Analysis of the Results of the LSTM Forecasting System for COVID-19 Cases Prediction in the Canary Islands

By Ginevra Iorio

Abstract

In this paper an analysis of the performance of the Long Short-Term Memory (LSTM) forecasting system built for predicting COVID-19 cases in the Canary Islands will be presented. The accuracy of the predictions made will be evaluated using two commonly used metrics, namely Root Mean Squared Error (RMSE) and Mean Absolute Percentage Error (MAPE). The findings suggest that the LSTM model may be a valuable tool for decision-makers in managing the COVID-19 pandemic.

Introduction

Long Short-Term Memory (LSTM) models, which have demonstrated high accuracy in time series forecasting, have been previously applied to predict COVID-19 cases in Canary Islands. Their evaluation has been performed using the Mean Squared Error (MSE) criterion so far. However, this is not the only suitable criterion for evaluating the performance of time series models.

Therefore, the main objective of this paper is to introduce two new evaluation metrics, namely Mean Absolute Percentage Error (MAPE) and Root Mean Square Error (RMSE) and conduct a comparative study of their performance with MSE of the LSTM-based forecasting. The MAPE metric is particularly useful in situations where there are significant differences between the magnitudes of the predicted and actual values. In contrast, the RMSE metric is a standard deviation of the differences between predicted and actual values, providing an estimate of the spread of the error distribution.

To achieve the objective, MAPE and RMSE metrics are introduced into the Python code and applied to fit the LSTM model. Based on the results obtained from the evaluation of the model with these criteria, adjustments are made to further improve the accuracy of the model.

LSTM Models Evaluation Methods

There are several evaluation methods that can be used to assess the performance of LSTM models:

- Mean Squared Error (MSE): it is a commonly used evaluation metric for LSTM models that measures the average squared differences between the predicted and actual values. A lower value of MSE indicates better performance of the model.
- Root Mean Squared Error (RMSE): it is the square root of MSE and provides an estimate of the standard deviation of the differences between the predicted and actual values.
- Mean Absolute Error (MAE): it measures the average absolute differences between the predicted and actual values. It provides a measure of the magnitude of the errors, but it does not take into account their direction.

- R-squared (R2): it is a measure of the proportion of variance in the dependent variable that is explained by the independent variables. It ranges from 0 to 1, with higher values indicating a better fit of the model to the data.
- Mean Absolute Percentage Error (MAPE): it measures the average percentage differences between the predicted and actual values. It is commonly used when the magnitudes of the predicted and actual values are significantly different.
- Symmetric Mean Absolute Percentage Error (SMAPE): it is similar to MAPE but takes into account the direction of the errors. It is calculated as the average of the absolute percentage differences between the predicted and actual values, divided by the sum of the predicted and actual values.

The choice of the evaluation method depends on the specific requirements of the forecasting task and the nature of the data. It is often recommended to use multiple metrics to evaluate the performance of LSTM models and compare their results [1]. So far, the model has been evaluated through MSE in the experimentation phase. Through the evaluation, hyperparameters have been fixed and changed with the goal of obtaining the best results possible.

Two additional evaluation methods, MAPE and RMSE, will now be introduced, and a study on the model will be conducted comparing the different metrics. By using these evaluation methods, the aim is to provide a more comprehensive analysis of the performance of the LSTM models. Ways to modify and improve the models will also be explored, based on the obtained results.

Root Mean Square Error (RSME)

The Root Mean Square Error is a widely used metric for evaluating the accuracy of predictive models, particularly in the field of time series analysis. It measures the differences between the predicted values and the actual values, by calculating the square root of the average squared differences between the two.

$$RMSE = \sqrt{\overline{(f-o)^2}}$$

In other words, RMSE provides an estimate of the overall deviation of the predicted values from the actual values. A lower value of RMSE indicates that the model is more accurate in predicting the values of the target variable. It is expressed in the same units as the target variable, and its interpretation is similar to that of the standard deviation. It is an important metric for evaluating the performance of predictive models because it provides a measure of how well the model fits the data. It is particularly useful for time series analysis because it takes into account the order of the data points, which is important for predicting future values. However, RMSE can be sensitive to outliers and extreme values in the data, and it does not provide information on the direction of the errors (whether they are positive or negative) [2]. The most frequent issue with using this statistic is its susceptibility to outliers. A check for outliers has already been made though, so they have been taken out from the datasets and will not be found in the model [3].

Pros are that it is easy to understand, it is computationally straightforward and easily differentiable, which many optimization algorithms seek. Because of the square root, RMSE does not penalize mistakes as severely as MSE does. Besides the sensitivity to outliers, cons are that it is influenced by the size of the data so if the error's size grows, its magnitude also grows. When the test sample size grows, the RMSE also grows, so when computing the results using various test samples it is a problem [4].

Mean Absolute Percentage Error (MAPE)

MAPE stands for Mean Absolute Percentage Error. It is a commonly used evaluation metric for forecasting models that measures the average percentage differences between the predicted and actual values of a variable. It is calculated as the average of the absolute differences between the predicted and actual values, divided by the actual values, multiplied by 100 [5].

$$MAPE = \frac{100\%}{n}\sum_{i=1}^{n}\left|\frac{y_i - \hat{y}_i}{y_i}\right|$$

MAPE provides a measure of the accuracy of the forecasting model, considering the magnitude of the actual values. It is useful when the data has significant variations in magnitude, such as when dealing with data on different scales or units. A lower value of it indicates better performance of the forecasting model. However, it has some limitations, such as being sensitive to extreme values and outliers in the data. In this case, a check for outliers has already been made, and, if present, they have been taken out from the datasets.

Pros of using MAPE as evaluation method are that it is simple to comprehend since all mistakes are standardized on a common scale, and that its error estimates are expressed in terms of percentages, which are independent from the magnitude of the variables. Cons are that the positive errors are penalized less than the negative ones, so MAPE is biased in the comparison of the precision of prediction algorithms, since it will select one whose results are too low. Moreover, the denominator cannot be zero for the formula to be valid [4].

Tests

In order to conduct the evaluation, the Root Mean Squared Error and the Mean Absolute Percentage Error metrics are used to compile the model. Being a loss, the RMSE needs to be first imported from the Keras' library in order to be used as a metric.

```
model.compile(loss='mse', metrics=[RootMeanSquaredError(name='rmse'), 'mape'], optimizer=Adam(learning_rate=0.001))
```

Two more graphs are now printed during the execution: one representing the train and validation RMSE and one for the train and validation MAPE. These graphs will help out for the detection of overfitting and underfitting models and for the adjustment of the hyperparameters.

```
plot_rmse(history)
plot_mape(history)
```

The functions that are called that build the graphs also need to be added to the code. They are identical, except for the fact that one takes the RMSE and the other the MAPE values from the compiling history.

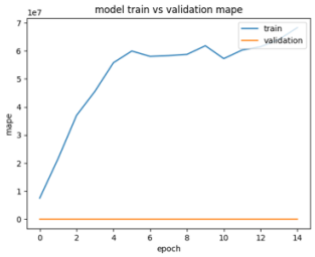
```python
def plot_rmse(history):
    plt.plot(history.history['rmse'])
    plt.plot(history.history['val_rmse'])
    plt.title('model train vs validation rmse')
    plt.ylabel('rmse')
    plt.xlabel('epoch')
    plt.legend(['train', 'validation'], loc='upper right')
    plt.show()
```

```python
def plot_mape(history):
    plt.plot(history.history['mape'])
    plt.plot(history.history['val_mape'])
    plt.title('model train vs validation mape')
    plt.ylabel('mape')
    plt.xlabel('epoch')
    plt.legend(['train', 'validation'], loc='upper right')
    plt.show()
```

The Python program is now run on all the different datasets, testing out all the implemented LSTM models. A study on the obtained RMSE and MAPE metrics' behavior during the training and validation phases is carried out by observing their graphs. A good result would involve decreasing metrics, with low values, and a slightly higher validation value. After various run, it was found that the results obtained for a specific dataset (for example, cases.csv) on the different LSTM models are similar. Therefore, there is no need to run a test for all the models on all datasets, but one single run using Vanilla LSTM is enough for this study.

While observing the graphs that are plotted running the program, there are some critical results. Firstly, some cases result in very high metrics' values which, as seen previously, are given by the small number of entries of the datasets, which only represent COVID data ranging from January 2020 to March 2022. A possible solution would be adding more layers to the LSTM cells or more input features. Moreover, in a few cases the MAPE seems to grow instead of decreasing:

eg. cases.csv – MAPE evaluation



The reason of this behavior is that MAPE is not a good metric when in the dataset compare one or more zeros. Observing its formula, it is visible that a zero in the data would end up at the denominator of a division, resulting in the Mean Absolute Percentage Error not to work properly.

Final Evaluation

After the compiling phase, a final evaluation on the model is made using the train and test datasets. The *evaluate()* function is called, and the results are saved in two variables, called *train_metrics* and *test_metrics*. Subsequently, train and test losses, RMSE and MAPE are printed on the screen in order to be collected and studied.

```
train_metrics = model.evaluate(X_train, y_train, return_dict=True)
test_metrics = model.evaluate(X_test, y_test, return_dict=True)

print('\n\nTrain loss: {:.2f}, Test loss: {:.2f}'.format(train_metrics['loss'],
                                                        test_metrics['loss']))

print('Train RMSE: {:.2f}, Test RMSE: {:.2f}'.format(train_metrics['rmse'],
                                                    test_metrics['rmse']))

print('Train MAPE: {:.2f}, Test MAPE: {:.2f}\n\n'.format(train_metrics['mape'],
                                                        test_metrics['mape']))
```

The Python program is now executed various times in order to evaluate all the different datasets, taking into consideration only the Vanilla LSTM model because, as seen previously, all the different LSTM models give similar results when used on the same dataset. The results are reported on the following table:

dataset	Train Loss	Test Loss	Train RMSE	Test RMSE	Train MAPE	Test MAPE
cases.csv	1478.01	653019.00	38.44	808.10	69511944.00	5591667712.00
discharged.csv	2048.25	451161.59	45.26	671.69	141947936.00	26.03
allCOVbeds.csv	134.42	242.56	11.59	15.57	6.80	4.99
accumulatedCases.csv	60265.69	10610372.00	245.49	3257.36	1.08	4.14
accumulatedDis.csv	21346.02	7952625.50	146.10	2820.04	6112884.50	0.98
inCOVpatients.csv	34.80	43.07	5.90	6.56	9238986.00	24.38
inTOTpatients.csv	4119.00	4755.48	64.18	68.96	13.81	12.56
resPatients.csv	94.19	109.14	9.71	10.45	5.40	4.63
deaths.csv	1.99	6.94	1.41	2.63	354511392.00	137399568.00
COVnoRes.csv	0.65	0.21	0.81	0.46	137520144.00	177197776.00
COVwithRes.csv	8.22	5.38	2.87	2.32	2245033.00	13.28
allBeds.csv	23493.45	19949.93	153.28	141.24	3.71	3.14
noResPatients.csv	56.81	37.41	7.54	6.12	27.70	26.62
accumulatedDeaths.csv	3.76	43.77	1.94	6.62	6311135.00	0.45

The obtained evaluation of machine learning models is a crucial step in the development process, as it allows to assess the model's performance and identify any areas that need improvement. While it is desirable to have as low as possible values for loss, RMSE, and MAPE, it is essential to keep in mind that these metrics are highly dependent on the range of the data being analyzed. For instance, a RMSE score of 3000 may be considered small when the data ranges from 0 to 100000, while it can be regarded as rather large if it ranges from 0 to 1.

In addition, the resulting high testing loss or RMSE compared to the training ones could indicate that the model is overfitting, learning the noise in the training data instead of the underlying patterns. Moreover, the MAPE metric may present enormous values when there are zeros in the datasets, causing its formula not to work correctly. Therefore, it is essential to consider the characteristics of the data when selecting the appropriate evaluation metrics to ensure that they provide meaningful and accurate results.

Improvements

Besides the problems connected to the chosen evaluation metrics, a major concern resulting from the evaluation is overfitting. There are several methods to remove overfitting in a machine learning model, among which increasing the size of the data, using regularization techniques, early stopping, dropout or augmentation [6].

In order to improve the existing model, some of these techniques will be applied and a new loss function will be introduced.

Early Stopping

Early stopping is a technique used in machine learning to prevent overfitting and improve the generalization performance of a model. The basic idea behind it is to stop the training process of a model when its performance on a validation set stops improving, instead of continuing to train until the training loss is minimized.

By stopping the training process early, before the model starts to overfit the training data, the model's generalization performance is improved, and it is better able to generalize to new, unseen data [7].

Early stopping is a simple but effective way to prevent overfitting. It is important to note, however, that early stopping may also cause underfitting if the training is stopped too early, that is why a patience of 3 epochs has been added to it. After having imported the *EarlyStopping()* function from Keras' callbacks library, it is called in the code right before fitting the model:

```
es = EarlyStopping(monitor='val_loss', mode='min', verbose=1, patience=3)
```

Regularization

L1 and L2 regularization can be used together in machine learning to improve the generalization performance of a model. This technique is known as Elastic Net regularization, and it is powerful for preventing overfitting.

The regularization term is a linear combination of the L1 and L2 penalties. The L1 penalty encourages sparsity in the model's parameters, while the L2 penalty encourages small parameter values [8].
During the simulation analysis phase, L1 and L2 regularizations had already been applied to the LSTM models that were overfitting. To further improve the results, they are now added to the remaining ones:

```
model.add(LSTM(units=32, activation='relu', input_shape=(n_steps, n_features), kernel_regularizer=regularizers.L1L2(
                                                                                    l1=0.05, l2=0.005)))
```

MSLE

MSLE stands for Mean Squared Logarithmic Error, which is a loss function used to evaluate regression models. Like MSE (Mean Squared Error), MSLE measures the difference between the predicted and actual values of a model, but it takes the logarithm of the predicted and actual values before calculating the squared difference. The use of logarithms is useful when dealing with predictions that cover a large range of values, as it helps to reduce the impact of outliers and extreme values on the loss function. In most of the used datasets, in fact, this is the case. By taking the logarithm of the values, the loss function focuses more on the relative differences between predicted and actual values rather than the absolute differences [9].

```
model.compile(loss='msle', metrics=[RootMeanSquaredError(name='rmse'), 'mape'], optimizer=Adam(learning_rate=0.001))
```

After the application of the above-described methods to reduce overfitting and minimize the loss function, the model is evaluated one last time. The following table represents the old values of MSE train and test losses next to the new values, calculated with the newly introduced MSLE function:

dataset	MSE Train Loss	MSE Test Loss	MSLE Train Loss	MSLE Test Loss
cases.csv	1478.01	653019.00	0.33	0.56
discharged.csv	2048.25	451161.59	0.38	0.26
allCOVbeds.csv	134.42	242.56	0.06	0.05
accumulatedCases.csv	60265.69	10610372.00	0.09	0.08
accumulatedDis.csv	21346.02	7952625.50	0.06	0.05
inCOVpatients.csv	34.80	43.07	0.22	0.17
inTOTpatients.csv	4119.00	4755.48	0.11	0.11
resPatients.csv	94.19	109.14	0.06	0.07
deaths.csv	1.99	6.94	0.30	0.29
COVnoRes.csv	0.65	0.21	0.10	0.12
COVwithRes.csv	8.22	5.38	0.10	0.12
allBeds.csv	23493.45	19949.93	0.07	0.07
noResPatients.csv	56.81	37.41	0.26	0.22
accumulatedDeaths.csv	3.76	43.77	0.09	0.06

When the target variable represents a count, such as in traffic flow, sales data or, in this case, hospital patients, MSLE is very effective. Since the logarithm is used in the calculation, larger errors are still weighted more heavily, but the scale of the target variable is now taken into account without the need of normalizing the data, resulting in more cohesive loss values.

Conclusions

In conclusion, the evaluation of LSTM models using different metrics is a crucial step towards improving their performance in forecasting applications. In this study, the Mean Absolute Percentage Error and the Root Mean Squared Error metrics have been used for evaluating a LSTM model's performance on forecasting Canary Islands' COVID-19 cases.
While MAPE is commonly used in forecasting applications, it may not be suitable in this case, as many of the given datasets present many zeros, which can lead to inaccurate predictions and really high MAPE scores.
In this case RMSE is a more optimal metric to use as it reduces the loss through its squared root and not too low results are acceptable as they are proportional to the dataset range.
However, it had already been acknowledged that the datasets used in this study are simple and relatively small, resulting in generally low accuracy levels.
Overall, the findings emphasize the importance of carefully selecting appropriate evaluation metrics for LSTM models and considering the characteristics of the dataset being used. Further research can explore the use of other evaluation metrics and more complex datasets to improve the accuracy and robustness of the LSTM models in forecasting applications.

Bibliography

[1] TrainDataHub (2022). *Interpretation of evaluation metrics for regression analysis (MAE, MSE, RMSE, MAPE, R-squared, and..., Medium*. Accessed: March 10, 2023. https://medium.com/@ooemma83/interpretation-of-evaluation-metrics-for-regression-analysis-mae-mse-rmse-mape-r-squared-and-5693b61a9833

[2] *RMSE: Root mean square error* (2023) *Statistics How To*. Accessed: March 10, 2023. https://www.statisticshowto.com/probability-and-statistics/regression-analysis/rmse-root-mean-square-error/

[3] *Root mean square error (RMSE) or mean absolute error (mae)? - GMD* (no date). Accessed: March 12, 2023. https://gmd.copernicus.org/preprints/7/1525/2014/gmdd-7-1525-2014.pdf

[4] M, P. (2022) *End-to-end introduction to evaluating Regression Models*, *Analytics Vidhya*. Accessed: March 15, 2023. https://www.analyticsvidhya.com/blog/2021/10/evaluation-metric-for-regression-models/

[5] *Calculate mean absolute percentage error using tensorflow 2* (2020) *Lindevs*. Accessed: March 15, 2023. https://lindevs.com/calculate-mean-absolute-percentage-error-using-tensorflow-2/

[6] *What is overfitting in deep learning [+10 ways to avoid it]*. Accessed: March 16, 2023. https://www.v7labs.com/blog/overfitting

[7] Brownlee, J. (2020) *Use early stopping to halt the training of neural networks at the Right Time*, *MachineLearningMastery.com*. Accessed: March 20, 2023. https://machinelearningmastery.com/how-to-stop-training-deep-neural-networks-at-the-right-time-using-early-stopping/

[8] Oleszak, M. (2019) *Regularization in R tutorial: Ridge, Lasso & Elastic Net Regression*, *DataCamp*. DataCamp. Accessed: March 23, 2023. https://www.datacamp.com/tutorial/tutorial-ridge-lasso-elastic-net

[9] *Mean squared logarithmic error loss*, *InsideAIML*. Accessed: March 26, 2023. https://insideaiml.com/blog/MeanSquared-Logarithmic-Error-Loss-1035