Meyers Lab Report #1

Advay Vyas

September 4, 2025

Contents

1 Forecast evaluation background

I read [2] and took notes to learn how forecast evaluation works.

1.1 Introduction

ML metrics are very different than forecasting metrics because time series data is much messier and the regular ways of determining model success fail to measure accurately. Forecast origin is self-explanatory and forecast horizon is the section of time that we are predicting upon. Fixed origin evaluation uses the same training data each iteration and the forecasts are computed "one-step ahead". On the other hand, rolling origin evaluation incorporates the new data into the testing set first, and then into the training set on the next iteration. In my opinion, rolling origin seems a lot better and I think that is what we use – not sure yet though.

Time series data also often has a series of issues that make predicting and measuring predictions much harder. Time series can have non-stationarities like seasonality, trends, or breaks; non-normality like fat tails and outliers; and series with a very short history which are inherently hard to use as training data.

1.2 Benchmarks and exaggerating success

Next, the paper moves on to the topic of benchmarks. The naive forecast (a.k.a no-change model) uses the last known observation as the forecast and actually has good performance in some scenarios (likely those with little to no change period to period, not for us with the prevalance of vaccine skepticism). While not a viable model for real prediction, it should definitely be used a strong benchmark to check the performance of other models. Rather than an abstract metric, performing better than the most simple prediction should be the bare minimum for an accurate model. For example, finance models sometimes involve heavy computations like neural networks only to fail against the naive forecast (random walk w/out drift). Essentially, ML research fails to account to be better than simple random modeling – benchmarks are useful!

In more complex scenarios like "clear seasonal patterns", a seasonal naive model should be used - makes sense. Even though that seems obvious, researchers often compare to non-seasonal benchmarks and can show great

results. To add more to this discussion, papers often use "overkill" ways of forecasting for no reason - the paper here mentions a modeling a 2-dimensional linear relationship with a neural network (page 801).

1.3 Evaluation metrics, plots, and data leakage

Metrics like MSE, RMSE, MAE are often used with many different time series yet it is very important to watch out for each time series having a different scale. Some (like myself last semester) have used the R^2 value to determine how well a model predicts (especially in random walks, while mine was definitely not random). MAPE is also another metric - luckily for us, we are going to be using a weighted interval score (next section!).

Forecasting plots are also another area to watch out for because they can be quite confusing when comparing different models. For example, watching fit by looking at the "horizontal" shift and finding it to be small instead of the "vertical" shift can give the model the illusion of fitting well. This paper makes the suggestion to only use plots of the forecasts for sanity checks and not for real use. I disagree and I think that those horizontal shifts imply that the model can predict spikes and dips after a delay (as long as it's about always the same delay) and that plots serve a very useful purpose of pinpointing far-off predictions without requiring brute force checking.

Lastly, we tackle the topic of data leakage (usage of the test/unseen data during the training process). In forecasting, since it involves time, it is often to hard to keep track of data seperation during rolling origin prediction. While that is unavoidable, indirect forms of data leakage are much more common. For example, in forecasting, smoothing, decomposition, or normalization over the series before prediction can indirectly help the model predict where the missing data will fit into the distribution. With data leakage, models can often easily outperform any existing or new frameworks, causing issues in result accuracy. This problem also arises when using multiple time series at once, where one series could help the model predict the entire outcome. In conclusion, the most important way to avoid data leakage are during preprocessing, feature extraction, and making sure that one series doesn't reveal the future of another series.

1.4 Guidelines and best practices

The paper states that the forecast model construction plus evaluation usually contains these steps:

- Data forecasting
- Forecasting
- Error calculation
- Error measure calculation
- Statistical tests for significance

1.4.1 Data partitioning

The paper now examines guidelines of data partitioning. Fixed origin setup is one of the fastest evaluation setups since no new training data is added. However, with single series forecasting, the setup only provides one forecast. It is recommended to have multiple forecasts instead to see the forecast distribution. A drawback of fixed origin setup is that errors might arise from the patterns in that particular region.

On the other hand, rolling origin (a.k.a. time series cross-validation) evaluation setups update the forecast origin every step and the model encounters new actual data. The two options available now are to recalibrate with new data completely or update the model as input. Traditional models normally recalibrate while ML models just accept new data and periodically retrain the model due to it not being an option. Rolling origin can be conducted through either expanding the window or rolling the window setup. Expanding the window retains all the data from the very beginning, effectively just slowly growing the total training data. Alternatively, a rolling window setup makes sure that the data "length" stays the same - the time period length as the old data is deleted to make room for the new data. It is also possible to start with an expanding window and then transition to a rolling window as needed.

A common misconception is that temporal order is important for the time series during training. A form of data partitioning that utilizes this fact is k-fold cross-validation, which splits up the data into different "folds" randomly and repartitions training and test data. However, it does have a

couple drawbacks like causing problems with non-stationarities, difficulty in capturing serial correlation, and training data containing future data while test data contains old data. Even with these problems, pure AR models can benefit from this technique. Additionally, data-partitioning for non-stationary data is very hard (the paper is quite vague on this, around page 811).

In conclusion, data partitioning boils down to the length of the series (kind of), where short series choose k-fold CV while the more common longer series use tsCV (time-series cross-validation).

1.4.2 Error measures

Error measures are a key focus of this week's work so this section should be particularly important. Scale-dependent forecast bias can be assessed with Mean Error, while two other scale-dependent measures usually used in regression are MSE and MAE. However, we should use scale-independent (next to impossible) but we will examine a wide variety of measures. A couple examples are MSE and RMSE that optimize for the mean, MAE and MASE that optimize for the median, and some researchers apply a mix of these error models. Actually, the mix/blend of these models is the most optimal method as of now and is highly recommended. I think we will be doing weighted interval score but I'm curious to see how it is similar/different to mixed error methods in time-series data. On pages 814–815, various common error measures are listed. On pages 816–820, a comphrensive list of error measures is listed by category for reference.

1.4.3 Statistical significance tests

The Diebold-Mariano test and Wilcoxon rank-sum test are two major tests for comparing between two competing forecasts. Another two-forecast test is the Giacomini-White test that can also assess the conditional predictive ability. A table of statistical signifiance tests and associated information follows from pages 822–825. Moreover, the Friedman test looks at detecting signifiance between multiple comepting methods, looking at the ranks of the methods sorted by mean errors. When performing significance testing, it is important to look at the amount of data included. For example, a very high number of series has a low CD $\rightarrow$ significant results for even small differences in models. This isn't a flaw – results become more reliable and are

statistically highly significant. On the other hand, having poor performing models may make the CD larger and make other intermediate methods have no significant methods.

1.5 Conclusion

Model evaluation is very important, one of the hardest tasks in forecasting. Basically, we need to benchmark the simplest and smartest ways, use accurate plots, avoid data leakage, use tsCV, and then choose the right evaluation measure before evenly distributing statistical models when testing against others.

2 Weighted interval score metric

I read [1] and then took notes on it.

2.1 Background

Epidemic forecasts are increasingly probabilisitic and require a way to store forecasts more effectively. The approach suggested by the paper is to store forecasts in the form of predictive quantiles/intervals. With this format, distributions can be stored in detail yet still require a better scoring metric than those currently available – this paper will provide that!

Logarithmic score is very popular, defined as $\log S(F, y) = \log(p_y)$, where p_y is the probability assigned to the observed outcome y by the forecast F. Larger values are better and FluSight uses this metric. Basically penalizes or gives gains if it falls in line with how likely it will be. An alternative to the logartihmic score (considered better) is the continuous ranked probability score which is a cumulative distribution function which simplifies to the ranked probability score with integers.

2.2 Interval format scores

Both scores cannot be evaluated directly if forecasts are stored as intervals. A proper score that only requires prediction intervals is the interval score $\rightarrow$ basically good score is within the prediction interval and worse (higher score) outside the farther you get, self-explanatory but useful. Consists of

the width of the interval (sharpness of the forecast), penalty term for the lower end, and penalty of the upper term which increase in penalty as you get farther. Figure on page 3 makes a lot of sense for understanding this.

For more detail on the predictive distribution, common to report several central prediction intervals. To combine all this data into one score, a weighted interval score can be evaluated (formula redacted for time). WIS is a special case of the quantile score, proper for any set of nonnegative weights. A natrual choice can also approximate the continuous ranked probability score if needed. More simply, the score can be understand as a "measure" of distance between the predictive distribution and the true observation.

To compare different prediction methods at the same time, we have to add their scores over time and for various forecasts. We also should try to look at forecast quality over time across different horizons, and longer-term characteristics of an epidemic curve.

As a comparison to other metrics in integer-valued outcomes, the WIS is a good approximation to the CRPS and take into account uncertainty in the forecast distribution, more so than the other methods.

It is important to know that WIS demonstrates sensitivity to distance while logS is a local score that ignores distance. While logcal scoring rules can be good for inference, sensitivity to distance is useful in decision-making. At the end of the day, it is up to the public health officials to what they value more, (WIS) sensitivity to distance with robust scoring that minimizes "across the board" or local logarithmic scores that make the penalty for a few very misguided forecasts much worse.

The paper next goes into some applications of the WIS in FluSight, mentions why relative errors should be used with caution, and then concludes by explaining why CRPS and WIS are great options.

3 Code investigation

I plan to read through the Flusion and Local-Level-Forecasting codebases and write about what I noticed and questions I have.

3.1 Flusion

Data-pipeline seems self-explanatory. The **eda.ipynb** file in the code/eda folder was very interesting, showcased all kinds of special datasets that were

investigated as a part of the process, a very enjoyable read. A nice way to organize research thoughts and brainstorming I think, and creative graph design as well. GBQ, sarix seemed like just model code – I'll look into it next week if I'm assigned it more.

Lastly, the **flusion.R** file in the code/flusion folder was nice to read. I think the animal names for the ensemble vs GB only was kind of funny. I also got to find out the weightage of the models that I was curious about during the paper $\rightarrow$ 25% to GB methods and 50% to sarix methods. While they don't add up to 2, I just assumed they wanted a clearer way of doubling the weightage of sarix without messing around with $\frac{1}{3}$ messing up calculations (this is an unnecessary tangent). Anyway, this code file showed me the entire "pipeline"/process of the ensemble training, albeit calling other functions at certain times instead of the entire codebase in one file but instructive regardless.

3.2 Local-Level-Forecasting

3.2.1 forecast_model.py

Code looks like implements LightGBM, more technical than the papers I was just reading, what stood out to me was the quantile forecast generation. I'm a bit confused by the concept of training at a specific percentile and was curious as to what percentiles/quantiles that the code uses right now.

3.2.2 GBM_US_NSSP_public_state_pct.ipynb

Looks like preprocessing and a train/test split, I've seen these before so this makes sense. I might need to take a further look on some of these packages and libraries about time series data, however, since I'm much more unfamiliar with that.

3.2.3 GBQR_forecasting_plot.R

I think that this just has a bunch of CSV reading in and out, looks like merging state data?

3.2.4 forecasting_summary_plot_ftn.R

This code file looks very interesting, computing the WIS score, estimating the median (that's the quantile we use). **What is ribbon data?** I'm a bit confused by that, is it a "band" of predictions or confidence intervals of the predictions? Also, I think this is the code file I might read more into the next week.

3.2.5 preprocess_and_plot.R

This file was just pre-processing, seems useful to write about here in case I'm curious or need to know how the data was filtered prior to the model training process. Still confused about ribbons, however.

4 Miscellaneous

I'll just store ideas I thought were interesting in this section and investigate them.

4.1 Taylor polynomials for forecasting

Just found about using the SIR model and approximating it with Taylor polynomials maybe, not much surpisingly compared to last week when I saw a bit more. I might look into this for my own thesis purposes to be honest, seems interesting.

References

[1] Johannes Bracher et al. "Evaluating epidemic forecasts in an interval format". In: *PLOS Computational Biology* 17.2 (Feb. 2021), pp. 1–15. DOI: 10.1371/journal.pcbi.1008618.

[2] Hansika Hewamalage, Klaus Ackermann, and Christoph Bergmeir. "Forecast evaluation for data scientists: common pitfalls and best practices". In: *Data Mining and Knowledge Discovery* 37.2 (2023), pp. 788–832.