods. The repository supports the operation of the 2019 Novel Coronavirus Visual Dashboard, in collaboration with the ESRI Living Atlas Team and the Johns Hopkins University Applied Physics Lab (JHU APL).

The human mobility dataset utilized in this project originates from the COVID-19 US Flows repository maintained by the GeoDS Lab at the Department of Geography, University of Wisconsin-Madison. This dataset offers a comprehensive collection of multiscale dynamic human mobility flow data across the United States during the COVID-19 epidemic, sourced from SafeGraph. It includes daily and weekly origin-to-destination (O-D) population flows at various geographic levels such as census tract, county, and state, spanning from January 2019 to 2021. Both the datasets are open sourced and anonymised, and hence used with prior permission.

B.5 Testing and Evaluation

The developed code for the GNN model will be tested using Python's unit testing framework, 'unittest'. Each individual block of code or module will undergo testing to ensure it functions as intended. The project will be evaluated based on key performance indicators (KPIs) such as accuracy, precision, recall, and F-score, as well as error metrics like MAE and RMSE. There will be no human testers (beta testing) or human feedback involved in this project.

B.6 Project Ethics and Human Participants

This project uses human related data which is sourced from publicly availably open-sourced data repositories which is already anonymised. The project does not involve any human participants nor includes any requirement analysis.

B.7 BCS project criteria

1. An ability to apply practical and analytical skills gained during the degree programme: This project applies the practical and analytical skills acquired during the degree program, including advanced data processing, machine learning, and graph theory. The implementation of a Graph Neural Network (GNN)

model demonstrates the practical application of these skills to solve real-world problems in human mobility prediction during pandemics. Detailed information on the methods and techniques employed can be found in the ”Development Summary” section.

2. Innovation and creativity: The project employs an innovative approach by leveraging a GNN to predict infection rates through the analysis of human mobility patterns. The creative integration of dynamic multiscale data and advanced neural network techniques is demonstrated by experimenting with various combinations of neural layers, activation functions, and optimizers. Additional details are available in the “Project Description” and “Development Summary” sections.

3. Synthesis of information, ideas, and practices to provide a quality solution together with an evaluation of that solution: This project synthesizes information from various sources, including infection rates, human mobility data and machine learning literature, to develop a robust GNN model. The evaluation of the solution will be based on key performance indicators like accuracy, precision, recall, F-score, and error metrics. Detailed evaluation methods are outlined in the “Testing and Evaluation” section.

4. That your project meets a real need in a wider context: The project addresses the real-world need for accurate infection rates predictions during pandemics, which can inform public health strategies and interventions. By providing a model that predicts movement patterns and spread of diseases, the project contributes to better decision-making in managing public health crises.

5. An ability to self-manage a significant piece of work: The project demonstrates self-management skills through the planning, development, and implementation phases, ensuring timely delivery of milestones. The use of project management tools and methodologies, as detailed in the “Project Plan” section, illustrates the structured approach taken to manage this significant piece of work.

6. Critical self-evaluation of the process: A critical self-evaluation will be conducted throughout and at the end of the project to assess the effectiveness of the methods used, the challenges encountered, and the lessons learned.

B.8 UI/UX Mock-up

Since this project is a research-focused endeavour with no user interface (UI) or user experience (UX) components, there is no mock-up or wireframe. The primary objective is to develop and validate a Graph Neural Network (GNN) model for predicting infection rates based on human mobility patterns. The emphasis is on the technical development and analysis of the model rather than on user interaction or interface design. As such, the deliverables will consist of code, datasets, and detailed documentation of the research findings and methodology, as outlined in the “Project Description” and “Development Summary” sections.

B.9 Project Plan

The project timeline is outlined in the figure B.1.

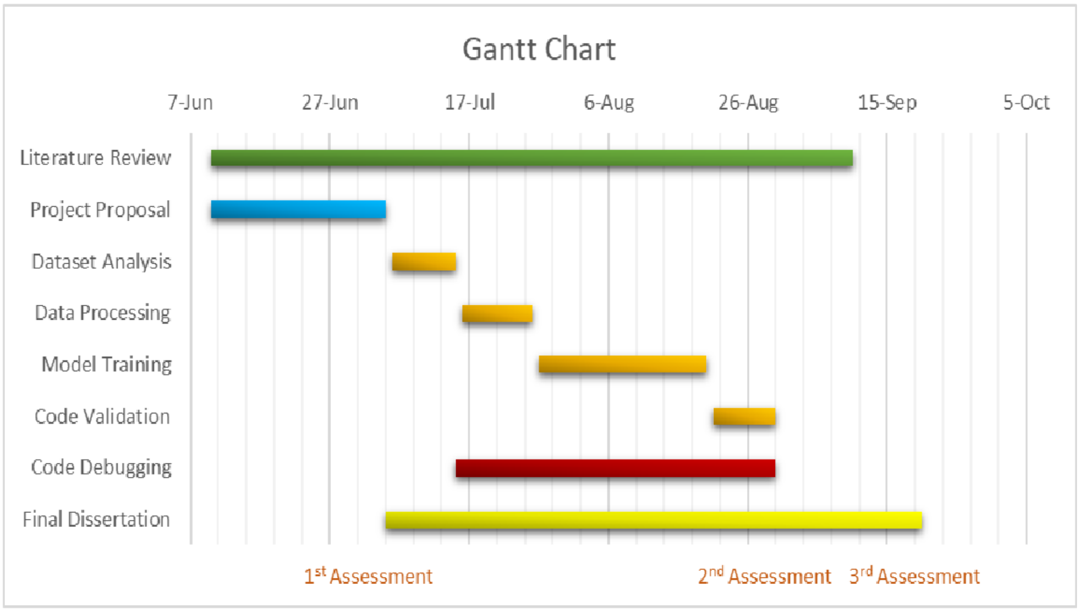


Figure B.1: Gantt Chart

B.10 Risks & Contingency Plans

Risk	Contingencies	Likelihood	Impact
Software Failure	Using version control systems; regular testing and debugging.	Medium	High
Programming Issues	Guidance from experienced academic staff, peer code reviews.	Low	Medium
Dataset Issues	Identify similar but alternative data sources.	Low	High
Loosing Backup Files	Maintain multiple versions of the code during development using version control such as git.	High	High
Insufficient computational resources	Using University systems and resources to access IDEs and GPUs remotely; code optimisation.	High	Medium
Time Management	Detailed project planning with milestones and regular progress monitoring.	High	High
Supervisor Miscommunication	Regular project meetings; frequent status updates	Low	Low

Table B.1: The risks and contingencies associated with the project