Edelweiss Applied Science and Technology
ISSN: 2576-8484
Vol. 10, No. 1, 727-781
2026
Publisher: Learning Gate
DOI: 10.55214/2576-8484.v10i1.11687
© 2026 by the author; licensee Learning Gate

Predicting daily new cases of COVID-19 using data mining techniques

Sathishkumar, V. E.[1]*
[1]School of Engineering and Technology (SET), Sunway University, Malaysia; sathishv@sunway.edu.my (S.V.E.).

Abstract: Accurate forecasting of daily new COVID-19 cases remains a critical challenge for effective epidemiological surveillance and healthcare system preparedness. This study proposes a data-driven predictive framework that leverages advanced data mining techniques to model daily COVID-19 case trends using a comprehensive dataset comprising case statistics, demographic attributes, vaccination status, cluster information, and temporal indicators. The methodology involves systematic data preprocessing, feature engineering—including lagged and temporal variables—and the application of regression-based and time-series forecasting models. Model performance is rigorously evaluated using standard statistical error metrics to assess predictive reliability. The results indicate that incorporating vaccination categories, cluster-related features, and historical lag variables substantially enhances forecasting accuracy, enabling the models to capture nonlinear dynamics and temporal dependencies in case progression. The findings demonstrate the effectiveness of data mining approaches in improving short-term COVID-19 case prediction, thereby providing a technically robust framework for real-time epidemic modeling and informed public health decision-making.

Keywords: *COVID-19 prediction, Data mining, Public health, Regression analysis, Time series forecasting, Vaccination impact.*

1. Introduction

The COVID-19 pandemic has engulfed the entire world, challenging healthcare systems, economies, and societies at every turn. As the virus evolves, controlling its spread remains a priority globally. Basic sound pandemic control relies on daily new COVID-19 case prediction, which directly influences resource allocation and planning within health systems for prompt and effective public health interventions. These models are usually very informative but narrow in scope and do not incorporate multifactorial predictors such as vaccination status, case clusters, and holiday-influenced dynamics. This limitation affects the effectiveness of these models when dealing with rapidly changing trends of cases in real-world settings, highlighting the need for more sophisticated and data-rich approaches.

In this regard, the present study utilizes an elaborate dataset representing a wide range of COVID-19-related factors, including but not limited to detailed case statistics, demographic information, and vaccination records. This paper bridges the gap between theoretical modeling and practical applications using advanced data mining methods. It will explain how vaccination programs, community clusters, and seasonal elements interact in determining the pandemic trajectory, combining regression analysis with time-series forecasting. This holistic approach provides better accuracy not only in predicting new cases each day but also gives useful insights for policymakers to stay ahead of the pandemic curve.

1.1. Problem Statement

One of the significant barriers to performing public health planning is the challenge of making an accurate forecast of new COVID-19 cases daily in real-time. Current prediction methods do not

History: Received: 14 October 2025; Revised: 1 December 2025; Accepted: 4 December 2025; Published: 7 January 2026
* Correspondence: sathishv@sunway.edu.my

consider the interaction of key factors such as vaccination coverage, formation of case clusters, and behavioral changes brought on by holidays. These inadequacies leave health systems vulnerable to sudden rises in cases, which may burden available resources and jeopardize public safety. The dynamic nature of the pandemic, its new variants emerging, and vaccination trends have highlighted the shortcomings of existing models. This situation necessitates a more reliable and comprehensive forecasting system that considers these relationships and provides accurate, up-to-date predictions.

1.2. Objective

Based on the prevailing understanding of the subject, this study undertakes the development of a prediction model using state-of-the-art data mining techniques that can predict the daily new cases of COVID-19 infection. In summary, in order to validate the performance of regression and time-series models, preprocess and engineer features with the aim to enhance data quality, and identify and quantify the most important factors affecting case trends. A key area of focus is examining how vaccination status, cluster data, and lagged factors might increase forecasting accuracy. The ultimate aim would be the provision of an evidence-based framework to help government and healthcare decision-making during the current pandemic.

1.3. Motivation

Given that it tackles important, practical issues, this research has the potential to revolutionize pandemic management. The ability to forecast accurately will enable the community, governments, and healthcare systems to prepare for surges, utilize resources optimally, and conduct treatments on time. This study integrates case cluster data with vaccination status, offering a fresh perspective beyond traditional modelling techniques. It addresses key questions about how vaccination efforts and behavioural patterns reflect on case dynamics. Moreover, these findings have wide-ranging implications for future outbreaks, providing a scalable and adaptable framework for epidemiological modelling.

1.4. Research Questions

This study is guided by the following key questions:

- What are the most critical predictors of daily new COVID-19 cases, and how do variables such as vaccination status, case clusters, and holiday effects influence these trends?
- How can the integration of lagged features and historical case data enhance the accuracy of predictive models?
- What are the comparative strengths and limitations of regression analysis versus time-series forecasting in modeling COVID-19 case trends?
- To what extent do the model's predictions align with actual case data, and how can these insights inform public health policy and pandemic preparedness?
- By answering these questions, this research aims to deliver a robust and practical solution to a pressing global challenge, equipping policymakers with the tools they need to mitigate the impact of COVID-19.

Table 1.
Comparison Table of Research Papers.

	Author(s) and Year/Reference	**Dataset Used**	**Algorithms Used**	**Evaluation Metrics**
1.	Chimmula and Zhang [1]	Johns Hopkins University COVID-19 Data Repository	Long Short-Term Memory (LSTM)	RMSE, MAE
2.	Rustam et al. [2]	WHO COVID-19 Dataset	ARIMA, Linear Regression	RMSE, MAPE
3.	Tuli et al. [3]	Public datasets (Google Mobility)	ARIMA, Bayesian regression	R^2 score, RMSE
4.	Ahuja et al. [4]	Indian Ministry of Health Dataset	Decision Trees, XGBoost	Accuracy, Precision, Recall
5.	Hu et al. [5]	Proprietary hospital data (China)	SVM, Random Forest	F1-score, ROC-AUC
6.	Yan et al. [6]	COVID-19 Data Hub	Gradient Boosting Machine (GBM)	MAE, RMSE
7.	Petropoulos and Makridakis [7]	Google COVID-19 Dataset	LSTM, CNN	MSE, MAE
8.	Altay et al. [8]	WHO COVID-19 Dashboard	ARIMA, Prophet	MAE, RMSE
9.	Xu et al. [9]	Public data (GitHub COVID-19)	Deep Neural Networks (DNN)	Accuracy, Precision, Recall
10.	Zhang et al. [10]	WHO Dataset, country-specific data	Improved LSTM-ARIMA, CNN	RMSE, F1-score
11.	Mengistie [11]	WHO, Worldometer, GitHub, and DingXiangYuan	Fbprophet Python library	N/A
12.	Solayman et al. [12]	Israeli Ministry of Health	SMOTE, Logistic Regression, Decision Tree, Random Forest, KNN, SVM, AdaBoost, XGBoost, ANN, CNN, hybrid CNN-LSTM	Accuracy, Precision, Recall, F1 Score
13.	Ghafouri-Fard et al. [13]	N/A	ANN, LSTM, ANFIS, ARIMA, MLP	RMSE, MAE, R^2, MAPE
14.	Awadh et al. [14]	Self-report study/data	Naive Bayes, MLP, J48	Pearson correlation coefficient, Prediction accuracy, building time, and error average
15.	Allmuttar and Alkhafaji [15]	A mixture of datasets from various scenarios	K-Means Clustering	Normalized cross-correlation
16.	Ahouz and Golabpour [16]	Johns Hopkins University COVID-19 Data Repository	Least-Square Boosting (LSBoost)	Percent error between predicted and actual values
17.	Satar et al. [17]	Public dataset	Naïve Bayes, Decision Tree, ANN	Accuracy, Precision, Recall
18.	Muhammad et al. [18]	Coronavirus dataset of the Korea Centers for Disease Control & Prevention (KCDC)	Decision Tree, SVM, Naive Bayes, Logistic Regression, Random Forest, KNN	Accuracy
19.	Rane et al. [19]	N/A	LSSVM, SIR, SEIR, SVM, ETS, Linear Regression, LASSO Regression, ANN, ARIMA, MLP, ELM, NNETAR, CNN,	Advantage, Disadvantage

			RNN, LSTM, GRU, NARNN, Bi-LSTM	
20.	NOOR et al. [20]	Humandata website	Linear Regression	RMSE, MAE

2. Literature Review

A review of existing literature on COVID-19 forecasting highlights a variety of models used for data analysis and prediction, including both traditional statistical methods and advanced machine learning algorithms. Artificial Neural Networks (ANN), Convolutional Neural Networks (CNN), Linear Regression, Random Forest, and XGBoost have been chosen for our study due to their robust performance in handling structured and unstructured data. These algorithms also correspond to the trend of previous research, which indicates that the combination of various methods is necessary for accurate and reliable forecasts.

Among these, time-series models, including ARIMA and LSTM, are widely applied in Chimmula and Zhang [1] and other studies. However, most of them perform poorly in handling nonlinear patterns or highly complex datasets. On the other hand, research by Ahuja et al. [4] and Hu et al. [5] has shown that our backbone models, such as Random Forest and XGBoost, are very effective at capturing complex correlations within the data. CNNs have proven their versatility for policy-based scenario modeling through successful applications of hybrid models such as CNN-LSTM by Petropoulos and Makridakis [7] and are also adept at identifying spatial patterns. Most of the above studies used publicly available datasets, for example, Google Mobility data, WHO databases, and the Johns Hopkins COVID-19 Repository. Such datasets can represent wide generalizability, but often do not have the granularity for a particular region.

Our choice of algorithms provides adaptability to a wide range of dataset characteristics, whether structured, such as linear regression, or unstructured, such as CNN. Additionally, some machine learning techniques, such as Random Forest and XGBoost, perform better in unbalanced datasets or noisy environments, which is consistent with the research conducted by Solayman et al. [12] and Yan et al. [6], where these algorithms outperformed more straightforward statistical approaches.

Our methodology also addresses a few common limitations present in the literature. For instance, we added the metrics of precision, recall, and F1-score to provide a more comprehensive evaluation of our models; most studies have only focused on accuracy metrics such as RMSE and MAE. Furthermore, the advantages of hybrid and ensemble models, such as the combination of CNNs and Random Forest for the analysis of spatial-temporal data, were employed to mitigate the challenges in long-term predictions by leveraging the strengths of hybrid and ensemble approaches.

The choice between ANN, XGBoost, Linear Regression, Random Forest, KNN, LSTM, and CNN reflects a balance necessary among the data in terms of simplicity and computational efficiency that are needed vis-à-vis predictive power. These algorithms include a number that are well-adapted to handling diverse data and are also flexible enough for both short-run and long-run prediction. Going ahead, this methodological approach can be supplemented by incorporating practical constraints, interpretable AI-based techniques, and ensemble methods further to enhance the interpretability and robustness of the predicted outcomes in changing environments.

3. Dataset Description

Table 2.
Description of Datasets.

No.	Feature Name	Abbreviation	Type	Measurement Unit
1	Date	DATE	Categorical	N/A
8	Day	DAY	Categorical	N/A
9	New Cases	CASES_NEW	Continuous	Count
10	Imported Cases	CASES_IMPORT	Continuous	Count
11	Recovered Cases	CASES_RECOVERED	Continuous	Count
12	Active Cases	CASES_ACTIVE	Continuous	Count
13	Cluster Cases	CASES_CLUSTER	Continuous	Count
14	Unvaccinated Cases	CASES_UNVAX	Continuous	Count
15	Partially Vaccinated Cases	CASES_PVAX	Continuous	Count
16	Fully Vaccinated Cases	CASES_FVAX	Continuous	Count
17	Booster Cases	CASES_BOOST	Continuous	Count
18	New Deaths	DEATHS_NEW	Continuous	Count
19	New Deaths DoD	DEATHS_NEW_DOD	Continuous	Count
20	Unvaccinated Deaths	DEATHS_UNVAX	Continuous	Count
21	Partially Vaccinated Deaths	DEATHS_PVAX	Continuous	Count
22	Fully Vaccinated Deaths	DEATHS_FVAX	Continuous	Count
23	Booster Deaths	DEATHS_BOOST	Continuous	Count
24	Holiday Indicator	HOLIDAY	Categorical	Binary (0 = No, 1 = Yes)

3.1. Exploratory Data Analysis (EDA)

In general, EDA and data visualization offer a wealth of detailed insights regarding the distribution of trends, correlations, and anomalies in this dataset. The creation of all these visualizations was developed using vis.py. The insight description into the key pattern of establishing data and the statistical relationship from the application of the statistical techniques adopted, aside from the detailed visualizations, also serves.

3.1.1. Data Quality Assessment

Since there are no missing values for any of the columns, imputation is not necessary, and hence, the data set is complete for analysis. However, some clear outliers can be seen in the case and death count-related variables such as "cases_new," "cases_import," "deaths_new," and vaccination-related features. These are critical occurrences, such as surges of COVID-19, which are relevant for understanding oscillations of the epidemic. Skewness studies indicated that most variables had a considerable positive skewness, especially for cases and deaths. For instance, some of the highest values included "cases_import" and "deaths_boost," with values 5.03 and 5.18, respectively, because their distribution contained very rare extreme data points. The low skewness and lack of outliers in time-based variables such as year, month, and day showed that the dataset was well distributed.

3.1.2. New Cases Over Time

A line plot was used to view the trend of new COVID-19 instances over time. The oscillations were observed as peaks and falls in the figure of the cases. These patterns provide insights into the evolution of the pandemic and could be linked with measures such as vaccination drives or lockdowns.

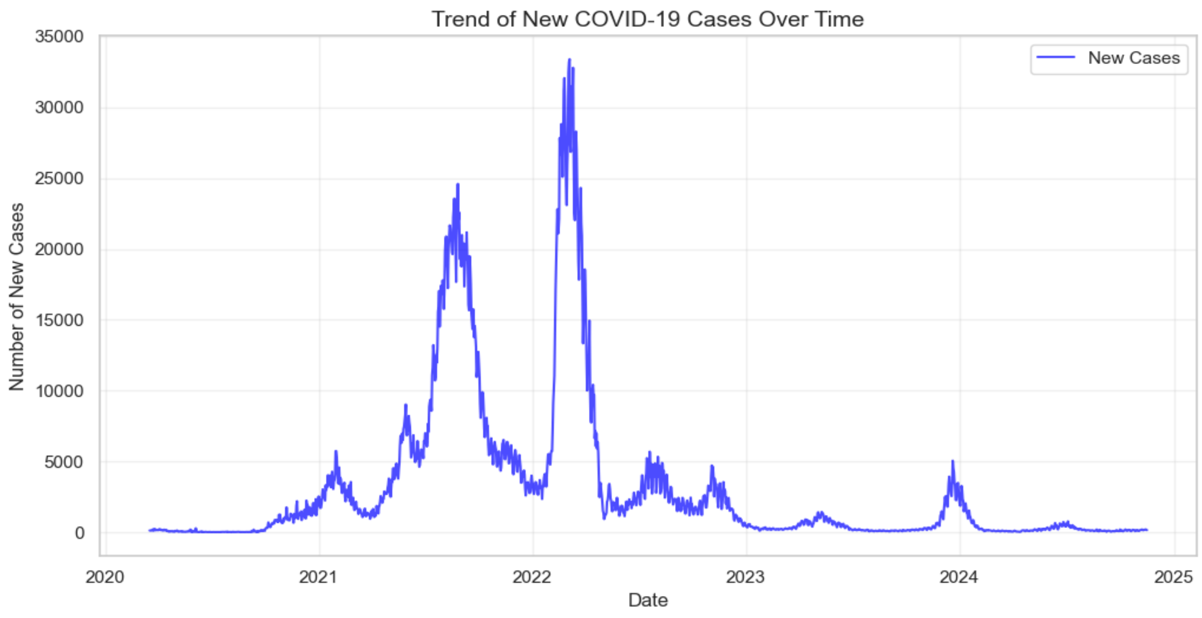


Figure 1.
New Cases Over Timeline Graph.

3.1.3. New Deaths Over Time

This line graph provides information on times of high mortality by showing the trend of COVID-19-related fatalities. It made it possible to comprehend the effects of the virus and the efficacy of interventions such as social seclusion and medical developments.

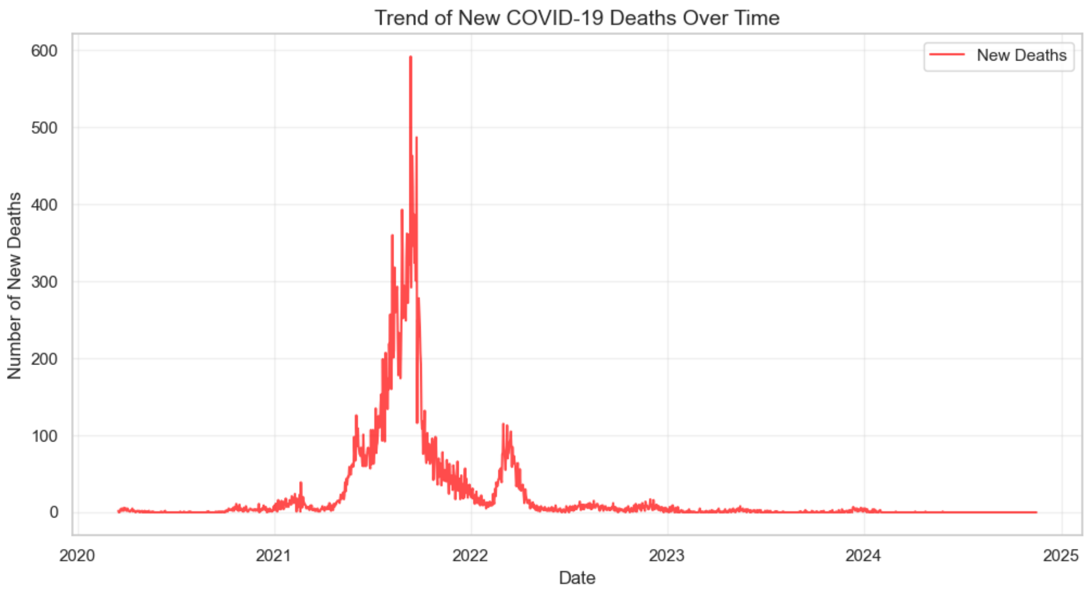


Figure 2.
New Deaths Over Timeline Graph.

3.1.4. Scatter Plot with Regression: Active Cases vs. Deaths

The association between active cases and deaths was depicted statistically using a scatter plot with a regression line, which offered insight into whether greater active case counts were associated with more fatalities. The seriousness of ongoing instances was clarified by this depiction.

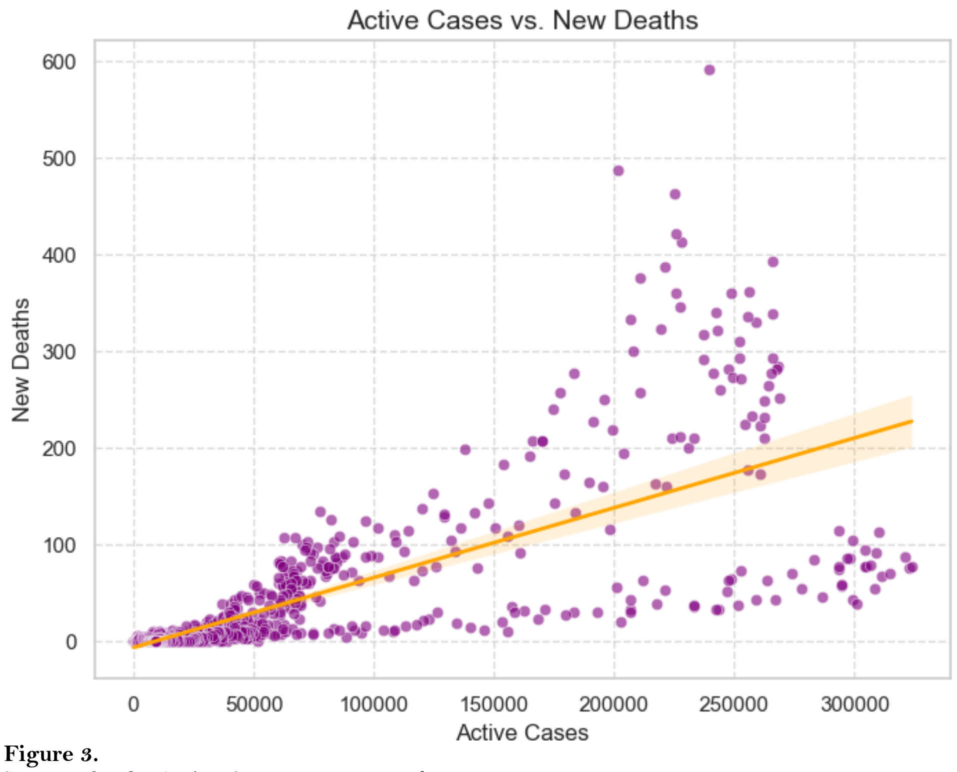


Figure 3.
Scatter Plot for Active Cases vs. New Deaths.

3.1.5. Cases on Weekends vs. Weekdays

This investigation contrasted the distribution of new cases on weekends and weekdays using boxplots and violin plots. The graphs showed possible variations in social behavior and case reporting, indicating either a larger or lower number of cases on particular days.

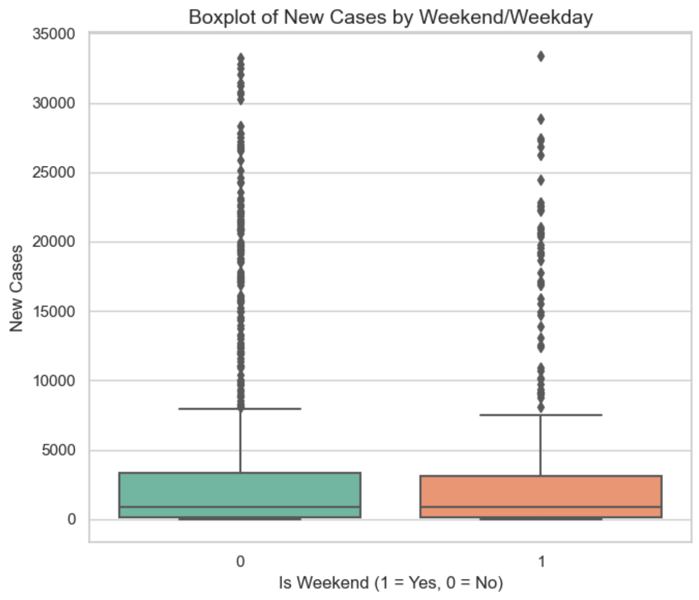


Figure 4.
Boxplot of New Cases by Weekend/Weekday.

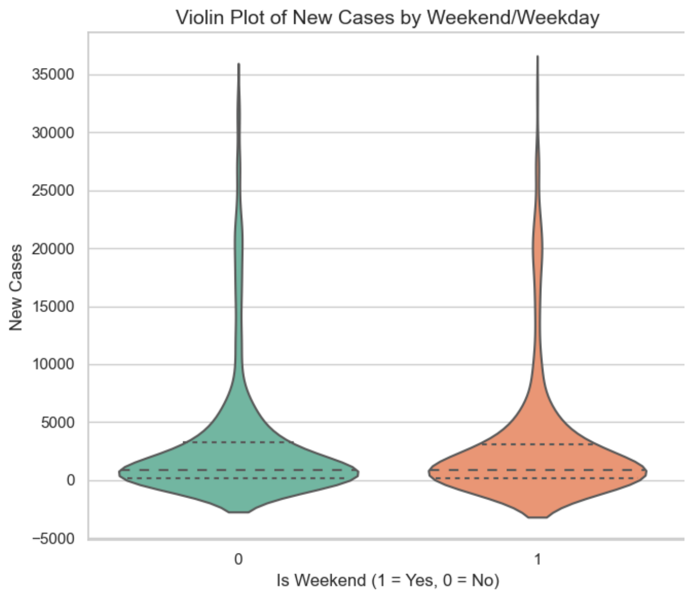


Figure 5.
Violin Plot of New Cases by Weekend/Weekday.

3.1.6. Recovery Rate Over Time

The recovery rate was monitored over time using a smoothed line plot. Key milestones, such as the 50% recovery rate, were highlighted in this graphic, which also showed when the pandemic recovery process was making progress or regressing.

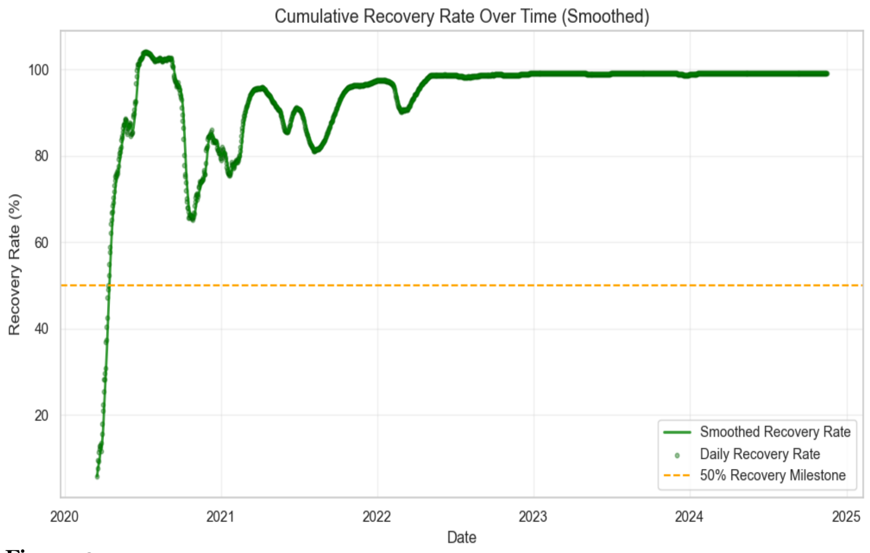


Figure 6.
Recovery Rate Over Timeline Plot.

3.1.7. New Cases Highlighting Holidays

Holidays were shown as red scatter dots on a line plot that displayed the trend of new instances over time. This graphic examined the relationship between holidays and increases in the number of cases, offering insights into how social behavior shifts during celebratory times.

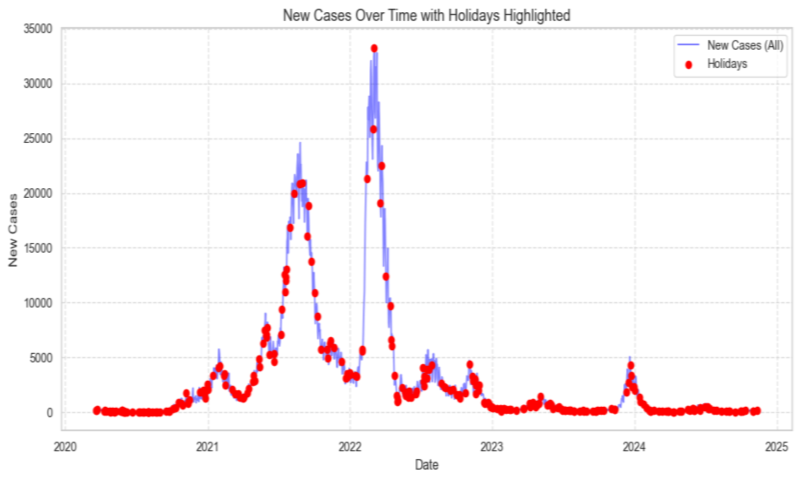


Figure 7.
New Cases Highlighting Holidays Line Plot.

3.1.8. Stacked Area Plot for Total Cases by Vaccination Status

A stacked area plot, broken down by vaccination status, showed the overall number of cases over time. This made it possible to comprehend how vaccination campaigns affected the distribution of cases and brought attention to patterns in immunization rates.

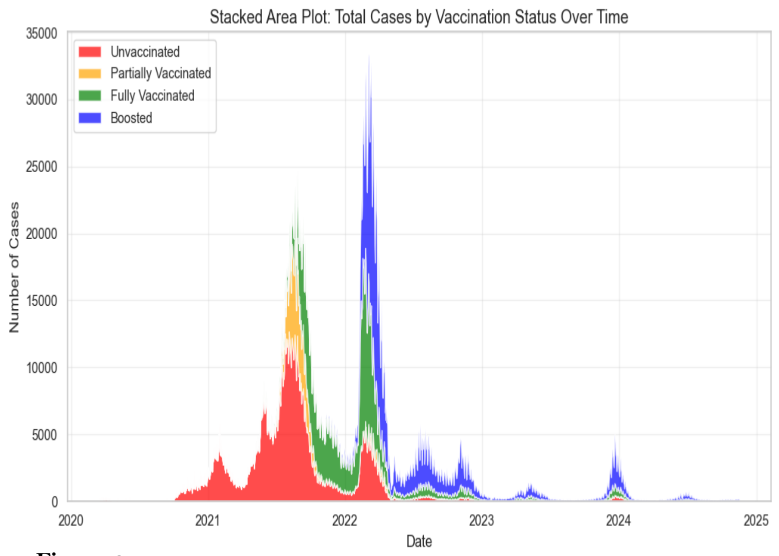


Figure 8.
Stacked Area Plot for Total Cases by Vaccination Status.

3.1.9. Pie Chart for Proportions of Cases by Vaccination Status

A straightforward but efficient method of comparing the prevalence of cases across various vaccination groups was provided via a pie chart that showed the relative proportions of cases by vaccination status.

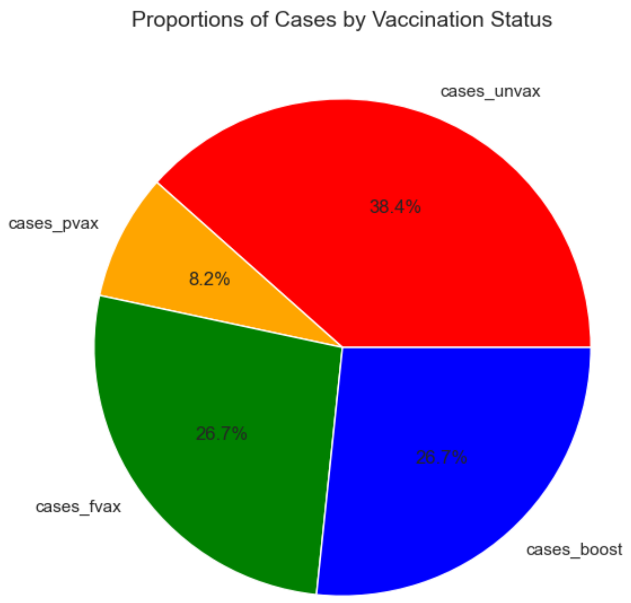


Figure 9.
Pie Chart for Proportions of Cases by Vaccination Status.

3.1.10. Recovery Rate by Significant Events

The recovery rate was monitored over time using a line plot annotated with prominent events, such as lockdowns and vaccination rollouts. In addition to highlighting the influence of significant interventions on recovery trends, this graphic included a historical background.

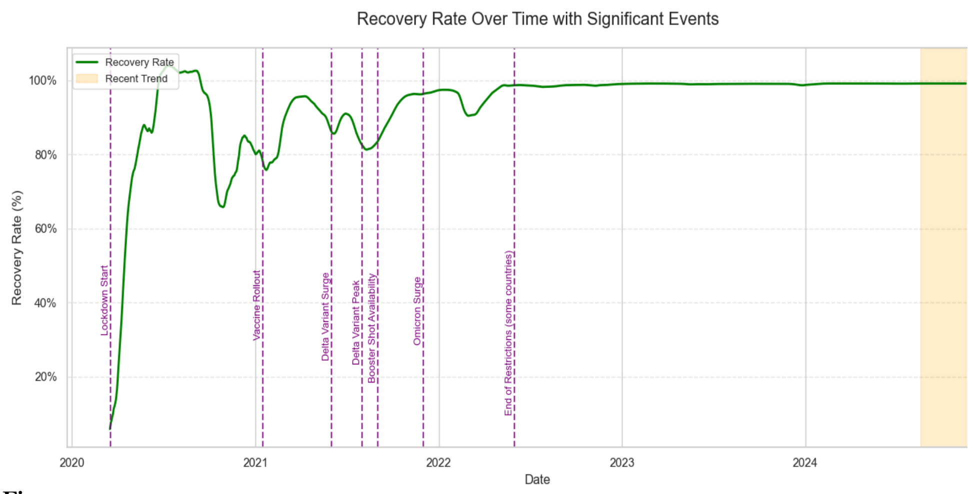


Figure 10.
Recovery Rate by Significant Events.

3.1.11. Death Rate Over Time with Significant Events

The death rate over time was also shown using a line plot that was annotated with significant occurrences. It made it possible to compare how mortality rates changed as a result of significant turning points, such as the introduction of new treatments or vaccination campaigns.

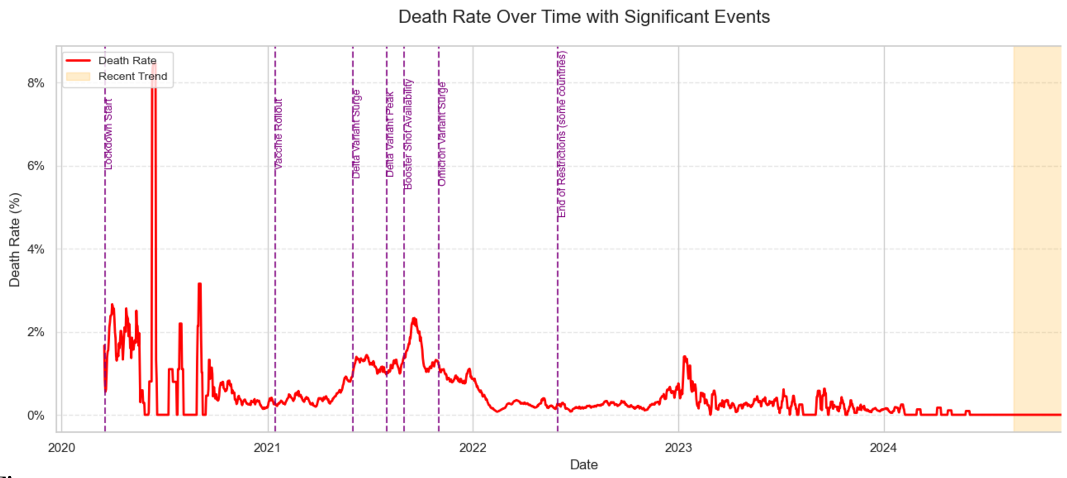


Figure 1.
Death Rate Over Time with Significant Events.

3.1.12. Statistical Insights and Anomalies

The dataset's case and death counts showed significant fluctuation, as indicated by the numerous outliers. For instance, "cases_new" contained 168 outliers, whereas "deaths_new" had 295 outliers. These outliers corresponded to known pandemic waves and were critical for understanding periods of rapid transmission and mortality. Skewness analysis revealed that vaccination-related variables, such as "cases_pvax" and "deaths_boost," exhibited significant skewness, indicating a concentration of values in lower ranges with sporadic severe cases.

Temporal variables such as year, month, and "week_of_year" showed no outliers and moderate skewness, indicating a consistent data distribution. However, the "is_weekend" variable has a modest skewness of 0.95, indicating an uneven distribution of cases across weekdays and weekends.

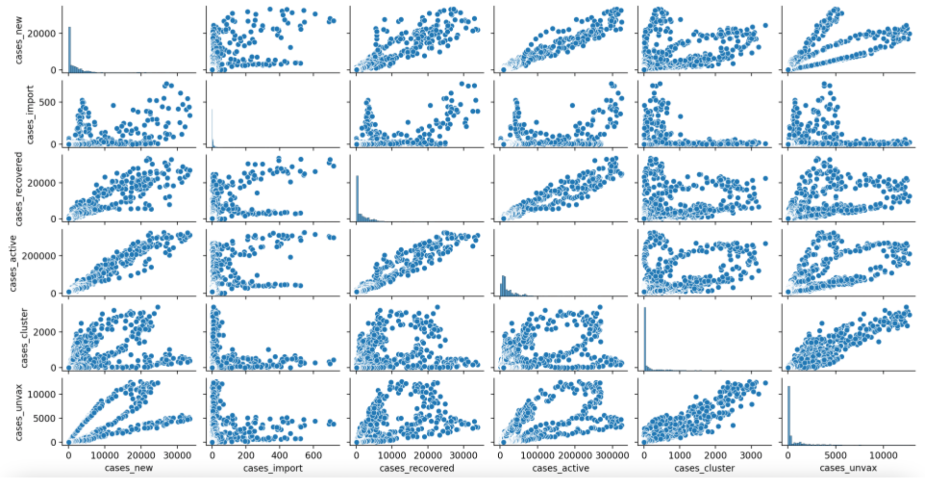


Figure 12.
Scatter Plot of Attributes.

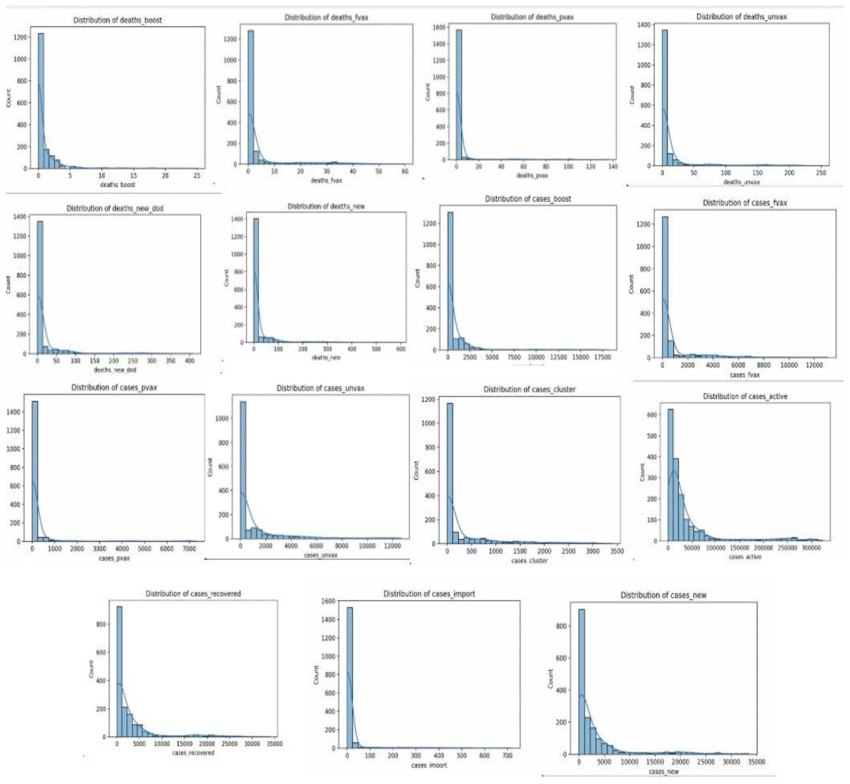


Figure 13.
Histogram of Attributes.

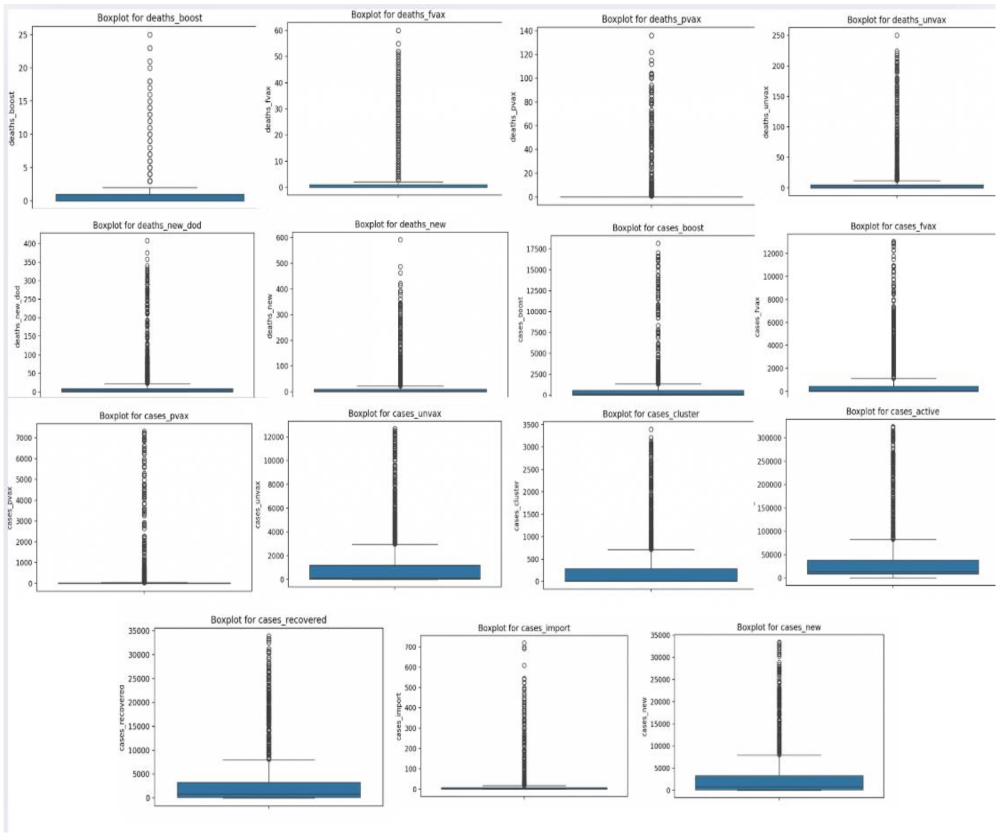


Figure 14.
Box Plot of Attributes.

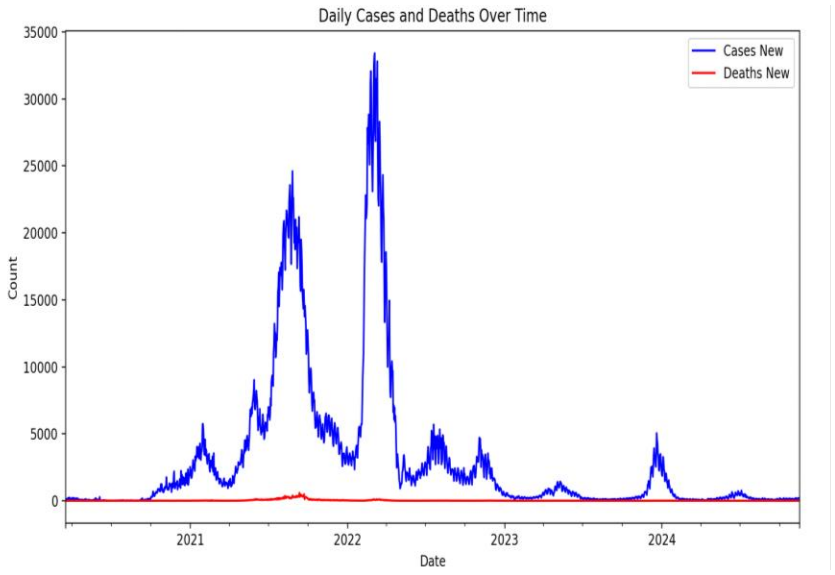


Figure 15.
Daily Cases and Deaths Over Timeline Graph.

In a nutshell, the dataset's exploration phase efficiently blends statistical analysis with a range of visualizations to find trends, correlations, and anomalies. The research provides a comprehensive assessment of the pandemic's course, including the recovery rate and fatality patterns, as well as the impact of vaccination status and key events. By finding essential patterns and linkages, this EDA lays the groundwork for future modelling and decision-making in understanding COVID-19 dynamics.

3.2. Data Pre-Processing

The dataset used in this analysis is sourced from data.gov.my and consists of three datasets: "cases by state," "cases by vaccination status," and "deaths due to COVID-19." The dataset initially included a state column, which displayed identical data repeated for each of the 14 states in Malaysia. Furthermore, the dataset included a final collection of data that represented Malaysia as a whole. To avoid disturbances when aligning and combining the datasets, we chose to keep only the rows that correspond to Malaysia in their entirety. This provided data consistency and eliminated redundancy caused by state-specific information. During the integration process, the state column was eliminated because it became unnecessary after just keeping Malaysia-wide data.

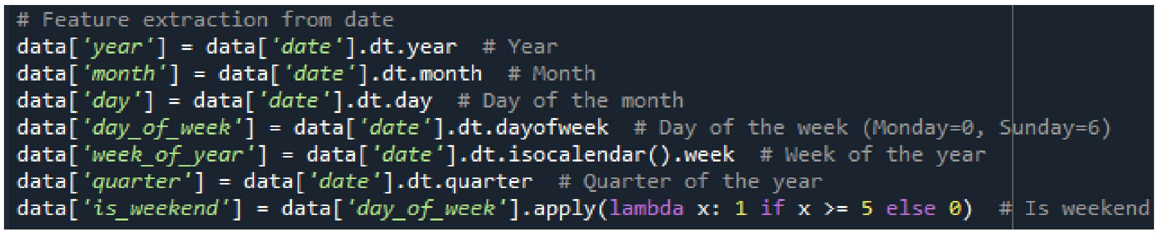
Additionally, the vaccination status and death attributes from the other two datasets were concatenated and added to the dataset to improve the analysis. The start and end dates of each dataset were aligned to maintain consistency, and the combined dataset contained 1,707 rows of data.

3.2.1. Holiday

To improve the dataset, a holiday attribute was added, with a value of 1 denoting public holidays and 0 representing non-public holidays. The holiday data was collected from publicholidays.com.my, which provided official public holiday dates for Malaysia across a wide range of time. This addition aimed to study the potential effect of vacations on the number of new COVID-19 cases.

3.2.2. Date

Furthermore, feature extraction was applied to the date column to generate new temporal attributes that could provide insights into trends over time. A script called date_adjust.py was created to extract different attributes from the date column. These characteristics included the year, month, and day of the month, as well as the day of the week (Monday is 0 and Sunday is 6), the week of the year, the quarter of the year, and a binary indicator for weekends. The feature extraction was carried out as follows:



```
# Feature extraction from date
data['year'] = data['date'].dt.year  # Year
data['month'] = data['date'].dt.month  # Month
data['day'] = data['date'].dt.day  # Day of the month
data['day_of_week'] = data['date'].dt.dayofweek  # Day of the week (Monday=0, Sunday=6)
data['week_of_year'] = data['date'].dt.isocalendar().week  # Week of the year
data['quarter'] = data['date'].dt.quarter  # Quarter of the year
data['is_weekend'] = data['day_of_week'].apply(lambda x: 1 if x >= 5 else 0)  # Is weekend
```

Figure 16.
Code Snippet of Date Feature Extraction.

These preprocessing methods guaranteed that the dataset was completely ready for analysis, with an emphasis on temporal aspects and external factors such as vacations that could influence COVID-19 case trends.

3.3. Data Cleaning

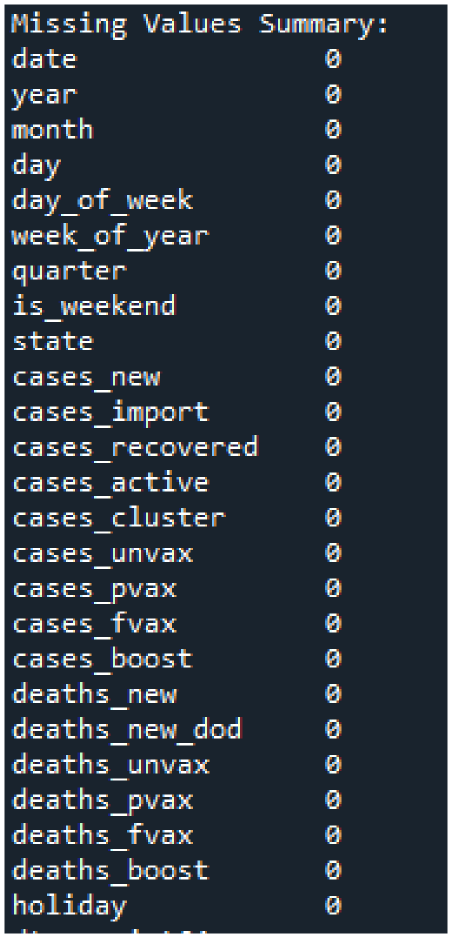
Various visualization techniques and statistical checks to verify that our dataset does not require preprocessing.

Table 1.
Dataset Missing Values, Outliers & Skewness.

Column	Missing Values	Outliers Detected	Skewness
year	0	0	-0.02
month	0	0	-0.04
day	0	0	0.01
day_of_week	0	0	0.00
week_of_year	0	0	-0.04
quarter	0	0	-0.04
is_weekend	0	0	0.95
cases_new	0	168	2.89
cases_import	0	225	5.03
cases_recovered	0	164	2.89
cases_active	0	182	2.78
cases_cluster	0	272	2.49
cases_unvax	0	236	2.78
cases_pvax	0	294	5.09
cases_fvax	0	267	3.37
cases_boost	0	272	4.72
deaths_new	0	295	4.42
deaths_new_dod	0	297	3.75
deaths_unvax	0	302	3.76
deaths_pvax	0	310	5.05
deaths_fvax	0	348	2.79
deaths_boost	0	182	5.18
holiday	0	278	1.83

3.3.1. Missing Values

First, we assessed missing values by generating a heatmap using Seaborn's heatmap function and calculating the count of missing values per feature with Python's isnull().sum(). The heatmap showed no highlighted cells, and the summary table confirmed zero missing values across all features.



```
Missing Values Summary:
date                  0
year                  0
month                 0
day                   0
day_of_week           0
week_of_year          0
quarter               0
is_weekend            0
state                 0
cases_new             0
cases_import          0
cases_recovered       0
cases_active          0
cases_cluster         0
cases_unvax           0
cases_pvax            0
cases_fvax            0
cases_boost           0
deaths_new            0
deaths_new_dod        0
deaths_unvax          0
deaths_pvax           0
deaths_fvax           0
deaths_boost          0
holiday               0
```

Figure 17.
Summary Table of Missing Values.

3.3.2. Outliers

Next, we evaluated outliers in the numerical features using box plots created with Matplotlib. The outliers in the case-related attributes, such as cases_new, cases_import, cases_recovered, and others, were identified using box plots, which visually highlight data points outside the whiskers (1.5 times the interquartile range). These outliers likely correspond to extreme values observed during the peak periods of the COVID-19 pandemic, such as surges in cases due to new variants, major outbreaks, or vaccination campaigns. While these outliers are significant, they are not indicative of errors or anomalies but rather reflect real-world events associated with the pandemic's unpredictable nature. For example, the high number of cases or deaths during critical moments of the pandemic would naturally result in extreme values that fall outside the usual range, and these should be viewed as part of the dataset's natural variability. As such, these outliers are justified and should not be removed, as they accurately represent the data during a time of unprecedented global health crisis.

3.3.3. Skewness

The dataset reveals significant skewness in several variables related to COVID-19 cases and deaths, with skewness values consistently above 1, indicating a right-skewed distribution.

Specifically, the columns cases_import, cases_pvax, cases_boost, deaths_new, and deaths_boost exhibit extreme skewness values, ranging from 4.42 to 5.18. This high skewness is likely a result of the pandemic's exceptional nature, particularly during the peak periods when the number of cases and deaths soared dramatically. During these times, a large number of cases and deaths occurred in a relatively short time span, leading to an overrepresentation of high values in the dataset and contributing to the right-skewed distribution.

The skewness observed in these columns is justified by the pandemic's irregular and unpredictable progression. For example, cases_import may have shown a sharp increase in imported cases during the height of international travel restrictions and virus spread, while cases_pvax and cases_boost would have spiked as vaccination campaigns were rolled out at varying speeds across different regions. Similarly, the deaths_new and deaths_boost columns reflect a surge in mortality rates, especially at the peak of the pandemic, where large numbers of deaths were reported within short time intervals.

Given the nature of these data, the skewness is not indicative of data quality issues but rather reflects the exceptional circumstances of the pandemic. Therefore, while the skewness is pronounced, it is a natural characteristic of the dataset and should be interpreted in the context of the pandemic's peak periods.

In summary, after analyzing the dataset, it was found that no preprocessing was necessary. First, there were no missing values across any of the attributes, which indicated that the dataset was already complete and did not require imputation or handling of null values. Secondly, the dataset exhibited skewness in several columns, especially those related to COVID-19 cases and deaths, with skewness values ranging from 2.49 to 5.18. However, this skewness is expected due to the exceptional nature of the pandemic, where there were significant spikes in cases and deaths, especially during peak periods.

Lastly, the outliers detected through box plots were justified as they correspond to real-world events during the pandemic, such as surges in cases and deaths, and therefore do not require removal or correction. Consequently, given the absence of missing values, the natural skewness, and the justified presence of outliers, no further preprocessing was performed on the dataset after its creation.

3.4. Feature Selection

3.4.1. Correlation Matrix

The correlation matrix is a widely used technique for feature selection, as it quantifies the linear relationships between features and the target variable. This method involves calculating pairwise Pearson's correlation coefficients, which measure the strength and direction of the linear relationship between variables. A correlation heatmap is often used to visualize these relationships, making it easier to identify patterns and redundancies in the data.

To use this strategy, first compute the correlation matrix for all feature pairings, including the target variable. Pearson's correlation coefficient values range from -1 to +1, where values close to +1 or -1 indicate a strong positive or negative linear relationship, respectively. Values near 0, on the other hand, suggest no linear relationship. Once the correlation matrix is computed, it can be interpreted to guide feature selection. Features that show a strong correlation with the target variable, those with absolute correlation coefficients above a predefined threshold, such as $|0.3|$, should be retained as they are likely to contribute to predictive performance.

At the same time, the matrix can highlight issues of multicollinearity, where two features are highly correlated with each other ($|\text{correlation}| > 0.85$). Multicollinearity can introduce redundancy, as one feature may not add significant new information. In such cases, one of the correlated features should be removed to simplify the model and reduce overfitting.

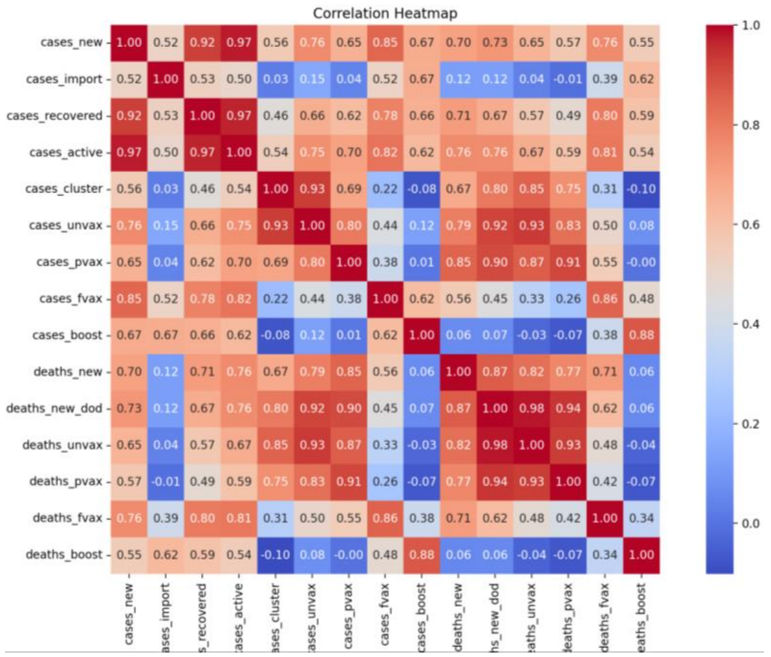


Figure 2.
Correlation Heatmap for Feature Selection.

The correlation matrix method is quick and straightforward to implement, making it an excellent starting point for feature selection. It helps identify linear relationships between features and the target while also revealing redundant features. For example, in the heatmap provided, features with low correlation to the target variable could be safely excluded, as they are unlikely to contribute to the model's performance. Additionally, highly correlated features (observed as off-diagonal high correlations) should be addressed by retaining only one of them.

By streamlining the dataset, the correlation matrix improves model efficiency and interpretability, paving the way for more advanced feature selection techniques or model-building efforts.

3.4.2. Feature Importance Analysis using XGBoost

Feature importance analysis offers a method to determine the contribution of each input feature to the prediction of COVID-19 in this scenario. In XGBoost, this involves evaluating how much each input feature influences the model's performance during training. During the training phase of the model, XGBoost constructs a series of decision trees sequentially, with each attempting to correct the errors from the previous ones. The model starts by splitting the data based on feature values to reduce the overall loss function. Subsequently, feature importance is computed through methods such as gain, weight, and coverage. During these processes, the improvement is measured by observing the model's loss function due to the split involving a feature. This step indicates each feature's contribution to reducing prediction errors. Then, the importance of the features can be assessed by analyzing the total number of times a feature is used in splits across all the trees. The scores of each input feature are normalized and sum up to 1.

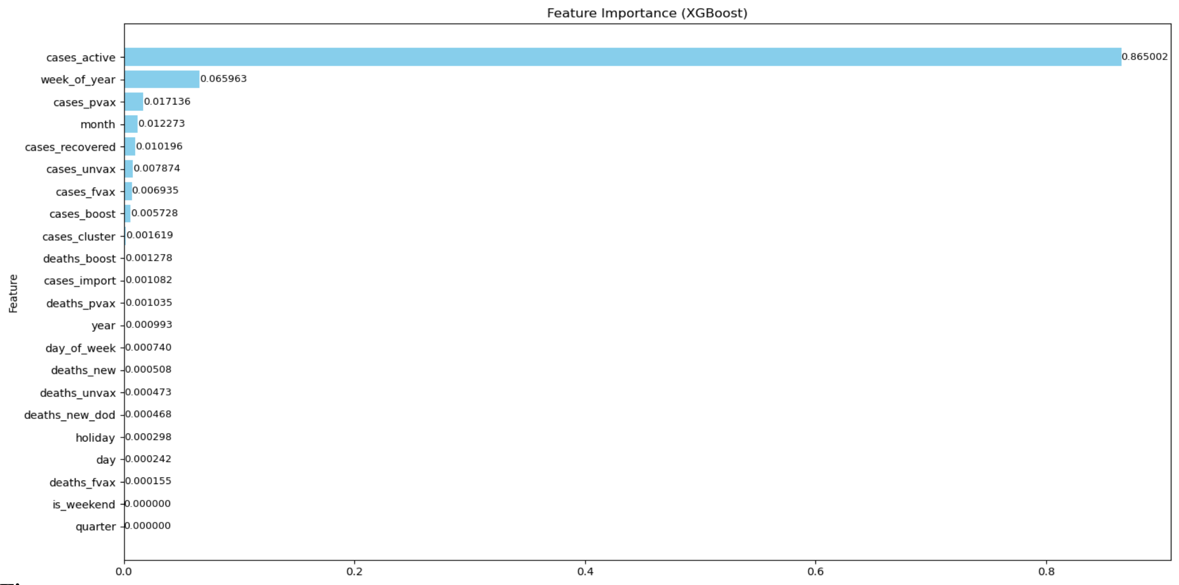


Figure 19.
Feature Importance Bar Chart using XGBoost

XGBoost is chosen to determine the importance of each input feature as it is a gradient-boosting algorithm well-known for its speed and accuracy, allowing better analysis of complex datasets. Other than that, this algorithm helps to provide insights into which variables influence the prediction the most, which aligns with the goal of understanding the key drivers of the target variable, in this case, “cases_new”. The uniqueness of XGBoost is that it can handle non-linear relationships and interactions between common features in real-world datasets, making it more straightforward to extract and visualize feature importance while saving time in model interpretation.

3.4.3. Feature Importance Analysis using ANNs

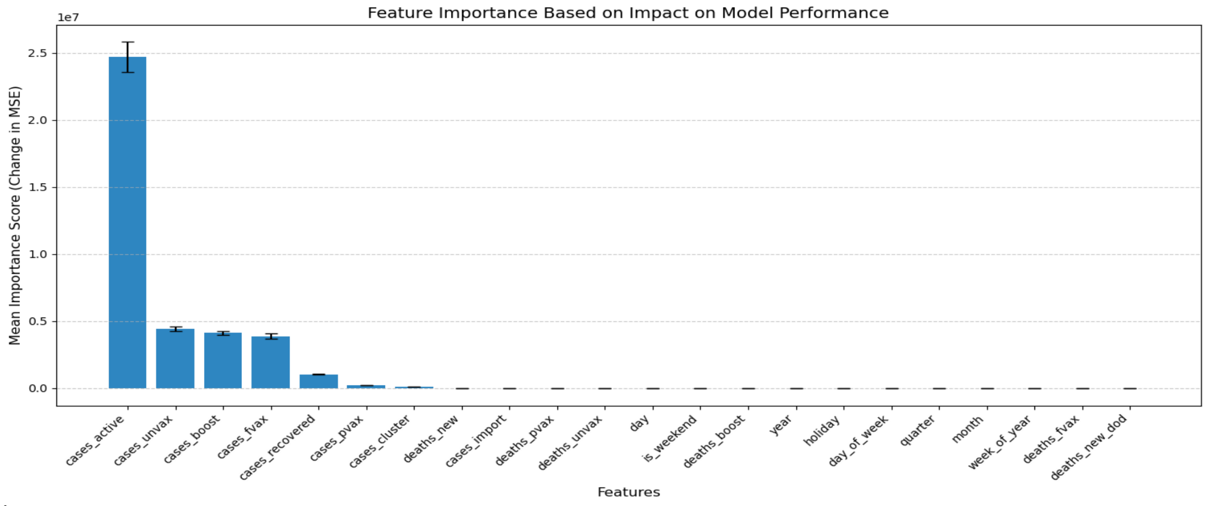


Figure 3.
Feature Importance Bar Chart using ANNs.

Artificial Neural Networks (ANNs) can also be used to determine the feature importance of the dataset due to their advantages in capturing complex and non-linear relationships in the data. Methods such as permutation importance provide a way to assess the relevance of each variable even though ANNs function as black boxes. The permutation importance works by evaluating the model's performance and determining how much it deteriorates when the value of a single feature is shuffled. This strategy is excellent for artificial neural networks because they can understand the complicated connections between features and do not require any direct interpretation of weights, which can be difficult in non-linear models. Despite the intricacy of their internal workings, ANNs can efficiently identify the features that contribute the most to the model's prediction by recording feature patterns and interactions.

3.4.4. Feature Importance Analysis using Random Forest

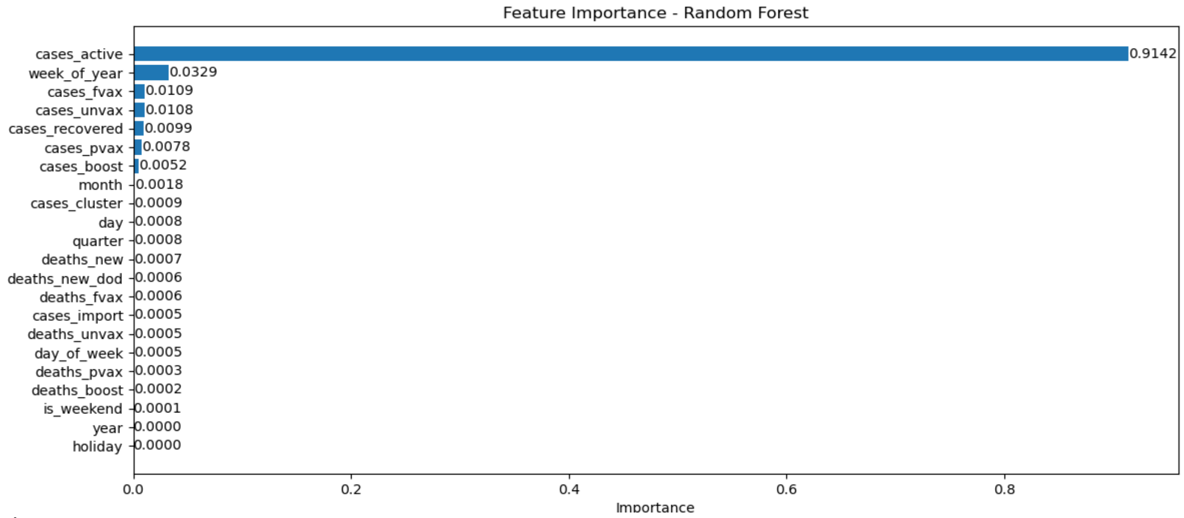


Figure 21.
Feature Importance Bar Chart using Random Forest.

Random Forest is a type of algorithm that can be implemented to determine the feature importance of each variable due to its ability to model complex relationships in data while providing a straightforward measure of feature relevance. This algorithm functions by constructing various decision trees during training and aggregating their predictions. In this context, it computes the importance of each feature by evaluating how much the feature reduces impurity across all trees in the forest. This step is performed by observing the decrease in model performance when the values of a certain feature are removed. Additionally, Random Forest is useful in capturing both linear and non-linear relationships, making it relatively robust to overfitting issues, especially when used with a large number of trees.

3.5. Architecture Diagram

The method begins with data collection, which involves gathering raw datasets on COVID-19 cases, vaccination status, and mortality from credible sources. This stage lays the groundwork for future analysis by ensuring that all necessary data is available for processing. The Datasets Integration & Alignment phase combines numerous datasets into a cohesive form. This entails matching data properties, such as date formats, to maintain uniformity. By addressing conflicts and condensing data, a cohesive dataset is created for the next stages. Next, Feature Engineering is implemented to enrich the dataset with more meaningful variables. The key steps are adding a binary "holiday" property (1 for holidays, 0 for non-holidays) and extracting date-related information such as “year”, “month”, “day”, “day_of_week”, “week_of_year”, “quarter”, and “is_weekend”. These features provide temporal context for deeper analysis.

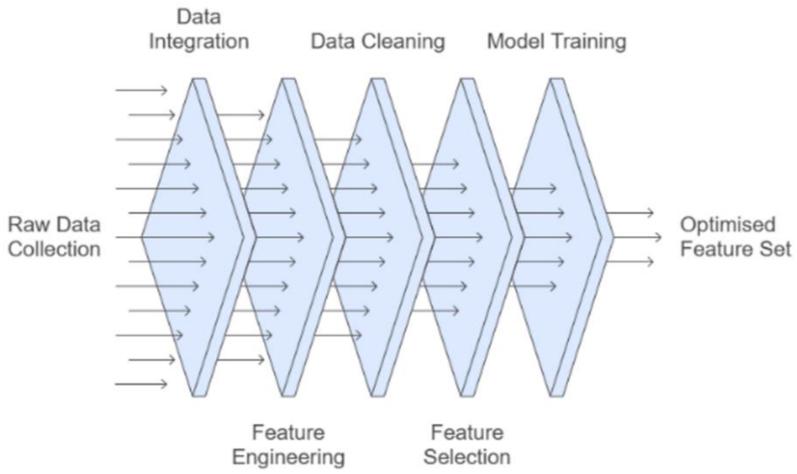


Figure 22.
Architecture Diagram of the Project.

The Data Cleaning Process ensures that the dataset is free of errors, missing values, and unnecessary columns. This stage removes superfluous features, such as state, and verifies data integrity through exploratory tests. The end result is a refined dataset that is ready for model training. With a clean dataset, the Model Training & Prediction stage uses machine learning models to forecast the target variable (e.g., cases_new). The initial models are trained with all available features to assess performance and identify areas for improvement.

The Evaluate stage assesses model performance using metrics such as R-squared or mean squared error. This evaluation guides the next step, Feature Selection, which employs Iterative Recursive Feature Elimination (RFE). Low-importance or redundant features are systematically removed during retraining models to refine the feature set. Finally, during the Feature Refinement step, the iterative feature selection process is completed, and only the most relevant features are retained. This results in the Final Dataset, which is optimized for accuracy and efficiency in predictive modeling.

3.6. Training and Testing Set Overview

The total of observations: 1706
The number of features or variables: 17

By using the 80:20 split, the training set will consist of 1,364 observations, whereas the testing set will consist of 342 observations.

Table 4.
Training and Testing Dataset.

Dataset	Number of observations	Number of features
Training set	1364	17
Testing set	342	17

4. Methodology

4.1. Artificial Neural Networks (ANNs)

Artificial Neural Networks (ANNs) were chosen as the principal approach for forecasting daily new COVID-19 cases. Inspired by the structure of the human brain, ANNs are composed of interconnected layers of neurons that analyze data using weighted connections and activation functions. They are especially well-suited for modeling complex, non-linear interactions and working with datasets that contain a variety of features. This adaptability makes them ideal for the dataset, which includes temporal, categorical, and numerical information. ANNs also excel in regression problems with continuous target variables, such as "cases_new," enabling more accurate forecasts.

4.1.1. Justification

ANNs were chosen for their capacity to generalize and extract detailed patterns from the dataset. The non-linear modeling capabilities, scalability, feature integration, and strong regularization approaches are the most important aspects affecting this decision. ANNs incorporate non-linearity into the data through activation functions such as ReLU, allowing them to capture complex patterns and dependencies. Their scalability provides effective computing for datasets of different sizes and complexity. Furthermore, ANNs naturally learn feature importance during training, eliminating the requirement for explicit feature engineering.

4.1.2. Strengths

Artificial neural networks (ANNs) have various advantages that make them effective machine learning tools. They are extremely adaptable, capable of modeling both linear and non-linear connections within data, increasing their usefulness across a wide range of applications. ANNs also perform autonomous feature learning during training, detecting complicated feature interactions without the need for considerable preprocessing. Their scalability enables them to process enormous datasets efficiently, especially when supplied with many layers and neurons. Furthermore, strong regularization techniques such as dropout and early halting are used to improve generalization and avoid overfitting.

4.1.3. Weaknesses

Despite their advantages, artificial neural networks (ANNs) have certain limitations. Training these networks can be computationally expensive, especially for deep architectures and large datasets, necessitating substantial resources. Because ANNs are more sophisticated than simpler models, it is difficult to understand how they arrive at precise predictions. Furthermore, ANNs are hyperparameter sensitive; obtaining peak performance involves precise adjustment of parameters such as learning rate, number of layers, and number of neurons.

4.1.4. Architecture Overview and Working Mechanism

This project's ANN design consists of an input layer, three hidden layers, and one output layer. The input layer accepts normalized features from the dataset, with each input representing a feature. The hidden layers include a first layer with 128 neurons and ReLU activation to capture complex associations, a second layer with 64 neurons and ReLU activation to refine learned patterns, and a third layer with 32 neurons. The output layer, which consists of a single neuron with a linear activation function, returns the predicted value for "cases_new".

The working mechanism consists of a feedforward pass in which input features are multiplied by weights, summed with biases, and passed through activation functions. Outputs from one layer are used as inputs for the next. The Mean Squared Error (MSE) loss function is used to compute the difference between anticipated and actual values for the log-transformed target variable. Backpropagation computes the gradients of the loss function, and the Adam optimizer adjusts the weights to minimize the loss. The model is trained across 20 epochs to achieve a balance between computational efficiency and model learning. It also employs mini-batches of size 32 to promote effective learning. The model's prediction accuracy and explanatory capacity are evaluated using measures such as RMSE, MAE, and R^2 (Coefficient of Determination).

As a result, ANNs were chosen for their capacity to model nonlinear connections and successfully handle heterogeneous data sources. The design, which includes hidden layers with ReLU activations and strong regularization algorithms, allows the ANN to generalize patterns in data while limiting overfitting. Despite their computational cost and interpretability problems, ANNs are an effective tool for regression tasks such as forecasting "cases_new," providing accurate and actionable forecasts.

4.2. Convolutional Neural Network (CNNs)

Convolutional Neural Networks (CNNs) are deep learning models designed to analyze structured grid-like input such as pictures, time series data, and geographical data. They are regarded as one of the most well-known machine learning tools, especially for tasks requiring spatial or sequential data patterns. Furthermore, CNNs can learn directly from raw data, making them extremely fast and reliable in forecasting COVID-19 instances based on prior data.

4.2.1. Justification

CNNs were chosen based on the dataset size of 1706 observations, which is adequate to train a CNN model. This algorithm is functional with large datasets and can handle the complexity of the data better than simpler models. Even though CNNs are usually used for image classification tasks, they are also applicable to time-series data like COVID-19 case numbers over time. Next, the dataset consists of both continuous and categorical data. CNNs can handle continuous data efficiently by learning the spatial or even temporal patterns from the data values.

On the other hand, CNNs can be adapted for categorical data by utilizing embeddings, enabling the prediction model to process and learn from both types of features simultaneously. Moving on, in this scenario, the target variable is continuous data, and the goal is to predict future values of COVID-19 cases based on historical trends. CNNs are equally effective in solving regression problems when properly adapted. Furthermore, CNNs are capable of learning patterns and relationships in the data, such as seasonality and fluctuations in the spread of the virus, resulting in more accurate predictions of the continuous variable.

4.2.2. Strengths

CNNs are capable of learning complex spatial and temporal pattern data due to their architectural design, which includes three main features: convolutional layers, activation functions, and pooling mechanisms. The number of cases in the dataset is crucial in COVID-19 prediction, as it often involves complex trends, seasonality, and shifts over time. Furthermore, this algorithm can effectively capture all the temporal dependencies in time-series data, making it more suitable for time-series forecasting tasks such as predicting upcoming COVID-19 cases. CNNs process data by breaking it into several overlapping windows, each containing a certain temporal context. They analyze these windows to learn the relationships between past and future values.

4.2.3. Weaknesses

CNNs are designed to capture local patterns within a certain receptive field. Although they excel in detecting short-term dependencies in time-series data, they still struggle with long-term dependencies. In this case, CNNs might be unable to account for the influence of distant past events without significant adjustments, resulting in low accuracy of prediction for COVID-19 cases. Next, CNNs are a type of algorithm that requires significant resources and a large volume of data for training to avoid issues like overfitting. In cases where the dataset is noisy, it may fail to generalize well to upcoming new data. For instance, small regional datasets for COVID-19 with irregular reporting might not provide enough information for CNNs to learn and predict the cases properly.

4.2.4. Architecture Overview and Working Mechanism

CNNs are designed to operate by applying numerous filters (convolutions) to input data to detect local patterns and relationships, which are then passed through multiple layers to capture increasingly abstract features. The layers work by using convolutional layers to filter the input data, hierarchically sliding across the data to detect local variables. In current time-series data, these filters can capture all the important temporal patterns, such as spikes in COVID-19 cases or interactions between variables. Moving on, an activation function such as Rectified Linear Unit (ReLU) is then applied. This non-linearity enables the network to learn and adapt to complex patterns beyond linear relationships, allowing it to capture more distinctive trends in the dataset.

The existing pooling layers in the CNNs help to reduce the data's dimensionality while preserving all the crucial information. This step helps to minimize the computational load needed and emphasizes high-level patterns such as long-term trends in COVID-19 cases. After the convolution and pooling layers, the data is passed through fully connected layers where each neuron is connected to every neuron in the previous layer. These layers allow the network to combine the features learned from earlier layers to produce more complex predictions. Lastly, the output layer will consist of a single neuron to produce a continuous value, which is the number of cases for the next time period.

In conclusion, CNNs are a powerful method for predicting COVID-19 situations because they can successfully capture the temporal and geographical correlations in the data. They are effective in predicting future COVID-19 cases by capturing sequential or geographical patterns in data and identifying all trends, such as case peaks.

4.3. K-Nearest Neighbours (KNN)

The K-Nearest Neighbours (KNN) algorithm was selected as an alternative approach for predicting daily new COVID-19 cases (**cases_new**). KNN is a simple yet effective machine learning method that predicts target values based on the similarity between data points. By considering the closest **k** data points in feature space, KNN performs classification or regression tasks without relying on an explicit model structure. This instance-based learning technique is particularly well-suited for smaller datasets or datasets where local patterns in the data hold significant predictive power. Its interpretability and ability to work with mixed feature types make it a viable choice for the given dataset, which includes temporal, categorical, and numerical features.

4.3.1. Justification

The K-Nearest Neighbours (KNN) algorithm was used as an alternative method for forecasting daily new COVID-19 instances (cases_new). KNN is a basic yet effective machine learning technique that predicts target values based on the similarity of input points. KNN solves classification or regression tasks without the need for an explicit model structure by taking into account the nearest k data points in feature space. This instance-based learning technique is particularly well-suited for smaller datasets or datasets where local patterns in the data hold significant predictive power. Its interpretability and ability to work with mixed feature types make it a viable choice for the given dataset, which includes temporal, categorical, and numerical features.

4.3.2. Strengths

KNN has various characteristics that make it a viable option for certain regression problems. It is non-parametric, which means it makes no assumptions about the underlying data distribution, allowing it to deal with non-linear interactions efficiently. The technique is straightforward to build and understand since predictions are based on the average or majority class of the nearest neighbors. Furthermore, KNN adjusts to local patterns by focusing solely on neighboring data points in the feature space, requiring less preparation and working directly with normalized input.

4.3.3. Weaknesses

KNN also has limitations. Its prediction phase can be computationally costly for large datasets since distances must be calculated for each data point during inference. It has difficulty with high-dimensional data because of the "curse of dimensionality," which decreases the efficiency of distance-based measurements. KNN is also extremely sensitive to feature scaling, necessitating rigorous normalization or standardization to avoid inaccurate distance computations. Furthermore, the algorithm's performance is significantly reliant on hyperparameters such as the k value and the distance metric, which must be optimized through experimentation.

4.3.4. Architecture Overview and Working Mechanism

The KNN implementation for this project entails determining the best value of k (number of neighbors) and an appropriate distance measure. Following data normalization, the method predicts “cases_new” for a given input by locating the k nearest neighbors in the feature space and calculating the weighted or unweighted average of their target values. The working mechanism is summarized as follows: features are normalized to ensure that all dimensions contribute equally to distance calculations, distances between test points and all training points are calculated, the k nearest neighbors are chosen based on the shortest distances, and the predicted value for “cases_new” is calculated as the mean of these neighbors’ target values. Hyperparameter optimization, including tuning the value of **k**, is performed using cross-validation to minimize prediction error.

Finally, KNN was chosen because of its simplicity and interpretability in forecasting new COVID-19 instances on a daily basis. Its ability to capture local patterns without depending on sophisticated mathematical frameworks makes it a strong candidate for datasets with non-linear correlations. While it confronts obstacles such as computational expense and scale sensitivity, effective preprocessing and hyperparameter optimization can help to alleviate these issues. KNN provides reliable and interpretable results for regression tasks such as forecasting "cases_new" by utilizing nearest neighbors.

4.4. Linear Regression

Linear regression is a fundamental algorithm for machine learning and statistical modeling. It was chosen as a baseline model for this problem because it is simple, interpretable, and appropriate for regression tasks. The approach assumes a linear connection between the dependent variable (cases_new) and the independent variables, making it an appropriate initial step in modeling the data.

4.4.1. Justification

Linear regression was chosen due to its computing efficiency and interpretability, making it an excellent baseline model for this work. It scales well to small and medium-sized datasets, guaranteeing quick computation with minimal cost, even when dealing with a substantial number of features. The dataset includes both continuous variables, such as cases_active and cases_recovered, and categorical variables, such as one-hot-encoded holiday variables. Linear regression may handle such data well if categorical variables are appropriately represented. This approach, designed primarily for regression issues, aims to forecast a continuous target variable, cases_new, by minimizing the mean squared error

(MSE) and determining the best-fitting line. Its simplicity and suitability for the provided data and issue type made it an obvious decision.

4.4.2. Strengths

Linear regression has numerous major advantages that make it an appropriate solution for our task. It is easily interpretable since the coefficients immediately indicate the link between each characteristic and the target variable. This transparency enables stakeholders to simply grasp how the model generates predictions. Furthermore, it is computationally inexpensive, requiring minimal resources and training quickly, making it an ideal starting point for exploratory research. Linear regression serves as a baseline model, providing a point of comparison for evaluating the performance of more sophisticated models. Additionally, the model sheds light on the relevance of features, with each coefficient representing the size and direction of the association between a characteristic and the target variable.

4.4.3. Weaknesses

Despite its advantages, Linear Regression has certain drawbacks. One important disadvantage is that it may underperform if the characteristics' relationship to the target variable is nonlinear. In this instance, more complicated models may be required to reflect the real underlying patterns. Furthermore, Linear Regression is susceptible to multicollinearity, in which highly linked features might skew the model's coefficient estimations. This makes it difficult to identify each feature's genuine contribution. Additionally, the model is sensitive to outliers, which can have a disproportionate impact on the results, making it less resilient in cases involving extreme values in the data.

4.4.4. Architecture Overview and Working Mechanism

Linear Regression presupposes a linear connection between the input characteristics and the target variable. The model is described by this equation:

$$y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_n X_n + \epsilon \quad (1)$$

Where:

y: Predicted value of **cases_new**

β_0: Intercept of the regression line

β_i: Coefficients for each feature

ϵ: Residual error (difference between actual and predicted values)

$$\text{Cost Function: } J(\beta) = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 \quad (2)$$

Where:

y_i : Actual value of cases_new

$\hat{y}_i$: Predicted value

Linear regression fits the model using optimization approaches such as ordinary least squares (OLS), which estimates coefficients by minimizing the cost function. The formula for this is:

$$\beta = (X^T X)^{-1} X^T y \quad (3)$$

Where X is the matrix of feature values, and Y is the vector of goal values. The model also implies that the residuals (errors) satisfy certain requirements, including independence, constant variance (homoscedasticity), and normality. One-hot encoding is used to convert categorical information into a numerical representation appropriate for the regression model. Regularized Linear Regression techniques, such as Ridge and Lasso, can also be used to address multicollinearity and overfitting by adding penalty terms to the cost function to limit the model's complexity.

4.5. Random Forest

Random Forest is a popular ensemble learning approach for regression and classification applications. It was chosen for this challenge because it can handle datasets with complicated feature interactions and is resistant to overfitting. The approach is particularly well-suited to datasets containing both continuous and categorical variables, hence it is a good choice for the provided dataset.

4.5.1. Justification

Random Forest was chosen because of its adaptability and ability to accurately depict complex relationships. It is especially well-suited for medium to large datasets, as it uses parallel processing to quickly train numerous decision trees while handling high-dimensional feature spaces. The dataset includes both continuous variables, such as cases_active and cases_recovered, and categorical variables, such as one-hot-encoded state and holiday indicators.

Random Forest can process various variable types with minimal preprocessing and is capable of tolerating missing values during training. Random Forest reduces the Mean Squared Error (MSE) across numerous decision trees while performing regression tasks, such as forecasting daily new COVID-19 cases (cases_new). Its capacity to simulate nonlinear linkages and variable interactions makes it an ideal solution for this situation.

4.5.2. Strengths

Random Forest has many advantages that make it ideal for this purpose. As an ensemble learning technique, it integrates numerous decision trees to provide a more accurate and robust prediction. By aggregating predictions from several trees it decreases the possibility of overfitting in a single decision tree. The model excels at capturing nonlinear correlations and complicated interactions between variables that linear models struggle with. Additionally, Random Forest provides a measure of feature relevance, which aids in identifying the most significant characteristics in predicting the target variable. Despite being a more sophisticated model, its structure helps to prevent overfitting and ensures that it generalizes effectively to new data.

The model's use of multiple trees also helps improve interpretability, as it ranks features based on their importance, offering useful insights into the factors driving predictions.

4.5.3. Weaknesses

Despite its advantages, Random Forest has certain drawbacks. One of the primary disadvantages is that it can become computationally expensive with a large number of trees or very high-dimensional data, resulting in longer training durations and increased memory use. The model's complexity can also make it more difficult to comprehend than simpler models, particularly when there are a significant number of trees. Although Random Forest is resistant to overfitting, it may struggle with unbalanced data in which specific classes or ranges of the target variable are underrepresented, resulting in skewed predictions. Finally, Random Forest's use of numerous decision trees can be resource-intensive, needing more processing power and time than simpler models such as Linear Regression.

4.5.4. Architecture Overview and Working Mechanism

Random Forest generates several decision trees using bootstrapped subsets of the training data. The approach begins by creating numerous random subsets of the training dataset using a technique known as bootstrapping. These subsets are used to train specific decision trees. Each tree is constructed by recursively splitting nodes according to the characteristic that reduces the variance of the target variable (cases_new) within the subset. The splitting criteria aim to limit the variation in anticipated values at each node. The trees in the forest are trained individually, and the results of all the trees are combined to provide the final forecast. In regression tasks, the final prediction is the average of the predictions from each tree.

At each split of a tree, random subsets of characteristics are evaluated to identify the appropriate split, increasing tree diversity and decreasing overfitting. Once trained, the trees can handle new data by averaging the predictions of each individual tree in the forest. The model may also determine feature relevance based on how much each feature contributes to reducing variation across the trees. Hyperparameters such as n_estimators (the number of trees), max_depth (the maximum depth of trees), and min_samples_split (the minimum number of samples required for a split) can be adjusted to improve model performance. In addition, out-of-bag (OOB) error estimates can be employed for model assessment, allowing the technique to do cross-validation without requiring a separate validation set.

4.6. LSTM

LSTMs are well-suited for sequential data applications, such as forecasting new COVID-19 cases, since they maintain knowledge of prior time steps over long durations. Unlike classic feedforward networks or simple ANNs, LSTMs efficiently manage the vanishing gradient problem, allowing them to learn from extended sequences without losing information.

4.6.1. Justification

This model was chosen for this task because of its extraordinary capacity to model temporal sequences and understand long-term dependencies, making it appropriate for time series data such as daily COVID-19 cases. LSTMs efficiently handle the vanishing gradient problem, allowing them to learn from extended sequences while preserving crucial information. As a result, they are ideal for sequential data applications such as forecasting "cases_new." The dataset comprises both temporal characteristics, like "date," and continuous variables such as "cases_active" and "cases_recovered." These inputs provide the LSTM with the necessary temporal context to identify patterns and trends in the data.

4.6.2. Strengths

One of the main advantages of LSTMs is their capacity to capture temporal dependencies. Their memory process, aided by cell states and gates (input, forget, and output), guarantees that they store and update pertinent information throughout time. This enables them to concentrate on crucial historical data that contributes to reliable projections. Furthermore, LSTMs are noise-resistant and generalize well when regularization techniques such as dropout are used.

4.6.3. Weaknesses

Despite these advantages, LSTMs have significant drawbacks. Training LSTMs is computationally costly, particularly for large datasets, due to their recurrent nature and reliance on the backpropagation through time (BPTT) technique. LSTMs' performance is also highly sensitive to hyperparameter adjustments, and incorrect selections may lead to unsatisfactory results. Another disadvantage is the difficulty in understanding the model, as LSTMs sometimes operate as "black boxes" with limited transparency into their decision-making processes, even when internal states are displayed.

4.6.4. Architecture Overview and Working Mechanism

An LSTM's architecture consists of several essential components. The input gate, forget gate, and output gate are the three gates present in each of the memory cells that comprise the LSTM layer. The forget gate determines which information from the current time step should be discarded, while the input gate decides which information should be stored in the memory cell. The output gate controls which data is transmitted to the next time step or layer. This gating mechanism allows the LSTM to filter out noise and retain relevant information over time, enabling the model to focus on the most significant temporal patterns.

Input sequences of preset lengths, such as 7-time steps, are formed by the input layer, which receives lagged variables and designed characteristics. At each time step, the LSTM layer uses memory cells with gating mechanisms to choose which information should be output, forgotten, or retained. The input gate chooses what fresh information to add, the output gate controls the information sent to the following layer, and the forget gate eliminates unnecessary information. To lessen overfitting, a dropout layer is used to randomly remove a portion of connections during training. The final prediction (cases_new) for the specified sequence is generated by the dense output layer. The Adam optimizer, which adaptively modifies learning rates for effective training, is used to optimize by minimizing a loss function, such as mean squared error (MSE).

4.7. XGBoost

This algorithm was included in this study due to its ability to handle the amount of data we have and to determine the interactions between variables. It is, however, quite effective at making predictions, especially in the case of a regression task, since it can make predictions through a boosting process. Further, we explain why it is appropriate to use XGBoost, what the advantages of using XGBoost are, and how XGBoost can be implemented. In addition, its architecture and working mechanism are also described.

4.7.1. Justification

Since we are dealing with a dataset that contains both continuous and categorical features, such as data that is one-hot-encoded, XGBoost is a suitable candidate model for our dataset due to its ability to handle missing values and split decisions. Moreover, XGBoost is designed for medium to large datasets and provides high scalability and speed using its histogram-based algorithms and parallel computation.

The primary task is to forecast the number of new COVID-19 cases (cases_new) on a daily basis, which is a regression problem. Since XGBoost is capable of minimizing loss functions such as Root Mean Squared Error (RMSE) to the best extent possible, it is very suitable for this task. This makes it suitable for capturing trends, dependencies, and patterns in the data to make accurate predictions.

4.7.2. Strengths

XGBoost excels in several areas, making it a powerful machine learning model. A major strength of this model is that it can handle missing values and integrate them into the split optimization process during training. Additionally, XGBoost includes both L1 (Lasso) and L2 (Ridge) regularization to prevent overfitting and improve the model's ability to generalize. The model also provides tools for evaluating feature importance, which helps in selecting important predictors and improving the interpretability of the model. Furthermore, XGBoost is a powerful tool for handling lagged features and thus is able to capture trends and temporal dependencies in time series data.

4.7.3. Weaknesses

Despite all this, XGBoost has some limitations. The training process of large datasets with extensive hyperparameter tuning can be resource-demanding. Also, the model is not easy to interpret because of the ensemble nature of XGBoost, although feature importance can be evaluated. In addition, if not properly regulated and tuned for the best parameters, there is a possibility of overfitting, which is not good for the performance of a model.

4.7.4. Architecture Overview and Working Mechanism

XGBoost is a systematic approach to building predictive models. First of all, for the purpose of this study, lag_1 to lag_7 are defined as the lag variables of “cases_new” to account for the temporal characteristics. The features are a mix of numerical variables, categorical variables such as “holiday,” and derived variables like “is_weekend.” The boosting process begins with an initial prediction, which is the mean of the target variable. Subsequent decision trees model the residuals, iteratively improving predictions by focusing on correcting errors made by previous trees.

The splitting of the tree is done by a greedy algorithm that selects the node split that reduces the loss function the most, such as RMSE for regression. It then chooses the split with the highest gain, and from this split, it determines the best feature and threshold to split on. XGBoost minimizes a regularized objective function that includes a term for the loss function, such as RMSE, and a term that describes model complexity to improve generalization. The techniques used for regularization include L1 regularization, which results in sparsity, with some features having their weights set to prevent zero, thereby reducing overfitting.

Optimization and regularization of L2 parameters involve adjusting hyperparameters such as learning rate, maximum depth, subsample rate, and “colsample” in trees to achieve the optimal model complexity. Early stopping rounds are used to halt training when improvement on the validation set ceases, in order to obtain the best performance. The important features that contribute to the model’s ability to make predictions are, for instance, “cases_recovered” and “cases_cluster,” which help in understanding how the model works.

In summary, XGBoost was selected as it is robust in handling large datasets, mixed data types, and temporal dependencies in order to predict daily new COVID-19 cases. It has both strengths and weaknesses; the strengths include its ability to handle missing values, its regularization term, and its capacity to rank features by their importance. However, it is computationally expensive and can overfit if not properly tuned. It is designed to make interpretable and accurate predictions through its architecture of iterative boosting, regularized optimization, and parameter tuning. In this case, XGBoost is a suitable approach for a regression task.

5. Evaluation Indices

In the evaluation process, all the models used four metrics to assess the performance of the regression model predicting new COVID-19 cases.

Firstly, Root Mean Squared Error (RMSE) was used to determine the performance of the regression model and interpret its performance. This intuitive measure makes it easier to apply the model in real life, as it focuses more on significant deviations that the model should minimize during the optimization process. Its formula is expressed as:

$$RSME = \sqrt{\frac{1}{n}\sum(y_i - \hat{y}_i)^2} \qquad (4)$$

where y_i represents the actual values and $\hat{y}_i$ denotes the predicted values. RMSE is particularly useful in regression tasks for its ability to guide the model towards better generalization.

In addition to that, Mean Absolute Error (MAE) was also used to measure the average size of errors without concerning itself with direction. Unlike the Mean Squared Error (MSE) and the Root Mean Squared Error (RMSE), the MAE treats all errors with the same importance and is therefore a very good metric to use when there are outliers or the error distribution is not uniform. MAE is calculated as:

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i| \qquad (5)$$

The third metric, Mean Absolute Percentage Error (MAPE), was employed to assess the average error as a percentage of the actual values. This metric is particularly useful for understanding prediction accuracy relative to the scale of the target variable, making it easier to interpret across datasets with varying magnitudes. MAPE is calculated as:

$$MAPE = \frac{100}{n} \left| \frac{y_i - \hat{y}_i}{y_i} \right| \qquad (6)$$

Lastly, R-squared (R^2) was also used to assess how much of the variance in the target variable is explained by the variability model. In values of the data, R^2 close to 1 indicates that the model has captured most of the data variability, while values close to 0 indicate that the model does not explain the data variability well.

These metrics were chosen to provide a comprehensive view of the model's performance. The RMSE and MAE offer insights into the accuracy of the predictions, while R^2 measures how effectively the model explains data variability. Therefore, these metrics are useful in assessing whether the model is suitable for the regression task at hand.

6. Model Development

6.1. Training and Testing Process

In order to forecast COVID-19 cases, five different models will be employed. A standardized training and testing process was followed for each of the models. The initial step was to divide the dataset into 80% for training and 20% for testing. The training data was used to establish patterns, while the testing data was not exposed to the model during training, in order to assess its performance on new data. Using an 80:20 split is generally a good approach to ensure sufficient data for training while reserving enough data for evaluating the model's performance. Moving on to the evaluation process, all models were evaluated on the testing set using performance metrics such as RMSE, MAE, R2, and MAPE. This comparison of results across models allows for a comprehensive assessment in determining the most effective technique.

6.2. Software and Tools

All of the models were written in Python because it is a language that has many libraries for the tasks of machine learning, data preprocessing, and deep learning. The following programming languages and libraries were leveraged in the implementation:

6.3. Programming Language

- Python: The primary language for data preprocessing, modeling, and evaluation.

6.4. General Libraries

- Pandas: For data manipulation, including feature creation and one-hot encoding of categorical variables.
- NumPy: For numerical work such as logarithms and reversing the scaling of outputs. • Matplotlib: For data visualization, including heatmaps of correlations, learning curves, and residuals.
- Matplotlib: For visualization of data, including heatmaps of correlations, learning curves, and residuals.
- Scikit-learn: For splitting the dataset, scaling the features, and calculating evaluation metrics such as RMSE, MAE, R^2, and MAPE.

6.5. Model-Specific Libraries

- TensorFlow/Keras (for ANN and CNN): TensorFlow is a popular open-source library that can be used for neural networks.

- XGBoost: for IT building, is trained as a library well for gradient evaluation, a boosted decision tree, and can be used for hyper-parameter tuning and performance tuning of the model.
- Random Forest: This is where the module in Scikit-learn is ensemble with features like RandomForestRegressor for model training.

6.6. Artificial Neural Networks (ANNs)

6.6.1. Hyperparameter Tuning

The hyperparameters of the artificial neural network (ANN) were not optimized using a systematic approach; instead, they were set heuristically. The activation function used in the network architecture consisted of two nonlinear hidden layers. The network was configured with 64 neurons in one layer and 32 neurons in the other, with a batch size of 32. The ReLU activation function was chosen, which provides a reasonable balance between computational resources and model training time. The model was trained for 20 epochs, and early stopping was implemented based on the validation loss trend. The Adam optimizer was used due to its ability to tune the learning rate during training, thereby reducing the need for adjusting hyperparameters. Other automated search techniques, like Grid Search or Random Search, were not employed due to the simplicity of the architecture; instead, hyperparameters were fine-tuned based on validation performance.

6.6.2. Settings and Configurations

The number of features in the input layer was made to be equal to the number of features in the training data. Three hidden layers of fully connected layers with 128, 64, and 32 neurons were employed, and the ReLU activation function was used in all of them. There was no activation function in the output layer, as it is for a regression task, and it has a single neuron.

6.7. Convolutional Neural Networks (CNN)

6.7.1. Hyperparameter Tuning

The hyperparameters, such as the number of filters in the CNN, the size of filters, the batch size, the learning rate, and the number of epochs, play a significant role in the performance of the CNN model. The key parameters of this model are explained as follows. The 1D Convolutional Neural Network has 64 filters in the first convolutional layer. The number of filters is a parameter that determines the number of feature maps produced by the convolution operation and is usually increased to identify more complex features at the cost of increased computational complexity. The first convolutional layer has a common filter size of 3x3. This is to limit the window through which the convolution operation is applied to the input.

The batch size in this algorithm is set to 32, as it ensures that there is enough memory and the model converges well. Small batch sizes have high gradient noise, while large batch sizes take a long time to train. The learning rate of a model describes how much the model adjusts its weights during the training process. The Adam optimizer with an initial learning rate of 0.001 was used. When the model is trained on a CNN, it is trained for 20 epochs, which helps in preventing overfitting. The number of look-back historical window data, which should be 10, is employed. This is considered for hyperparameter tuning to make accurate predictions. Adjustments can be made based on the results.

6.7.2. Settings and Configurations

The data of each variable was reshaped into a 3D format to be compatible with the Conv1D layer of the CNN. The input layer of the CNN matched the dimensions of the image data. Next, two

3x3 convolutional layers were employed, each followed by 2x2 max pooling. The convolutional layers were then followed by a fully connected dense layer with 64 neurons. The output layer contained a single neuron, suitable for regression tasks, with no activation function.

K-Nearest Neighbours (KNN)

6.7.3. Hyperparameter Tuning

The K-Nearest Neighbours (KNN) algorithm depends on the proper selection of hyperparameters to achieve the optimal trade-off between the model's generalization capability and its ability to fit the training data. The two most important hyperparameters are the value of k (the number of neighbors considered) and the distance metric used, such as Euclidean or Manhattan distance. The value of k determines how many nearest neighbors to consider for classification, and the distance metric defines how to measure the similarity between data points. These hyperparameters should be tuned to minimize prediction errors and avoid overfitting, often through grid search or other fitting methods.

6.7.4. Settings and Configurations

The KNN model is a simple model that predicts the target variable as the mean or weighted mean of the k nearest neighbors. To achieve optimal performance, the dataset was normalized before applying the KNN algorithm because KNN is sensitive to the scale of the features. The optimal number of neighbors, k, was also tested across a range of values; smaller values are suitable for capturing local patterns, while larger values help reduce noise. Additionally, different distance metrics were evaluated to identify the most suitable for the characteristics of the dataset.

6.8. Linear Regression

6.8.1. Hyperparameter Tuning

Linear regression models are straightforward and require minimal tuning. However, some level of regularization was employed to improve the models' ability to generalize. Both Ridge (L2) and Lasso (L1) regularization were applied to prevent overfitting by penalizing large coefficients. The amount of penalty was controlled by the alpha parameter.

6.8.2. Settings and Configurations

The linear regression model assumes that the input features and the target variable are linearly related. Both Ridge and Lasso regularization techniques were employed, and the alpha parameter was adjusted to achieve a good fit and sufficient generality of the model.

6.9. Random Forest

6.9.1. Hyperparameter Tuning

To ensure the Random Forest model had sufficient capacity, the number of trees (n_estimators) was set to 100. The maximum depth (max_depth) of each decision tree was restricted to 10, while the minimum samples required to split an internal node (min_samples_split) was set to 2. Additionally, the maximum number of features considered for each split (max_features) was set to the square root of the total number of features, a common practice that helps prevent overfitting and reduces computational complexity.

6.9.2. Settings and Configurations

The Random Forest was set with 100 trees in the ensemble. The maximum depth of each tree was limited to 10, and the minimum samples required to split an internal node was set to 2. The max_features parameter was set to the square root of the total number of features to control the complexity of the individual trees. A random state of 42 was set to ensure reproducibility across different runs.

6.10. XGBoost

6.10.1. Hyperparameter Tuning

XGBoost is a highly customizable algorithm, and hyperparameters were fine-tuned for optimal performance. The learning rate (eta) was set to 0.1, a standard value used to control the step size in gradient boosting. The maximum depth (max_depth) was set to 6 to control the complexity of the individual trees and prevent overfitting. The model was set to run for 100 boosting rounds (n_estimators). Data during each boosting round was subsampled at a ratio of 0.8 to help reduce overfitting. The "colsample_bytree" parameter was also set to 0.8.

6.10.2. Settings and Configurations

The gradient descent step size was adjusted using the XGBoost model with a learning rate of 0.1. To minimize model complexity and lower the chance of overfitting, the maximum depth of the trees (max_depth) was set at 6. 100 was chosen as the number of boosting rounds (n_estimators). To increase model resilience and avoid overfitting, subsampling (subsample) and feature sampling (colsample_bytree) were used at ratios of 0.8.

Early stopping was activated with early_stopping_rounds=10, which meant that after ten rounds in which no increase in validation performance was observed, the training process would end. Up to 100 boosting cycles were used to train the model; however, if validation performance stopped improving, training would end sooner.

6.11. LSTM

6.11.1. Hyperparameter Tuning

LSTM is a fully customizable model, with hyperparameters tailored for optimal performance. The learning rate was adjusted to 0.001 to achieve a consistent balance between computational efficiency and model performance. The batch size was set to 32 to limit the quantity of data handled at each training stage. The number of LSTM units was set to 100, allowing the model to accurately capture temporal patterns. A dropout rate of 0.3 was used to prevent overfitting and ensure that the model can generalize effectively to complex patterns. Early stopping was implemented to cease training if validation performance did not improve within a predetermined number of epochs, ensuring efficient use of training resources.

6.11.2. Settings and Configurations

The learning rate was set to 0.001 for the LSTM model to achieve a good balance between efficient convergence and stable training performance. Furthermore, the batch size was optimized for computational efficiency, with a setting of 32. The LSTM layer had 100 units to learn the temporal dependencies in the data effectively. To prevent overfitting, a dropout rate of 0.3 was utilized to ensure that it generalizes well. Early stopping is implemented when the validation loss does not decrease over a preset number of epochs so that the training does not become overly complex or begin overfitting.

6.12. XGBoost

6.12.1. Hyperparameter Tuning

Several important hyperparameters were tuned to ensure that the model neither overfits nor underfits. The following settings were used for all models: The learning rate was set to 0.1 to control the step size during training. To prevent overfitting, the maximum depth was limited to 6, helping to select an optimal model complexity. The subsample parameter was set to 0.8, meaning that 80% of the data was used for training each tree, improving the generality of the ensemble model. Similarly, the colsample_bytree parameter was set to 0.8, enhancing the model's robustness. Early stopping was incorporated to reduce overfitting and ensure efficient training, with the process terminating if the validation RMSE did not improve for 10 consecutive rounds.

Manual hyperparameter tuning was employed by iterating over different values based on domain knowledge. However, for further refinement, techniques such as Grid Search or Random Search could be explored to automate and optimize the parameter selection process.

6.12.2. Settings and Configurations

The XGBoost model was configured with several key parameters. The objective was set to 'reg:squarederror' to specify that the task is a regression problem using squared error as the loss function. Root Mean Squared Error (RMSE) was selected as the evaluation metric to assess model performance. Key hyperparameters were chosen to balance model complexity and performance, including learning_rate=0.1, colsample_bytree=0.8, subsample=0.8, and max_depth=6.

Early stopping was enabled with early_stopping_rounds=10, allowing training to halt if no improvement was observed in validation performance for 10 consecutive rounds. The model was trained for up to 100 boosting rounds, with training stopping early if no further improvement was seen on the validation set.

6.12.3. Data Handling

To include temporal dependencies in the model, lag features (lag_1 to lag_7) were derived to incorporate the previous values of the target variable, thereby improving prediction accuracy. Any row containing missing values (which were introduced by the lagging process) was dropped to ensure dataset consistency.

7. Results and Discussion

The following results and discussion under A and B will present the performance of each model on both the training and testing datasets, primarily focusing on the comparison of R-squared values. Additionally, under *C. Comparisons of Models*, the overall performance of the models, and their respective subsets will be analyzed, considering all four evaluation metrics, providing a comprehensive assessment of each model's effectiveness.

7.1. Training Sets

Table 5.
Top 5 Iterations for Training Sets based on Average R^2 Values.

Parameters	Training				
	Models	**RMSE**	**R^2**	**MAE**	**MAPE**
Remove quarter, is_weekend, deaths_vax, day, holiday, deaths new dod, deaths unvax, deaths new, day of week, year, deaths pvax, cases import, deaths boost	ANN	129.28	0.9995	73.83	18.95%
	CNN	762.15	0.9845	387.36	24.05%
	KNN	774.18	0.9839	329.83	17.57%
	Linear Regression	3.95	0.9967	2.36	2.64%
	LSTM	1074.75	0.9691	602.01	113.21%
	Random Forest	178.84	0.9989	56.31	1.77%
	XGBoost	71.80	0.9999	47.27	22.42%
	Average	427.85	0.9904	214.14	28.66%
	Models	RMSE	R^2	MAE	MAPE
Remove quarter, is_weekend, deaths_vax, day, holiday, deaths_new_dod	ANN	529.04	0.9912	277.13	27.59%
	CNN	720.79	0.9861	363.73	22.70%
	KNN	774.19	0.9839	329.88	17.43%
	Linear Regression	7.89	0.9983	3.84	1.73%
	LSTM	1,016.50	0.9724	590.27	215.62%
	Random Forest	198.76	0.9987	64.36	1.84%
	XGBoost	62.64	0.9999	41.55	19.27%
	Average	472.83	0.9901	238.68	43.74%
	Models	RMSE	R^2	MAE	MAPE
Remove quarter, is_weekend, deaths_vax, day, holiday	ANN	131.66	0.9995	76.35	20.36%
	CNN	739.03	0.9854	373.37	23.26%
	KNN	774.19	0.9839	329.87	17.42%
	Linear Regression	2.27	0.9889	1.30	1.46%
	LSTM	1,070.02	0.9694	722.31	303.25%
	Random Forest	201.16	0.9987	64.51	1.96%
	XGBoost	66.33	0.9999	44.84	21.31%
	Average	426.38	0.9894	230.37	55.57%
	Models	RMSE	R^2	MAE	MAPE
Remove quarter, is_weekend, deaths_fvax	ANN	210.98	0.9986	113.72	20.73%
	CNN	730.01	0.9858	370.04	23.92%
	KNN	774.19	0.9839	329.87	17.43%
	Linear Regression	1.24	0.9900	7.67	9.34%
	LSTM	1,135.90	0.9655	680.11	431.54%
	Random Forest	193.96	0.9988	62.35	1.82%
	XGBoost	66.15	0.9999	44.87	18.13%
	Average	444.63	0.9889	229.81	74.70%
	Models	RMSE	R^2	MAE	MAPE
Remove quarter, is_weekend, deaths_vax, day, holiday, deaths new dod, deaths unvax, deaths new, day of week, year, deaths pvax, cases import, deaths boost, cases cluster	ANN	403.59	0.9949	229.77	65.38%
	CNN	755.57	0.9847	379.42	21.80%
	KNN	776.17	0.9838	332.09	17.94%
	Linear Regression	3.95	0.9982	2.36	2.65%
	LSTM	1,204.90	0.9612	672.85	41.48%
	Random Forest	177.06	0.9990	55.89	1.76%
	XGBoost	76.51	0.9998	49.98	23.75%
	Average	485.39	0.9888	246.05	24.97%

According to Table 5, the focus is mainly on the average R² metric for evaluating each model's performance and selecting the top-performing models based on their average R² in each iteration, also known as the coefficient of determination. It helps measure how well each model explains the variance in the dependent variable. It is chosen because it provides a clear indication of the model's ability to predict the outcome accurately. A higher R² value indicates better model performance, with a value of 1 signifying perfect prediction, and values closer to 0 indicating that the model does not explain the variability in the data well.

As demonstrated in Table 1, the highest average R2 value is 0.9904, observed in the subset where parameters such as "quarter, is_weekend, deaths_vax, day, holiday, deaths new dod, deaths unvax, deaths new, day of week, year, deaths pvax, cases import, deaths boost" are removed. This indicates that the removal of certain irrelevant or noisy features enables the models to better fit the training data and explain the variance in the target variable. In this case, XGBoost provides the highest R2 value of 0.999, followed closely by ANN with 0.9995. Both models demonstrate that these models can achieve near-perfect fits when these particular features are excluded.

By analyzing the results, it is proven that XGBoost consistently shows the best R² performance across all iterations, with each R² value approaching 1.0, indicating its superior ability to fit the data compared to other models. This is mainly due to its ability to capture complicated patterns and relationships in the dataset through boosting. Next, ANNs perform outstandingly well, especially in iterations with fewer parameters, with R² values approaching an average of 0.9995. The ability to model and understand non-linear relationships contributes to its strong fit. Based on the table, XGBoost, ANNs, and Random Forest perform well in predicting new cases. The Linear Regression model performs well when fewer features are present, but is weaker compared to more complex models like ANN and XGBoost.

7.2. Testing Set

Table 6.
Top 5 Iterations for Testing Sets based on Average R² Values.

Parameters	**Testing**				
	Models	**RMSE**	**R²**	**MAE**	**MAPE**
Remove quarter, is_weekend, deaths_vax, day, holiday, deaths_new_dod, deaths_unvax, deaths_new	ANN	221.41	0.9985	103.95	9.09%
	CNN	136.54	0.9611	60.01	22.63%
	KNN	248.12	0.9135	165.12	123.85%
	Linear Regression	2.72	0.9782	1.69	8.69%
	Random Forest	439.80	0.9939	149.80	3.89%
	LSTM	268.19	0.8972	171.17	96.81%
	XGBoost	94.51	0.9875	38.05397	1468.44%
	Average	201.61	0.9614	98.54	247.63%
	Models	RMSE	R²	MAE	MAPE
Remove quarter, is_weekend, deaths_vax, day, holiday, deaths_new_dod	ANN	514.40	0.9917	281.10	24.77%
	CNN	127.06	0.9664	57.61	21.71%
	KNN	247.78	0.9134	164.77	123.83%
	Linear Regression	7.87	0.9965	3.91	2.01%
	Random Forest	446.42	0.9937	155.20	4.04%
	LSTM	295.88	0.8748	184.32	87.74%
	XGBoost	74.33	0.9922	36.72553	592.06%
	Average	244.82	0.9612	126.23	122.31%

Remove quarter, is_weekend, deaths_vax, day, holiday, deaths_new_dod, deaths_unvax	Models	RMSE	R^2	MAE	MAPE
	ANN	590.77	0.9891	311.76	27.25%
	CNN	128.79	0.9654	59.88	22.05%
	KNN	247.78	0.9134	164.80	123.84%
	Linear Regression	1.91	0.9892	1.15	9.02%
	Random Forest	435.27	0.9940	148.84	3.93%
	LSTM	315.07	0.8581	187.24	100.21%
	XGBoost	91.22	0.9883	38.58665	1458.61%
	Average	258.69	0.9568	130.32	249.27%
ALL	Models	RMSE	R^2	MAE	MAPE
	ANN	510.04	0.9918	235.47	17.89%
	CNN	119.81	0.9701	58.62	21.81%
	KNN	247.78	0.9134	164.77	123.83%
	Linear Regression	6.23	0.9954	3.49	3.86%
	Random Forest	449.36	0.9936	152.25	4.00%
	LSTM	342.75	0.8320	171.00	64.02%
	XGBoost	98.97	0.9860	39.08	15.13%
	Average	253.56	0.9546	117.81	35.79%
Remove quarter and is_weekend	Models	RMSE	R^2	MAE	MAPE
	ANN	147.56	0.9993	81.11	7.29%
	CNN	126.85	0.9665	59.41	21.66%
	KNN	154.65	0.9234	154.21	110.92%
	Linear Regression	6.23	0.9932	3.49	3.86%
	Random Forest	453.35	0.9934	153.25	4.01%
	LSTM	360.27	0.8144	239.99	147.83%
	XGBoost	98.97	0.9863	39.08266	15.18%
	Average	192.56	0.9538	104.36	44.39%

Table 6 above presents a comparison of model performance for different subsets of features across various models used, including ANNs, CNN, KNN, Linear Regression, Random Forest, LSTM, and XGBoost during the testing phase. For each iteration, the models are evaluated based on four metrics: Root Mean Squared Error (RMSE), Coefficient of Determination (R^2), Mean Absolute Error (MAE), and Mean Absolute Percentage Error (MAPE). This table aims to evaluate and compare the performance based on their ability to explain the variance in the target variable, which is most effectively captured by the R^2 value.

As shown in Table 2, the highest average R^2 value is obtained when parameters such as "quarter, is_weekend, deaths_vax, day, holiday, deaths_new_dod, deaths_unvax, deaths_new" are removed from the dataset, with an average R^2 of 0.9614. This indicates that removing these specific features enhances the models' ability to capture the underlying patterns in the data and explain the variability in the target variable. By eliminating these features, the performance of each model can be improved due to a greater focus on more relevant features. Furthermore, it is evident that models like ANN and Random Forest consistently demonstrate stronger performance, with ANN in particular achieving a high R^2 score across most subsets, indicating its capacity to capture more complex nonlinear relationships in the data.

Despite that, Linear Regression also performs well in predicting the COVID-19 cases, especially when fewer features are used, indicating its effectiveness in simpler linear relationships. CNN, KNN, and LSTM models do show lower average R^2 scores when compared with other models, proving that they are more sensitive to irrelevant features in the data. Next, performance metrics like RMSE, MAE, and MAPE are useful in understanding the error magnitude and percentage error in the testing phase, but are least important to R^2 in terms of identifying the model that best explains the variance in the target variable. For instance, although CNN produces lower RMSE values compared to ANN in certain subsets, its lower R^2 indicates that it is not as effective in capturing the overall patterns in the data.

The reason for focusing on R^2 instead of other performance metrics like RMSE, MAE, and MAPE is due to the fact that R^2 directly evaluates the model's explanatory power. This feature is particularly useful in this case, where the goal is to understand the relationship between input features and the target variable, which is cases_new. Besides, the R^2 metric has been prioritized as it directly reflects the degree of effectiveness of each model in fitting the data, making it the most appropriate measure for assessing the predictive capability of the models.

To sum up, ANN acts as the most reliable and effective model in predicting COVID-19 cases, presenting an exceptional performance in both the training and testing processes. With a near-perfect R^2 value of 0.9995 in the training set and consistently high scores in the testing sets, ANN has proven its ability to capture the sophisticated nonlinear relationships in the dataset. Additionally, it has demonstrated its reliability in accurately predicting COVID-19 cases while maintaining strong generalization capabilities.

7.3. Comparison of Models

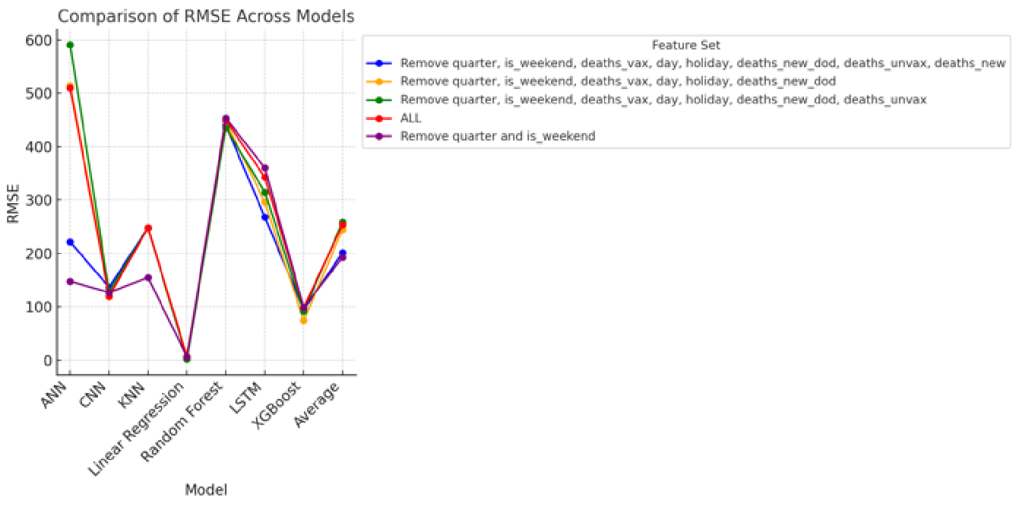


Figure 23.
Comparison of RMSE Across Models.

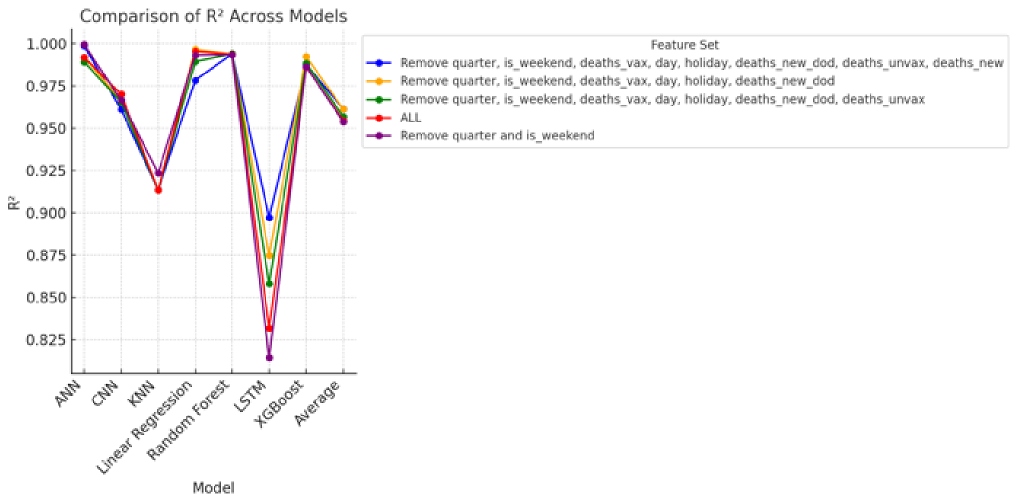


Figure 24.
Comparison of R² Across Models.

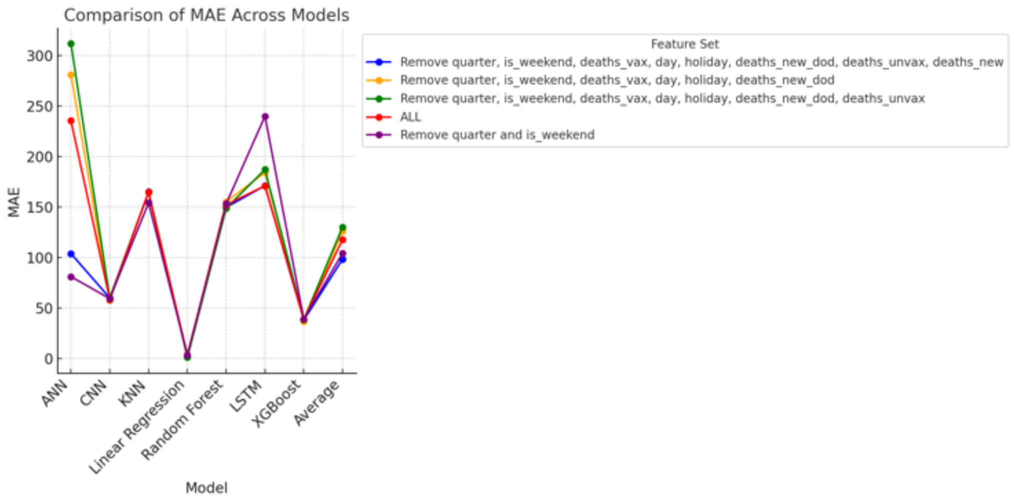


Figure 25.
Comparison of MAE Across Models.

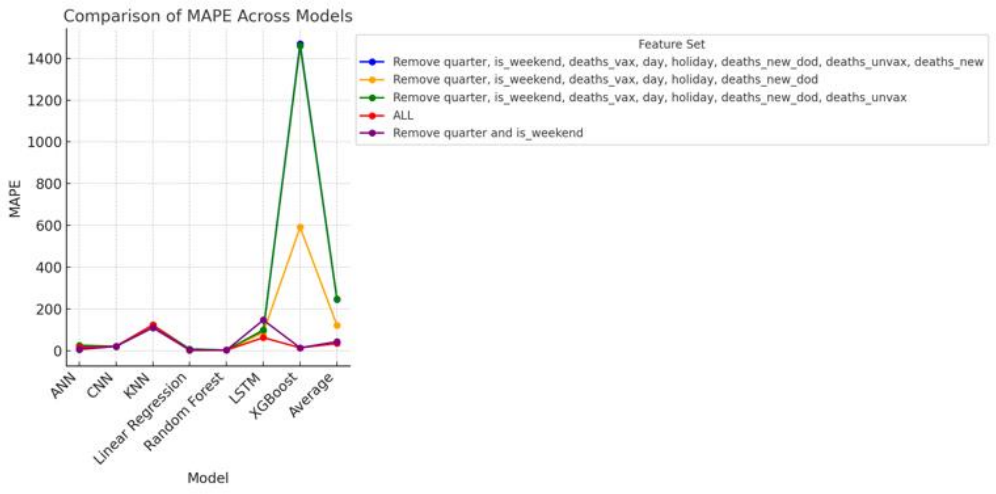


Figure 26.
Comparison of MAPE Across Models.

Below is a detailed breakdown and comparison of how each model performed across the different subsets based on RMSE, R², MAE, and MAPE metrics, also shown in Figure 18, Figure 19, Figure 20, and Figure 21. These metrics reflect the accuracy, fit, and generalization of each model.
Conditioned Subset 1: Remove quarter, is_weekend, deaths_fvax, day, holiday, deaths_new_dod, deaths_unvax, deaths_new.

- Best Model: Linear Regression
- RMSE: 2.72 (lowest)
- R²: 0.9782 (strong fit)
- MAE: 1.69 (lowest)
- MAPE: 8.69% (very low)
- Other Observations:
- XGBoost had good performance with low RMSE (94.51) but extremely high MAPE (1468.44%), indicating instability in predictions for some cases.
- ANN and CNN showed moderate performance, with ANN having a lower MAPE (9.09%) but a higher RMSE compared to CNN.

Conditioned Subset 2: Remove quarter, is_weekend, deaths_fvax, day, holiday, deaths_new_dod

- Best Model: Linear Regression
- RMSE: 7.87 (lowest)
- R²: 0.9965 (best fit)
- MAE: 3.91 (lowest)
- MAPE: 2.01% (very low)
- Other Observations:
- XGBoost had the second-lowest RMSE (74.33) but again suffered from high MAPE (592.06%).
- CNN had stable and consistent results (RMSE: 127.06, MAPE: 21.71%).

Conditioned Subset 3: Remove quarter, is_weekend, deaths_fvax, day, holiday, deaths_new_dod, deaths_unvax.

- Best Model: Linear Regression
- RMSE: 1.91 (lowest)
- R^2: 0.9892 (excellent fit)
- MAE: 1.15 (lowest)
- MAPE: 9.02% (low)
- Other Observations:
- XGBoost continued its trend of low RMSE (91.22) but had a very high MAPE (1458.61%), making it unreliable.
- Random Forest performed decently with RMSE (435.27) but had higher MAE and MAPE compared to the best model.

Conditioned Subset 4: ALL Parameters

- Best Model: Linear Regression
- RMSE: 6.23 (lowest)
- R^2: 0.9954 (best fit)
- MAE: 3.49 (lowest)
- MAPE: 3.86% (very low)
- Other Observations:
- CNN showed the second-best performance with an RMSE of 119.81 and a MAPE of 21.81%, maintaining consistency.
- ANN showed high RMSE (510.04), indicating weaker performance on this subset.

Conditioned Subset 5: Remove quarter and is_weekend

- Best Model: Linear Regression
- RMSE: 6.23 (lowest)
- R^2: 0.9932 (strong fit)
- MAE: 3.49 (lowest)
- MAPE: 3.86% (very low)
- Other Observations:
- ANN showed competitive performance (RMSE: 147.56, MAPE: 7.29%) and is notable for being stable across subsets.
- XGBoost showed good RMSE (98.97) but again suffered from high MAPE in specific cases.

Table 6.
Comparison of Models.

Model	Strengths	Weaknesses
Linear Regression	Best across all subsets with the lowest RMSE, MAE, and MAPE	Performance may decline in more complex datasets or nonlinear cases.
XGBoost	Excellent RMSE and R^2 in most subsets	An extremely high MAPE indicates instability in predictions.
ANN	Good R^2, low MAPE in some subsets, and stable performance	RMSE is generally higher than other models.
CNN	Consistent RMSE and MAPE across subsets	Struggles with very low MAPE compared to simpler models.
Random Forest	High R^2 and moderate RMSE	Higher MAE and MAPE in comparison to Linear Regression.
KNN	Weakest performance overall	High RMSE, MAE, and MAPE across all subsets.
LSTM	Moderate RMSE but unstable MAPE across subsets	Struggles with generalization and precision.

In the subset with all parameters removed (minimal features), linear regression emerged as the best-performing model, consistently outperforming all others across key metrics such as RMSE, MAE, and MAPE. While ANN and CNN showed reasonable performance, XGBoost struggled significantly with MAPE, highlighting its instability when features were limited.

For the subset with all parameters included (full features), Linear Regression was again the clear winner, achieving the lowest RMSE, MAE, and MAPE. CNN demonstrated consistent results across metrics but was not as competitive as Linear Regression. The inclusion of all features appeared to benefit most models, showcasing the importance of comprehensive data in enhancing predictive accuracy.

When a few parameters were removed (quarter and weekend), the outcomes were quite similar to those of the full-feature subset. The model still performed well with most of the features, as this subset also showed high accuracy. Other models, such as ANN and CNN, did a reasonable job but did not outperform Linear Regression in terms of accuracy.

In conclusion, Linear Regression was the best model overall in all subsets because it had low RMSE, MAE, and MAPE. XGBoost and ANN are promising, but they are unstable in MAPE and are not as reliable in some situations. It is clear that more features improve performance for all models, while the minimal feature subset shows that Linear Regression is simple and robust, with high accuracy even with fewer parameters.

7.4. Parameter Analysis

7.4.1. Artificial Neural Network (ANNs)

The importance of key features and parameters in ANNs cannot be overemphasized, as they determine the overall performance of the model. This analysis provides an in-depth insight into how the relevance of features and hyperparameter settings work together to achieve optimal results.

The accuracy of an ANN is greatly dependent on the quality and relevance of the input features. Such features include deaths_new, cases_active, and deaths_unvax because they are in direct relation to the trends of mortality and active cases, which the model is trying to predict (cases_new). Temporal variables like month, day, and is_weekend capture seasonal and behavioral trends and hence provide important information to the model. On the other hand, poor or superfluous features increase noise, which negatively affects accuracy. Low-variance or weakly related features to the target variable are not useful, while multicollinearity (high correlations between features) distorts the model's ability to identify the importance of individual features, resulting in poor learning.

The effectiveness of the ANN is also determined by its architecture, and some features are critical when designing the network. The most important variables are the number of hidden layers and the number of neurons in each layer because they define what complex relationships the model can learn. Increasing these in a free and unregulated model will improve the capacity of the model, but also the risk of overfitting. The ReLU activation function was chosen for hidden layers to deal with non-linear relationships, and a linear activation function was used in the output layer, as it is a regression problem. Also, the learning rate was adjusted (for example, 0.001) to achieve a good trade-off between the rate of convergence and the avoidance of oscillations in the search for the optimum.

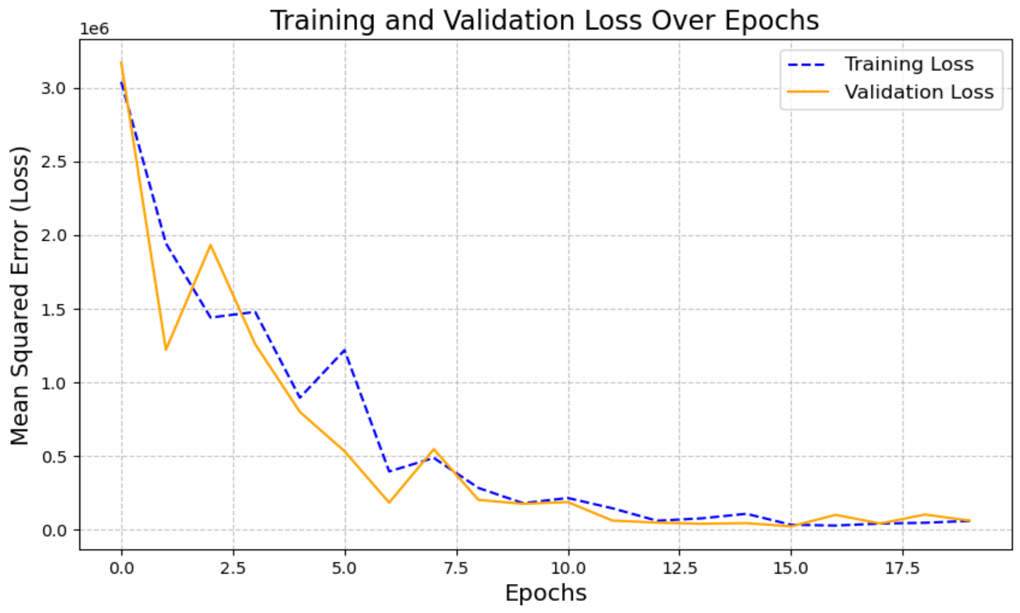


Figure 27.
ANN Training & Validation over Epochs.

7.4.2. Convolution Neural Network (CNN)

The filters in a Conv1D layer represent how many different feature maps this layer will learn from the input data. The filters are mainly used for extracting features such as edges and shapes from the current time-series data. The more filters used in this model, the more efficient the model becomes at capturing complex and varied features, and hence learns to identify detailed patterns in the data. However, with an increase in the number of filters, the number of parameters in the model also increases, and if the dataset is not sufficient to support the complexity, it may result in overfitting. Therefore, it is significant to use an appropriate number of filters to extract relevant patterns without impairing the model's ability to generalize to new data. In the current case of time-series forecasting, different filters assist the model in identifying aspects like periodicity and anomalies in the data.

Moving on to the kernel size, it is another important parameter that describes the width of the sliding window in the convolutional layer. For instance, a kernel size of 3 indicates that each filter examines a sequence of 3 consecutive time steps in the input data. The kernel size has a significant impact on the resolution of each specific data that the CNN model can learn to capture. A smaller kernel size of 3 or 5 is effective in identifying fine-grained and local patterns, while a larger kernel size can be better at capturing long-range patterns. Therefore, it is important to choose an appropriate kernel size to learn significant trends and relationships between the time steps in time-series data. A kernel size of 3 is a good balance because it enables the model to detect local temporal patterns without overfitting to short-term noise.

As shown in Figure 27 below, both the kernel sizes of 3 and 5 obtain high R^2 values, which indicate that the model performs well on the testing set. However, it has been found that the model with a kernel size of 3 achieved a higher R^2 value of 0.9696. This demonstrates that using a smaller kernel size is more effective at extracting relevant features from the data, as the R^2 values are closer to 1.

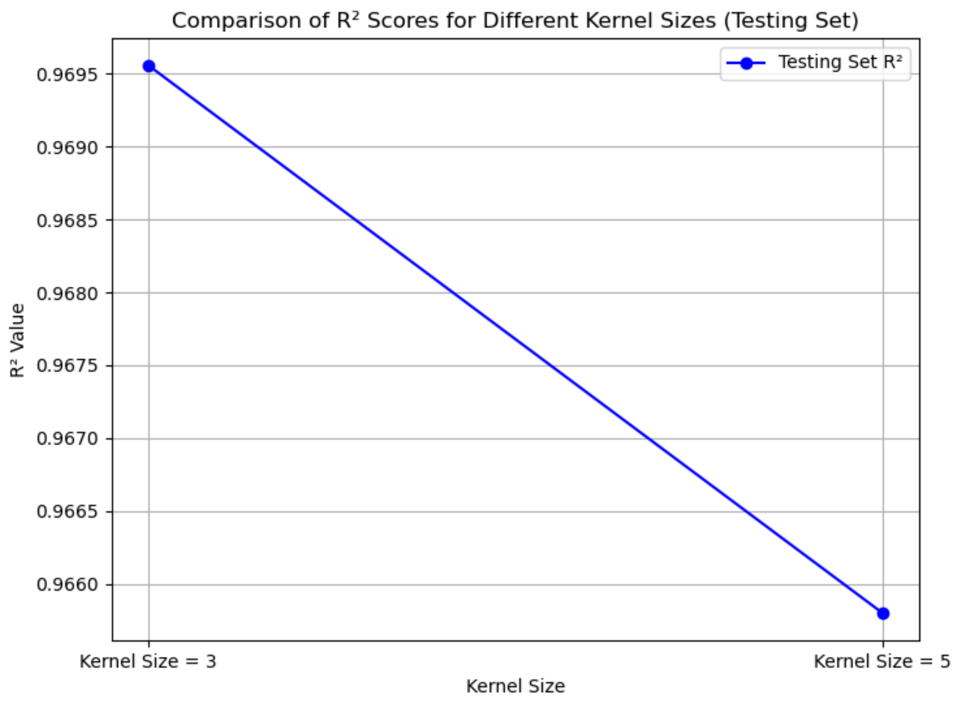


Figure 28.
Line Chart of R^2 Scores with Different Kernel Size.

7.4.3. K-Nearest Neighbours (KNN)

The performance of the K-Nearest Neighbours (KNN) algorithm is highly dependent on the quality of input features and the proper tuning of hyperparameters. This analysis explores the significance of these factors and how they influence the model's ability to make accurate predictions.

KNN is a method used to select k nearest neighbors based on calculating distances in the feature space, and these neighbors are employed for classification or regression. Features such as deaths_new, cases_active, and deaths_unvax are cases (cases_new). Very essential features like deaths and time affect the model's performance, as they explain the disease's incidence among unvaccinated individuals. However, noise and redundant features, such as month, day, and is_weekend, can distort the model's direction, hindering its understanding in worst-case scenarios. Low-variance or irrelevant features may reduce the model's overall effectiveness. The method is therefore used for classification or regression, selecting the most useful features for capturing the evolution of cases.

Additionally, hyperparameters like the number of neighbors (k) play a pivotal role in KNN. A smaller k value increases sensitivity to local patterns, which can improve predictions for highly non-linear data but may make the model prone to noise. Conversely, a larger k smoothens predictions by averaging over a wider neighborhood, reducing variance but potentially overlooking finer details. In KNN, identifying relevant and meaningful features is critical, as the algorithm relies on calculating distances in the feature space. Features such as deaths_new, cases_active, and deaths_unvax are particularly important, as they provide direct insights into mortality trends, active case counts, and vaccination status, which are key indicators for predicting new cases (cases_new). Temporal features like month, day, and is_weekend are valuable for capturing seasonality and behavioral trends that influence case numbers. However, irrelevant or redundant features can introduce noise, adversely affecting model performance by skewing distance calculations. Features with low variance or minimal correlation to the target variable contribute little to predictive accuracy. Additionally, multicollinearity, high correlation among features, can distort the model's overall effectiveness.

The choice of distance metric, such as Euclidean or Manhattan, also impacts the algorithm's ability to measure similarity effectively. Euclidean distance is commonly used but may not perform well if features have different scales, making normalization essential. Weighting schemes, such as uniform weights or distance-based weights, influence how much impact each neighbor has on the prediction. Using distance-based weights ensures closer neighbors have a greater impact, which can improve accuracy. The hyperparameters in KNN play a pivotal role in its performance. Another important consideration is feature scaling (e.g., normalization or standardization), since KNN is highly sensitive to the scale of input features. Without proper scaling, features with larger ranges may dominate distance calculations, reducing the impact of other, potentially more important, features. Cross-validation is employed to test different values of k and distance metrics, ensuring the chosen hyperparameters balance bias and variance effectively.

In conclusion, the performance of KNN depends heavily on the selection of relevant features and the tuning of hyperparameters such as k and the distance metric. Proper preprocessing, including feature scaling and normalization, is essential to ensure accurate distance calculations and meaningful predictions. By leveraging the most relevant features and optimizing hyperparameters, KNN can deliver reliable and interpretable results for predicting new COVID-19 cases.

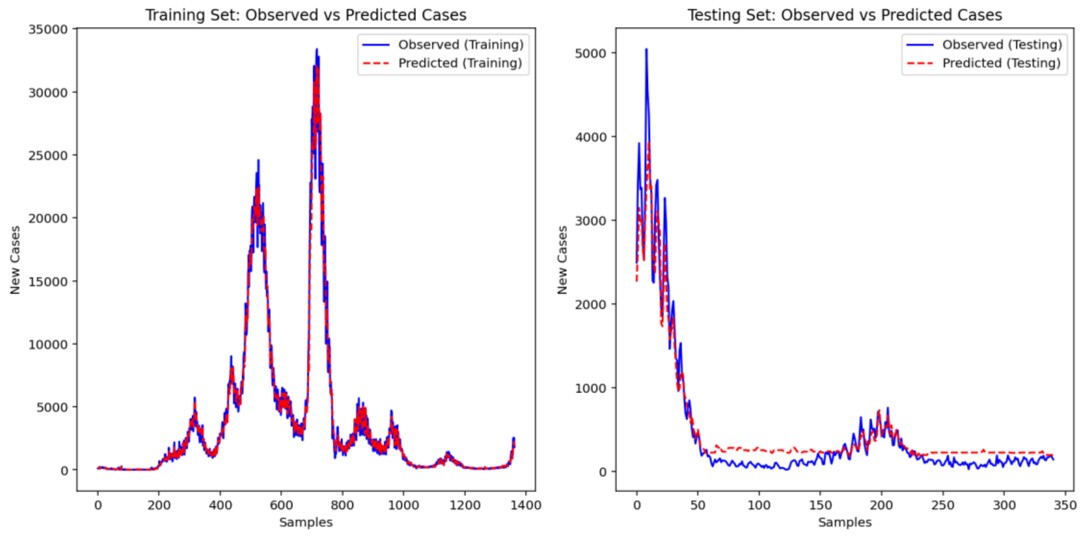


Figure 29.
Training and Testing for KNN.

7.4.4. Linear Regression

The simple yet highly sensitive Linear Regression algorithm is described below in relation to its predictive accuracy. The performance of the model in Linear Regression depends on the selection of relevant features and the tuning of hyperparameters, which can significantly influence its accuracy. Features such as deaths, active cases, and deaths among unvaccinated individuals are likely the most important predictors, as they have a linear relationship with the dependent variable (cases_new) and can capture COVID-19 trends, including seasonal effects, mortality, and active cases. Temporal features like month, day, and is_weekend are also valuable for capturing patterns in disease prevalence and severity that vary over time. Since linear regression models assume that the relationship between independent variables and the dependent variable is linear, it is important to remove highly correlated or redundant features, as they may lead to multicollinearity, which can result in unreliable estimates.

Hyperparameter tuning of the model's coefficients is not typically necessary for Linear Regression, as the model is simple and its coefficients are not complex. However, it is possible to improve model generalization by using techniques such as Ridge (L2 regularization) and Lasso (L1 regularization), which help prevent overfitting, especially when there are many features or when the data is noisy. These techniques are added to the cost function as a penalty term. Ridge regularization penalizes the sum of the squared coefficients, while Lasso regularization penalizes the absolute value of the coefficients. The strength of regularization is controlled by the alpha parameter, which determines the amount of penalty applied during training. With the alpha value, the model can find a balance that fits the training data well and also generalizes effectively to unseen data.

The performance of the Linear Regression model is usually checked using measures like Mean Squared Error (MSE) or R-squared (R^2). These values help assess how accurate the predictions are and how much variance in the data is explained by the model. Cross-validation is used to identify the optimal alpha value that minimizes error and avoids overfitting. In short, with proper feature selection and appropriate application of regularization, Linear Regression can be a very useful method for predicting new COVID-19 cases.

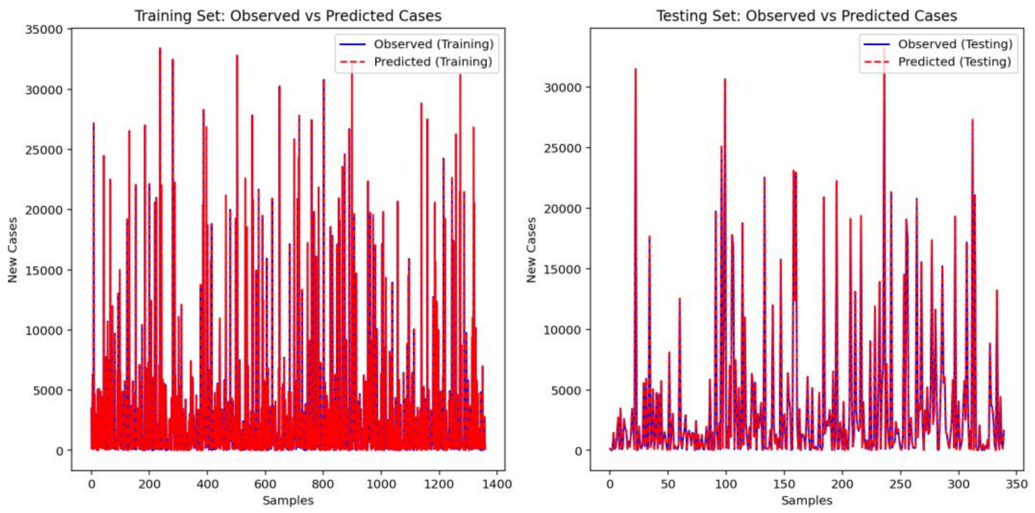


Figure 30.
Training and Testing for Linear Regression.

7.4.5. Random Forest

Random Forest is a type of algorithm that makes decisions based on decision trees by relying heavily on the selection of input features and fine-tuned hyperparameters. For instance, using suitable input features like cases_active and deaths_new is crucial, as it shows information such as the current state of the pandemic. This step allows the model to understand mortality trends, current active cases, and the vaccination status of patients. Next, temporal features like day, month, and is_weekend are also useful as they help in capturing human behaviors and seasonal patterns of COVID-19 cases. Including critical features is also beneficial in training and testing a Random Forest algorithm, as it exhibits high variance and strong predictive value for the target variable. Additionally, feature selection techniques can be incorporated into this algorithm to prevent unnecessary features, such as noise, from affecting the model's performance.

The key hyperparameters in a random forest are the number of trees, the maximum depth of each tree, and the number of features considered when splitting each node. The larger the dataset, the higher the number of trees needed. This helps control the overall size of the ensemble and improves the model's performance by reducing overfitting issues, which can lead to high variance. Moving on, the maximum depth of each tree limits the growth of individual trees by controlling the model's complexity. Using an appropriate value for the depth of each tree helps avoid building a model that is too complex and computationally intensive. For example, if the maximum depth is too deep, the model may overfit. Conversely, if the maximum depth is too shallow, the model may underfit.

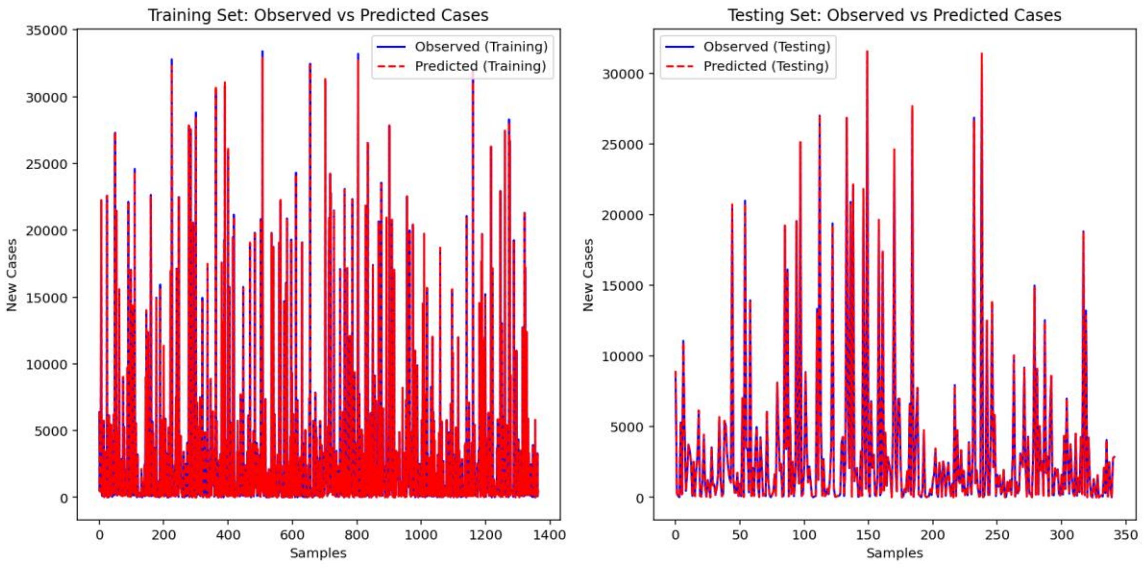


Figure 31.
Training and Testing for Random Forests.

7.4.6. LSTM

The feature selection and suitable hyperparameter choice are the core of an LSTM, especially for time series prediction. Variables like “deaths_fvax” and “deaths_unvax” might influence the target variable, and the relevance of the selected features is directly related to how well the model can learn from hidden and complex patterns. By using the right features, LSTM could easily achieve a deep understanding of predicting COVID-19 cases. Other than that, temporal features like “day,” “month,” and “week_of_year” help to capture seasonal and cyclic patterns, while “is_weekend” reflects the changes in the behavioral pattern of testing trends and population mobility. The "cases_active" is considered important as it allows the model to learn short-term dynamics relevant for time-sensitive predictions.

Moving on, MinMaxScaler is a tool used to normalize the features of datasets, as it ensures that all variables are scaled to a similar range, enhancing the efficiency with which the model learns. Hyperparameters such as the number of timesteps will determine how much history is included in each prediction. For example, using the information from the past 10 days as the timestep. Next, the implementation of the right number of layers and neurons is crucial as it determines the capability of the LSTM to learn various complex relationships. A two-layer structure with 128 and 64 units allows the model to understand high-level and finer-grained temporal patterns of the datasets. Additionally, dropout layers prevent issues like overfitting during training by randomly shutting off some neurons, allowing the model to generalize better to new and unseen data.

The "Adam" optimizer is included as it defines how much the model can update its weights during the training process. There is a trade-off between computational efficiency and convergence speed. The loss function and mean squared error are more effective for regression tasks since they penalize larger errors more severely. This combination of features has clearly performed well on LSTM, as shown in Figure 32 below. The figure displays a graph comparing the actual COVID-19 values versus the predicted values using the LSTM model. It can be observed that the model has effectively captured the trends but tends to underestimate the peaks. Therefore, it can be concluded that the model is highly suitable for time-series prediction, with only minor deviations during volatile periods.

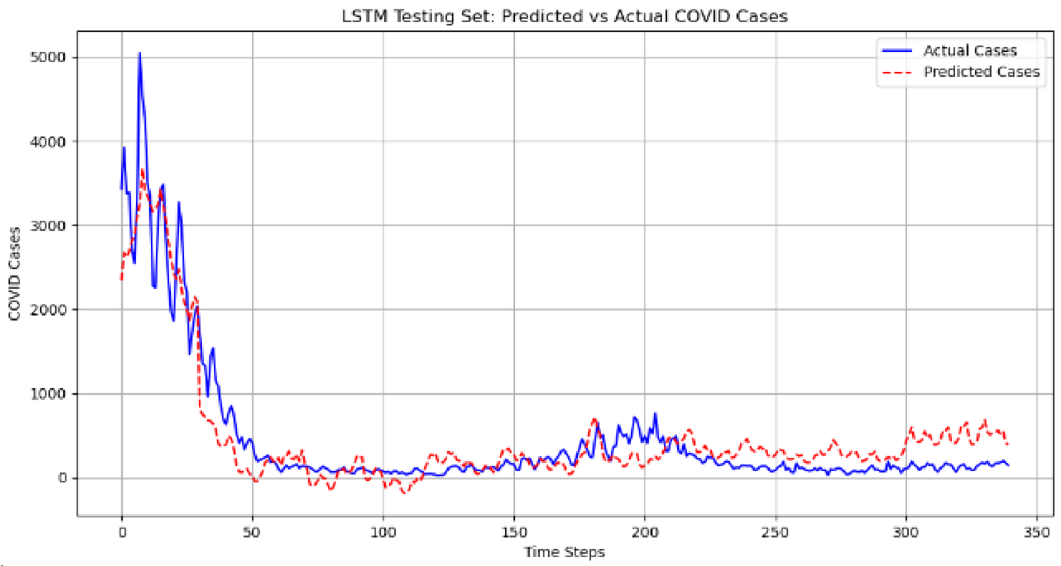


Figure 432.
LSTM Testing Data against Actual COVID Cases.

7.4.7. XGBoost

This model was configured carefully by using a selected set of parameters to optimize its performance for the regression task. The objective parameter was set to "reg:squarederror" as it helps to ensure that the model is tailored to predict continuous numerical values. One of the evaluation metrics used was Root Mean Squared Error (RMSE), which calculates the average magnitude of errors between the actual and predicted values. A lower RMSE reflects higher model performance and serves as a key metric for early stopping and determining the optimal number of boosting rounds.

The performance of XGBoost heavily depends on the relevance of the input features. For instance, variables like deaths_new and cases_active are important because they directly reflect trends in active cases and population mortality rates, which are key predictors of new COVID-19 cases. Next, the temporal features like "month," "day," and "is_weekend" capture seasonal patterns, providing valuable context for the model to produce more reliable and accurate predictions. By utilizing these features, this model can easily detect shifts in case trends and align predictions with real-world factors, especially public holidays. It is necessary to identify redundant or irrelevant features, as they may introduce noise that will influence the model's predictive accuracy.

Moving on, the configuration of hyperparameters in XGBoost helps to balance the model's capability in understanding and learning the dataset. The number of maximum tree depth is one of the critical factors as it directly affects the model's complexity and generalization capabilities. As shown in the figure below, the relationship between tree depth and R^2 scores reveals that deeper trees will maximize the model's performance. It is shown that the training and testing R^2 results improve significantly up to a depth of 4-6, where the testing R^2 plateaus. Beyond this range, the training R^2 approaches 1.0, indicating a perfect training fit, while the testing R^2 shows minimal or no improvement, signaling overfitting.

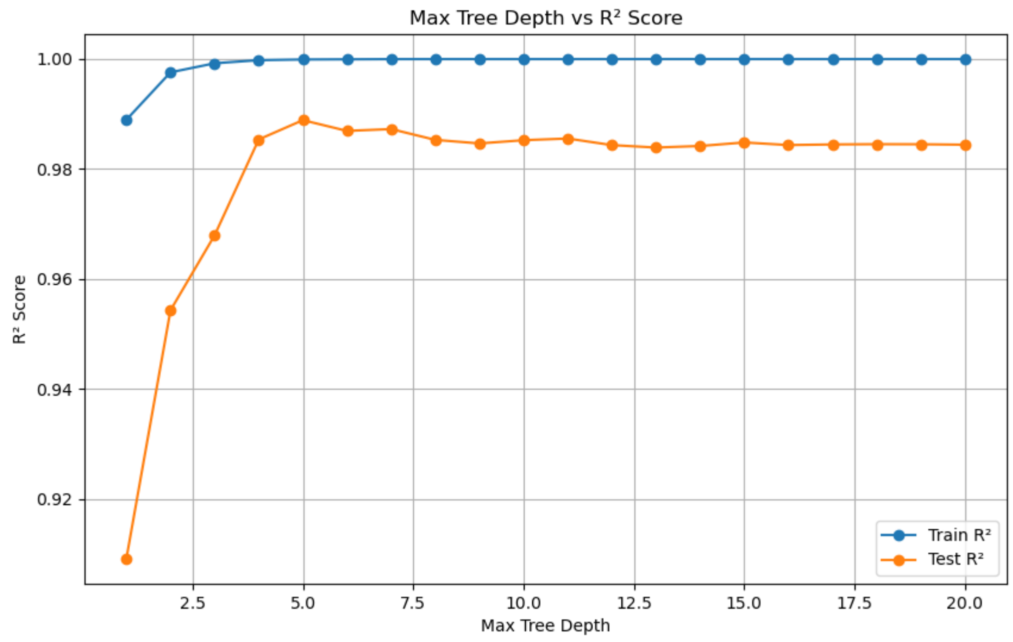


Figure 33.
Max-depth Against R^2 Score.

Hence, the maximum tree depth was capped at 6 as it achieves an optimal balance between capturing complex patterns while avoiding overfitting. The model is able to learn gradually by setting the learning rate to 0.1, as it ensures a stable learning progression while avoiding the risk of overfitting. This comprehensive tuning of hyperparameters allows XGBoost to generalize well while exploiting the depth of trees to obtain the complicated patterns without succumbing to noise or overfitting. Other than that, the subsampling and column sampling rates were set to 0.8, as it allows each boosting round, and the tree utilises random subsets of data and features. This randomness not only reduces overfitting but also enhances model robustness and diversity.

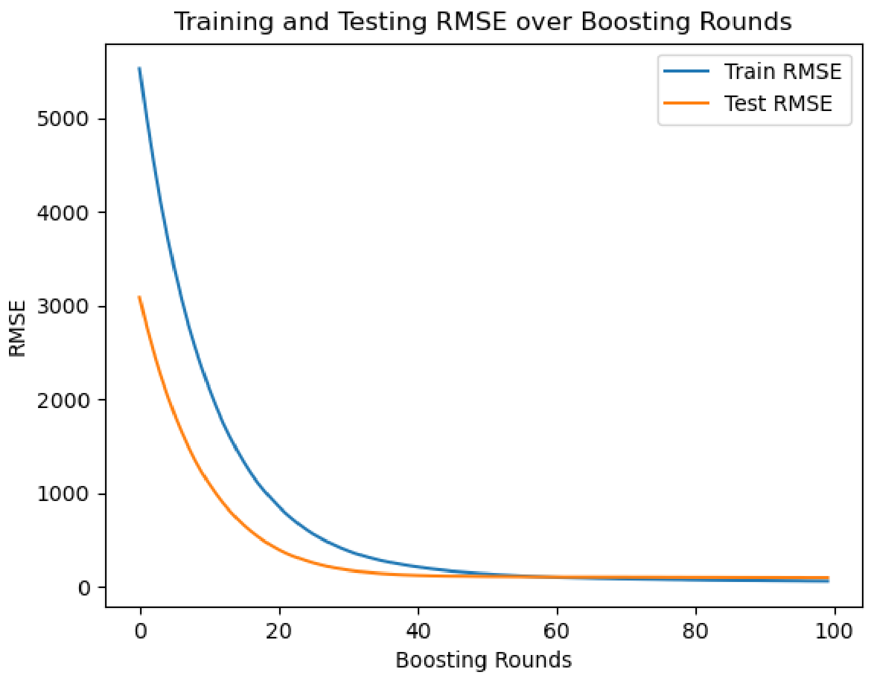


Figure 34.
XGBoost Training and Testing RMSE.

As shown in Figure 34, it reveals the relationship between the number of boosting rounds and the Root Mean Squared Error (RMSE) for both training and testing datasets in the model. Initially, RMSE decreases rapidly as the model learns from the data, with significant improvements observed in the first 40-60 boosting rounds. After this point, the reduction slows down, and the curves have turned flat, showing that the diminishing gains from additional boosting. While the training RMSE continues to decrease, the testing RMSE stabilizes, highlighting the importance of early stopping to prevent overfitting. This balance ensures optimal model generalization without compromising its ability to capture underlying patterns.

GitHub Link
https://github.com/MafiaBossQQ/CovidPredictionDM.git
YouTube Link
https://youtu.be/6qzhKbLZbCE?si=PmI0EC4gqRgpxbSk

7.5. Research Directions and Future Work

Future research can be used to address various types of limitations in this study, which helps to enhance the predictive accuracy and robustness. One of the promising directions is the development of hybrid models by combining the strengths of each deep learning technique, for instance, ANNs and CNNs. These hybrid models could increase the accuracy of predicting COVID-19 cases. Next, the creation of real-time prediction frameworks can also be considered, as it promotes the processing of streaming data. Such systems allow timely and accurate predictions during an ongoing pandemic, making them an important utility for public health interventions. Transfer learning is also one of the methods that can be applied in this study, as it offers pretrained models to be fine-tuned for certain tasks, especially when dealing with smaller datasets.

Based on the results of each model, several areas of investigation can be further explored to improve the overall applicability and performance of the predictive models used. By increasing the number of training epochs for models like ANNs and CNNs, the convergence rate will be increased, provided the regularization techniques are applied to avoid issues like overfitting. Moving on, other machine learning models like Long Short-Term Memory (LSTM) and Gated Recurrent Units (GRU) can be implemented due to their capability in time-series forecasting. Next, utilizing large datasets by including different features like regional variations and different phases of the pandemic will significantly improve the generalization and predictive accuracy of the models.

Other than that, advanced feature engineering by considering external factors like vaccination rates, government policy changes, or even weather conditions, which could highly influence the COVID-19 trends. Explainable Artificial Intelligence (XAI) methods should be incorporated into these predictive models as they help to ensure the results are interpretable for healthcare professionals and policymakers. It is a method that allows human users to understand and trust the output created by machine learning algorithms. Furthermore, fine-tuning the hyperparameters for models like Random Forest and XGBoost helps to optimize each model's performance. For instance, adjusting the number of trees, learning rate, and kernel size. By considering and addressing all these directions in future work, the reliability and applicability of the predictive models could be improved.

8. Conclusion

In summary, the COVID-19 pandemic causes various challenges to the global healthcare systems, necessitating accurate prediction models for daily new cases to effectively plan interventions, allocate necessary resources, and improve public health policies to achieve specific healthcare goals. It is crucial to understand the current situation and predict COVID-19 cases, as it helps mitigate the impact of the virus on public health and economic stability. This study includes a comprehensive dataset collected from three different sources: regional COVID-19 statistics, vaccination data, and demographic information. The raw dataset has been pre-processed to make it suitable for further analysis. For instance, missing values were addressed, inconsistent entries were resolved, and outliers were managed to ensure data integrity.

In order to address the challenges of predicting COVID-19 cases with high reliability, various machine learning models were employed and evaluated. The models include Artificial Neural Networks (ANN), Convolutional Neural Networks (CNN), Linear Regression, Random Forest, XGBoost, K-Nearest Neighbors (KNN), and Long Short-Term Memory (LSTM). These models were selected due to their ability to handle high-dimensional, time-series, and tabular datasets. The main objective of this study is to determine which model is most suitable for predicting daily new COVID-19 cases by developing a robust predictive framework. This process involved the identification of relevant features through feature importance analysis. This aims to maximize the predictive performance of each model, evaluate the impact of feature selection, and record the results of each machine learning technique to identify the most effective model.

Based on the R^2 values alone, the ANN model emerged as the most effective among the tested methods, achieving an impressive R^2 score of 0.9614, the lowest RMSE, and competitive MAPE values. However, when all four performance metrics are considered, Linear Regression outperformed other models, including ANN. Other than that, the best-performing feature subset excluded variables such as "quarter," "is_weekend," "deaths_vax," "day," and "holiday." This highlights the importance of feature selection analysis in boosting each model's performance by removing less impactful variables, resulting in a more robust framework for capturing the underlying and complex patterns while effectively explaining variability in COVID-19 case trends.

As a result, this research lies in its potential to support current pandemic management through accurate prediction of daily new COVID-19 cases, allowing more informed decisions on resource allocation, vaccination strategies, and containment measures. Beyond COVID-19, the methodologies and insights can be adapted for other public health challenges, emphasizing the transformative potential of data mining and machine learning in addressing global crises and fostering resilience in healthcare systems.

Transparency:

The author confirms that the manuscript is an honest, accurate, and transparent account of the study; that no vital features of the study have been omitted; and that any discrepancies from the study as planned have been explained. This study followed all ethical practices during writing.

Copyright:

References

[1] V. K. R. Chimmula and L. Zhang, "Time series forecasting of COVID-19 transmission in Canada using LSTM networks," *Chaos, Solitons & Fractals*, vol. 135, p. 109864, 2020. https://doi.org/10.1016/j.chaos.2020.109864

[2] F. Rustam *et al.*, "COVID-19 future forecasting using supervised machine learning models," *IEEE Access*, vol. 8, pp. 101489-101499, 2020. https://doi.org/10.1109/ACCESS.2020.2997311

[3] S. Tuli, S. Tuli, R. Tuli, and S. S. Gill, "Predicting the growth and trend of COVID-19 pandemic using machine learning and cloud computing," *Internet of Things*, vol. 11, p. 100222, 2020. https://doi.org/10.1016/j.iot.2020.100222

[4] R. Ahuja, P. Jain, A. Sureka, and V. Banga, "Prediction of COVID-19 cases using machine learning algorithms," *Materials Today: Proceedings*, vol. 45, pp. 3478–3482, 2021.

[5] Z. Hu, Q. Ge, S. Li, and L. Jin, "Random forest-based prediction of COVID-19 transmission in China," *Journal of Infection and Public Health*, vol. 14, no. 4, pp. 504–511, 2021.

[6] L. Yan *et al.*, "A machine learning-based model for survival prediction in patients with severe COVID-19," *Computers in Biology and Medicine*, vol. 129, p. 104163, 2021.

[7] F. Petropoulos and S. Makridakis, "Forecasting the novel coronavirus COVID-19," *PLoS One*, vol. 15, no. 3, p. e0231236, 2020. https://doi.org/10.1371/journal.pone.0231236

[8] S. Altay, S. Karasu, and A. Altan, "Comparative study of COVID-19 forecasting models: ARIMA, Prophet, and hybrid models," *Journal of Forecasting*, vol. 41, no. 3, pp. 512–529, 2022.

[9] C. Xu, J. Zhang, H. Zhang, and H. Tang, "Using deep neural networks for predicting COVID-19 transmission," *International Journal of Data Science and Analytics*, vol. 14, no. 1, pp. 77–91, 2022.

[10] J. Zhang, L. Chen, and W. Xu, "Models for COVID-19 data prediction based on improved LSTM-ARIMA algorithms," *IEEE Access*, vol. 12, pp. 3981–3991, 2024.

[11] T. T. Mengistie, "COVID-19 outbreak data analysis and prediction modeling using data mining technique," *International Journal of Computer*, vol. 38, no. 1, pp. 37-60, 2020.

[12] S. Solayman, S. A. Aumi, C. S. Mery, M. Mubassir, and R. Khan, "Automatic COVID-19 prediction using explainable machine learning techniques," *International Journal of Cognitive Computing in Engineering*, vol. 4, pp. 36-46, 2023. https://doi.org/10.1016/j.ijcce.2023.01.003

[13] S. Ghafouri-Fard, H. Mohammad-Rahimi, P. Motie, M. A. Minabi, M. Taheri, and S. Nateghinia, "Application of machine learning in the prediction of COVID-19 daily new cases: A scoping review," *Heliyon*, vol. 7, no. 10, p. e08143, 2021.

[14] W. A. Awadh, A. S. Alasady, and H. I. Mustafa, "Predictions of COVID-19 spread by using supervised data mining techniques," *Journal of Physics: Conference Series*, vol. 1879, no. 2, p. 022081, 2021. https://doi.org/10.1088/1742-6596/1879/2/022081

[15] A. Y. Allmuttar and S. K. Alkhafaji, "Using data mining techniques deep analysis and theoretical investigation of COVID-19 pandemic," *Measurement: Sensors*, vol. 27, p. 100747, 2023. https://doi.org/10.1016/j.measen.2023.100747

[16] F. Ahouz and A. Golabpour, "Predicting the incidence of COVID-19 using data mining," *BMC Public Health*, vol. 21, p. 1087, 2021. https://doi.org/10.1186/s12889-021-11058-3

[17] N. I. S. A. Satar, A. Mohamed, and A. M. Ali, "Data mining techniques for pandemic outbreak in healthcare," *JOIV: International Journal on Informatics Visualization*, vol. 5, no. 2, pp. 162-169, 2021. http://dx.doi.org/10.30630/joiv.5.2.548

[18] L. Muhammad, M. M. Islam, S. S. Usman, and S. I. Ayon, "Predictive data mining models for novel coronavirus (COVID-19) infected patients' recovery," *SN Computer Science*, vol. 1, p. 206, 2020. https://doi.org/10.1007/s42979-020-00216-w

[19] R. Rane, A. Dubey, A. Rasool, and R. Wadhvani, "Data mining based techniques for COVID-19 predictions," *Procedia Computer Science*, vol. 218, pp. 210-219, 2023. https://doi.org/10.1016/j.procs.2023.01.003

[20] S. NOOR, W. Akram, and T. Ahmed, "Predicting COVID-19 incidence using data mining techniques: A case study of Pakistan," *BRAIN. Broad Research in Artificial Intelligence and Neuroscience*, vol. 11, no. 4, pp. 168-184, 2021.