Indian Institute of Technology Madras Zanzibar Campus

Shaastra Techathon 2024 - AI ML Challenge

Project Report

zda23b019@iitmz.ac.in

Manas R Pandya

May 2024

1 Introduction

This report details my participation in the Shaastra Techathon 2024, hosted by the Indian Institute of Technology Madras. I, too, am actually the part of IITM, but of the *Zanzibar* campus.
This Techathon has challenged participants to develop innovative solutions in the field of Artificial Intelligence and Machine Learning, addressing real-world problem of covid death prediction through AI solutions. This event has not only tested technical skills but also contributes to our understanding of how AI and ML can be leveraged to solve practical issues in various domains.

2 Understanding the Problem Statement

2.1 Problem Description

The competition revolves around predicting deaths caused by COVID-19. It is based on a myriad of indicators and variables provided in a dataset. The dataset includes the following colomns:

Sno	Date	Time	State/UnionTerritory

ConfirmedIndianNational	ConfirmedForeignNational

Cured	Deaths	Confirmed	OxygenTanks

PopulationDensityPerSqKm

The challenge involves using this data to forecast future trends of death rates, which were crucial for planning and resource allocation during the pandemic.

2.2 Objectives

The primary objectives of this project were to:

- Implement feature engineering to enhance the model's performance by extracting and selecting significant predictors from the data.
- Develop a predictive (machine learning) model that can accurately forecast the number of deaths from COVID-19 on a daily basis, using the features available.
- Explore the various techniques and models to find the most effective solution for the given problem.

3 Data and Feature Engg.

We made a lot of changes to the existing data.

- Firstly, we decided to skip using the columns `ConfirmedIndianNational` & `ConfirmedForeignNational`. We did this, as the data points for all but a few dozen rows for these columns were empty.
- After initial experiments with existing data we realised we needed to experiment with various modifications of the data available to us.

These experiments have been discussed in detail below:

3.1 Feature Experiments

We created various modified versions of features to capture the intricacies of the data better than using them as given, for predicting the number of deaths:

- **Temporal Features:**
  We added temporal features like `DayOfYear`, `DayOfWeek`, and `Month` to account for seasonal trends and weekly patterns in deaths.
- **Lag Features:**
  We created lag features for both `Confirmed` and `Cured` cases to **help capture the temporal dependencies** in the data. For example, `Confirmed_lag_1` and `Cured_lag_1` represent the number of confirmed and cured cases from one day prior, respectively.
- **Interaction Features:**
  Interaction terms allow us to capture combined effects between features. For example, `Confirmed_Cured_Interact` represents the product

of `Confirmed` and `Cured` cases, highlighting interactions between them.

- **Rolling Mean and Rolling Standard Deviation:**
  Rolling statistics are useful for capturing trends and variability over a specific window. `Confirmed_Rolling_Mean/Std` and `Cured_Rolling_Mean/Std` are rolling averages and standard deviations of confirmed and cured cases, respectively.

- **Exponentially Weighted Moving Average (EWMA):**
  EWMA is another type of moving average that **gives more weight to recent observations**. We used `Confirmed_EWMA` and `Cured_EWMA` to capture the recent trends in confirmed and cured cases.

- **Categorical Feature:**
  `State/UnionTerritory` is a categorical feature representing the states or territories of India.
  Although we let it remain unchanged, we **used one-hot encoding** to use it as a feature as well.

By experimenting with these different features, **amongst others**, we aimed to capture the various factors influencing COVID-19 deaths. After thorough experimentation (as given in the subsequent sections), we narrowed down the final set of features that provided the best predictive performance.

3.2 Analysis of the data

To find out how much each feature is predictive of the target variable, `Deaths`, we plotted various graphs to see the correlation, like shown in Figure 1 (a high correlation) and 2 (a low correlation). To further speed up the process of

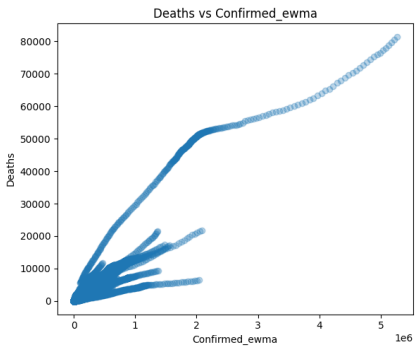

Figure 1: **Example** graph to show the (high) correlation of `Deaths` with `Confirmed_ewma`.

comparing features for better correlation with `Deaths`, we calculated correlation using the ***Pearson Correlation Coefficient*** , as shown in Table 1

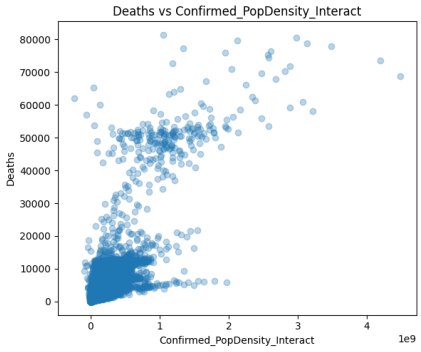

Figure 2: **Example** graph to show the (low) correlation of `Deaths` with `Confirmed_PopDensity_Interact`.

Feature	Correlation
Deaths	1.000000
Confirmed_lag_12	0.912652
Confirmed_lag_11	0.912639
Confirmed_lag_13	0.912580
Confirmed_ewma	0.909764
Confirmed	0.907696
Cured	0.905926
Cured_ewma	0.905650
Cured_lag_11	0.901575
Cured_lag_12	0.900733
Cured_lag_13	0.899825
Confirmed_PopDensity_Interact	0.827552
Confirmed_Cured_Interact	0.788709
Sno	0.268309
cos_day_year	0.159071
sin_day_year	0.020486
PopulationDensityPerSqKm	0.003340
DayOfWeek	0.001666
OxygenTanks	-0.006296
Month	-0.016089
DayOfYear	-0.017089

Table 1: **Pearson Correlation Correlation** of Features with Deaths

3.3 Deductions from the Analysis

We experimented with different features and analysing their correlations, and hence the predictivity, of `Deaths`. After testing some of our feature combinations, we finalised on the following features for our final model:

- `Confirmed_lag_12`
- `Cured_lag_12`
- `State/UnionTerritory`
- `Confirmed_Rolling_Mean`
- `Cured_Rolling_Mean`
- `Confirmed_Rolling_Std`
- `Cured_Rolling_Std`
- `Confirmed_EWMA`
- `Cured_EWMA`

4 Experimental Setup

4.1 Models Explored

- **Machine Learning Models:** We applied traditional machine learning models including Random Forest, SVM, and Decision Trees. These models produced a Modified Mean Absolute Percentage Error (MMAPE) in the range of 30,000 to 50,000, which indicated a need for more complex approaches.

- **DNN Models:** Deep Neural Networks were explored next. They performed better than the traditional machine learning models but still did not meet our expectations for accuracy.

- **RNN Models:** We started with simple Recurrent Neural Networks (RNNs), then advanced to more complex structures including Long Short-Term Memory (LSTM) networks, Gated Recurrent Units (GRU), and Convolutional Neural Networks (CNN) integrated into RNN architectures. We concluded with a focus on Bidirectional LSTM and Sequence-to-Sequence LSTM models due to their superior performance in handling time-series data.

After experimenting with various models, we finalized on the Sequence-to-Sequence LSTM model for its superior performance and ability to handle complex temporal dependencies.

4.2 Model Evaluation

Initially, we employed default metrics such as Mean Absolute Error (MAE) and Root Mean Squared Error (RMSE), which showed good correlation with the competition's custom metric, MMAPE. After narrowing down our model choices, we switched to using MMAPE to directly optimize for the competition's evaluating criteria.

4.3 Hyperparameter Tuning

For the machine learning models, we utilized Grid Search for hyperparameter tuning, although this approach could have been optimized further. For our final model, the Sequence-to-Sequence LSTM, we employed Bayesian optimization to fine-tune the parameters effectively.

5 Final Model Architecture

5.1 Architecture Description

The final model architecture (as depicted in Figure 3) is designed specifically for time-series forecasting, utilizing the Sequence-to-Sequence (Seq2Seq) framework with Long Short-Term Memory (LSTM) cells. This architecture is particularly suited for problems where the input and output are sequences of potentially different lengths.

Input Layer: The model begins with an input layer that takes in sequences from the dataset. Each sequence corresponds to a set of features derived from the data.

Reshape Layer: Following the input, there is a reshape layer that adjusts the dimensions of the input data to be compatible with the LSTM layers.

LSTM Encoder: The first LSTM layer acts as the encoder component of the Seq2Seq model. This layer processes the entire input sequence and encodes it into a fixed-size internal representation. This representation captures the temporal dependencies within the input data, which is crucial for predicting future outcomes based on past data.

Repeat Vector: The output of the encoder is then repeated multiple times to match the length of the output sequence. This repetition is necessary to connect the encoder and decoder components, allowing the decoder to generate the output sequence step-by-step.

LSTM Decoder: The second LSTM layer is the decoder part of the Seq2Seq model. It takes the repeated vector as input and generates the output sequence one step at a time. The decoder learns to predict the next step in the sequence based on the previous steps and the encoded information passed from the encoder.

Time Distributed Dense Layer: The final layer is a fully connected Dense layer that is applied to each time step independently. This layer converts the output of the decoder LSTM into the final prediction for each time step in the output sequence.

The Seq2Seq model enhances this by allowing for the input and output sequences to have different lengths, providing flexibility and making it ideal for predicting COVID-19 deaths.

5.2 Reasoning Behind the Architecture

The choice of a Sequence-to-Sequence (Seq2Seq) LSTM architecture was driven by its established proficiency in handling time-series data. Equipped with LSTM units, it is adept at understanding the entire sequence of inputs. This capability makes it exceptionally suitable for forecasting COVID-19 death rates **where past context significantly influences future outcomes.**

The architecture specifically utilizes two LSTM layers—one as an encoder and the other as a decoder. This design helps in compressing the input sequence into a dense representation, which is then used to generate the output sequence, thereby facilitating better learning of temporal dependencies.

Additionally, we employed **Bayesian optimization** to fine-tune the hyperparameters, such as the number of units in the LSTM layers. This method helped us systematically explore the hyperparameter space and optimize the model's performance by finding an ideal configuration that balances computational efficiency with predictive accuracy.

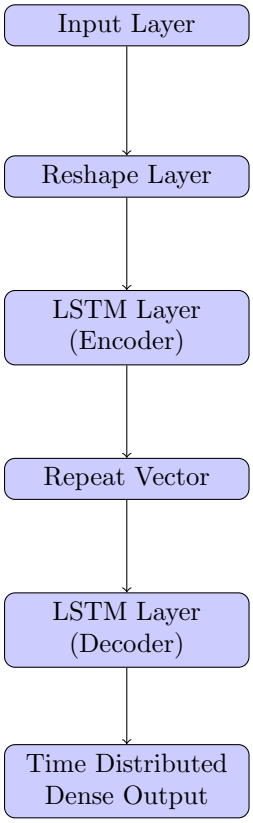

Figure 3: Flowchart of the Sequence-to-Sequence LSTM Model

5.3 Experimental Details

During our experimentation, we experimented with various number of units in LSTM layers from 64 to 256 to observe the impact on model performance. This range was chosen based on preliminary tests that indicated it provides **a good trade-off between model complexity and performance.**

The final model uses two LSTM layers optimized through **Bayesian optimization**, which proved effective in refining the model by adjusting hyperparameters systematically, **better than manual guesswork or simple grid searches**.

5.4 Results

The final model configuration achieved a Root Mean Squared Error (RMSE) of 39 and a custom Modified Mean Absolute Percentage Error (MMAPE) of 1,462.

These metrics not only help us understand the model's robustness, but also its superior forecasting ability compared to earlier machine learning models we tested, which had significantly higher errors. The improvement in MMAPE is particularly noteworthy as it reflects a substantial enhancement in the model's accuracy and reliability in predicting actual death figures.

6 Shortcomings & Conclusion

Despite our successes, our approach had limitations primarily due to our nascent experience in the field of machine learning. Initially, **we underestimated the effectiveness of traditional machine learning models for structured data**, pivoting too quickly to deep learning solutions.

In hindsight, a more balanced exploration of machine learning techniques might have yielded even better results. However, our journey through this competition has significantly bolstered our understanding and skills in both domains, setting a robust foundation for future challenges.

Despite these shortcomings, our experience in the Shaastra Techathon has been immensely rewarding, providing us with valuable hands-on experience in tackling real-world problems using AI and ML technologies.