Final Report for Time Series Forecasting

Aaron Mullen

University of Kentucky

CS 395: Independent Study

Dr. Cody Bumgardner, Supervisor

December 6, 2022

Final Report for Time Series Forecasting

Introduction

Time series forecasting is the process of looking at time-oriented data and making predictions about future data points[1]. It is an important process used in a variety of fields whenever future predictions are necessary, and there is a wide range of models that can be used to perform this analysis. Some models are built for time series forecasting, while others are applications of more general machine learning techniques. Different techniques have different strengths, such as working with seasonal trends. Additionally, there are different types of time series datasets. Some have covariates, which are other variables besides the timestamp that can aid in the prediction of certain data values. Others are multivariate sets, where multiple variables are predicted at once. In this project, I worked with these types of models and datasets to compare their performances and learn more about the area of time series forecasting.

Existing Work

My intention with this project was to research existing work in the area and try to implement it myself. A paper written by De Gooijer and Hyndman describes work in the time series forecasting field over the last 25 years[2]. The paper discusses the history and development of different models and accuracy metrics, many of which are used in my work. Historically, ARIMA and Exponential Smoothing were two of the most important forecasting models, but as the machine learning field has developed, more complex deep learning models have gained popularity.

An example of specific work comes from a paper written by Chimmula and Zhang, which analyzed time series data for Covid-19 transmission[3]. In their work, they used a Long Short Term Memory (LSTM) recurrent neural network to make predictions. This type of deep-

learning model seems to be used frequently in modern time series forecasting[4,5]. Therefore, I implemented an LSTM recurrent neural network model for every dataset I tested to ensure accurate comparisons between that model and others.

Methods

I worked with these models using Python, specifically using the Darts library[6], which provided the models and accuracy metrics used, as well as some of the datasets. Other Python packages include different models as well, such as SKTime[7] and Pyflux[8], in addition to libraries that are dedicated to a single model, such as Prophet[9] and TSFresh[10]. In testing the models, I wrote functions that would first initialize the model, which frequently required parameter tuning and testing to ensure the best possible performance given the size and complexity of the dataset used. Some important parameters to tune included input and output chunk lengths, hidden layer size, number of layers, batch size, and forecast horizon length. Then, depending on the model, the training and validation sets would be normalized before the model was trained on the data. After the training, the model would make predictions on the validation set. Finally, it would graph the entire dataset, with the predictions laid on top, and with an accuracy metric as a title.

I successfully tested thirteen models in total, which were outlined in my initial project report. As a summary, the models I used are listed below, with explanations.

1. ARIMA- Auto Regressive Integrated Moving Average[11]
    - This model is built specifically for time series forecasting, using the dataset's previous values, as well as the models' previous errors, to make predictions about future data points.
2. VARIMA[12]

- This model is similar to ARIMA, but it is generalized to multivariate datasets instead of just univariate ones.

3. Exponential Smoothing[13]
    - This model also works similarly to ARIMA, except in how the dataset's prior values are used. In ARIMA, all of the lags are weighted equally, but in exponential smoothing, their relevance to the prediction exponentially decreases as they get older, so more recent data points are weighted as more important to the model.
4. Fast Fourier Transform[14]
    - This model uses a discrete Fourier transform to analyze the time series data. It takes in the time step data as input and generates the frequencies present in the dependent variable, which encodes information about the time series' trend and seasonality.
5. Four Theta[15]
    - This method creates two Theta lines, one that represents the series' linear trend and one that represents its curvature. Depending on the value of the theta coefficient, the line can approximate long-term or short-term trends. After the predictions are generated from each Theta line, these forecasts are simply averaged together to generate the final prediction.
6. Kalman Forecaster[16]
    - This model uses measurements, noise, and inaccuracies to produce estimates of the unknown variable. It is a recursive algorithm that updates weights given the uncertainties and errors of current estimates.
7. TBATS- Trigonometric, Box-Cox, ARIMA, Trend, Seasonal[17]

- This method produces many models, using the different components that make up its name. For example, it will consider models with and without Box-Cox transformations, with and without a seasonal model, with and without trend damping, etc. After this, it will choose which model performed the best using the Akaike information criterion, which is an estimator of prediction error.

8. Regression / Linear Regression[18]
    - This method builds a feature matrix using current and past observations and estimates a linear relationship between the matrix and the prediction variable. It can use both time-step features for more time-dependent data and lagged features for more serial-dependent data (more dependent on past observations than time).
9. Random Forest[19]
    - This model builds an ensemble of decision trees using slightly different datasets for each one. Predictions from each tree are averaged together to produce the final estimate.
10. Recurrent Neural Network[20]
    - This works as an ordinary RNN, using memory blocks and backpropagation to produce predictions. Specifically, a Long Short Term Memory network was used, which uses a set of gated cells to keep track of previous inputs and manage information better, using input, output, and forget gates to categorize the importance of different data points and trends to the predictions.
11. NBEATS- Neural Basis Expansion Analysis for interpretable Time Series forecasting[21]
    - This model requires a different definition of the training and validation sets. It begins with a small training set, tries to predict the next block of time, observes the actual

values, and remembers its error. Then it incorporates that block into the training set and runs again, predicting the next block. This continues until it has mapped the entire series, at which point it is equipped well to make future predictions.

12. NHiTS- Neural Hierarchical Interpolation for Time Series forecasting[22]
    - This model is very similar to NBEATS, but it uses multi-rate sampling of the inputs.
13. Temporal Fusion Transformer[23]
    - This is a Torch deep neural network architecture that works very well with covariates.

I worked with these models over several datasets, depending on when the models were applicable. Additionally, I summarize below the different evaluation and accuracy metrics I used, which were also obtained from the Darts library[6].

1. Mean Absolute Percentage Error[24]
    - This metric sums together, for each data point, the ratio of the prediction and actual values, with the formula (actual – prediction) / actual.
2. Mean Absolute Scaled Error[25]
    - This metric takes the mean absolute error of the forecast values and divides that by the mean absolute error of a naïve forecast that always predicts at time t whatever the value of the variable was at time t-1.
3. Root Mean Square Error[26]
    - This is the square root of the mean square error, which is just the average of the squared errors of each prediction data point.
4. Coefficient of Correlation[27]

- This is a general measurement of the relationship between two variables. In this case, it is used to measure the linear correlation between the prediction and the validation set.

5. R2 Score[28]
    - This statistic measures the proportion of variance in the dependent variable that is explainable by the model and the independent variables.

At first, the Mean Absolute Percentage Error was used to evaluate the models, but there were problems with that metric that are discussed later. The Root Mean Square Error was used instead to compare the performance of the models.

I consider the models in two separate groups. The first, which I will refer to as Group 1, is the class of simpler models that were developed specifically for time series forecasting, which includes ARIMA, Exponential Smoothing, Four Theta, Kalman Forecaster, and TBATS. The second group, referred to from here on as Group 2, is made up of the more complex models that were created for general machine learning tasks, which include regression, random forest, recurrent neural network, and Fast Fourier Transform. Additionally, models such as NBEATS, NHiTS, and the Temporal Fusion Transformer are very complex but were developed specifically for time series forecasting. I chose to consider these as part of the second group, as their architecture more similarly aligns with those models rather than the first group's.

Additionally, the models can be distinguished by the types of datasets they support. Some support the use of covariates, while some support multivariate datasets, and some support both or neither. Below is a table showing the capabilities of each model.

Model	Multivariate?	Covariates?
ARIMA		Yes

Exponential Smoothing		
Four Theta		
Fast Fourier Transform		
Kalman Forecaster	Yes	Yes
TBATS		
Regression	Yes	Yes
Random Forest	Yes	Yes
Recurrent Neural Network	Yes	Yes
NBEATS	Yes	
NHiTS	Yes	
Temporal Fusion Transformer	Yes	Yes
VARIMA	Yes	Yes

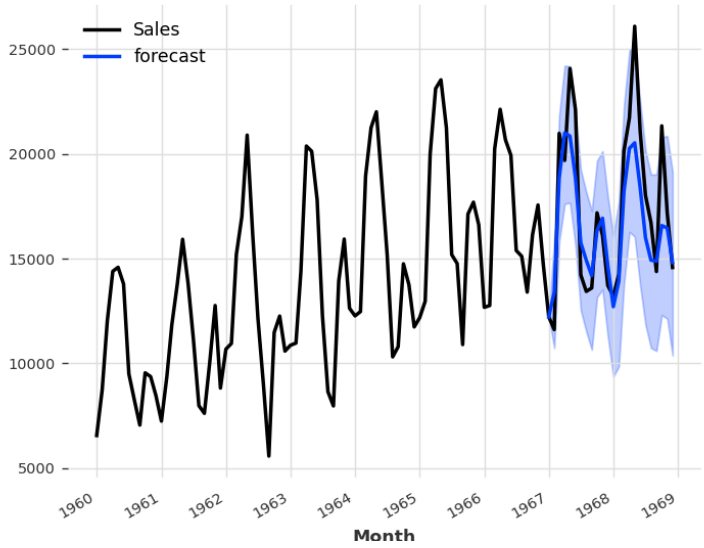
Figure 1

Results- Dataset 1- Simple, Univariate

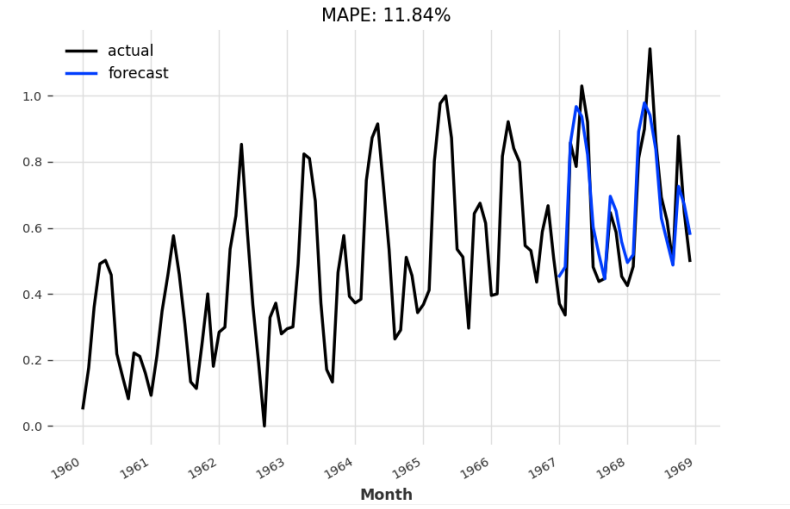
The first dataset I used was a simple car sales dataset. It only had two columns, with the month and year in one and the number of car sales in the other. This was monthly data that spanned over ten years, so it only consisted of 120 rows. This was a good one to start with because it allowed me to get a sense of how the models work and how to implement them without worrying about their performance on large, complex datasets. I learned much through this process that influenced how I continued my work with other datasets.

One conclusion I drew from my results on this dataset was that the models contained in Group 1 tended to work as well, if not better, than those in Group 2, and typically at a lower

training time. This was a simple dataset, and as a result, the simple models could avoid overfitting and create successful predictions. Below, Figure 1 shows the results from ARIMA, a model built for time series forecasting, and the results from a recurrent neural network.



ARIMA method: MAPE of 8.9%, RMSE of 0.1029



Recurrent Neural Network: MAPE of 11.8%, RMSE of 0.1293

Figure 2

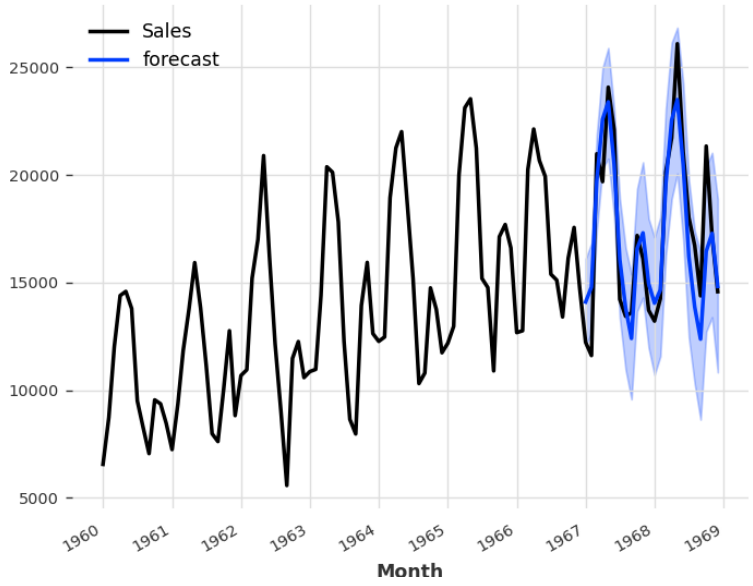
These results are similar, but ARIMA produced its predictions in one-third of the time that it took the Recurrent Neural Network. Below is a table listing the results of every model, as well as the training time for each. The Root Mean Square Error is an interpretation of the model's error, and thus, a lower RMSE indicates a more accurate performance[26].

Model	Root Mean Square Error	Time to Train (seconds)
ARIMA	0.1029	5
Exponential Smoothing	0.1237	5
Four Theta	0.2880	4
Fast Fourier Transform	0.2957	4
Kalman Forecaster	0.1932	4
TBATS	0.0908	75
Regression	0.1199	4
Random Forest	0.1596	4
Recurrent Neural Network	0.1293	15
NBEATS	0.0984	9
NHiTS	0.1208	9
Temporal Fusion Transformer	0.0968	113

Figure 3

The best results were from the TBATS model, with an RMSE of 0.0908, but as a drawback, it took the second longest to train of any of the methods, at 75 seconds. Because this method builds several models to determine which is the best, the process is lengthy and could be more impractical on large datasets. The only model that took longer was the Temporal Fusion Transformer, at 113 seconds, and this model achieved similarly good results to TBATS, with an

RMSE of 0.0968. Finally, the NBEATS model performed the next best, with an RMSE of 0.0984 at only 9 seconds of training. Below is the resulting graph of the TBATS model.



TBATS- MAPE of 8.8%, RMSE of 0.0908

Figure 4

Results- Dataset 2- Simple, multivariate

After trying the models on a simple, univariate time series, I next tried them on a simple, multivariate time series. In this case, the model must predict more than one variable at a time. Not every model supports multivariate datasets, so my options were more limited (see Figure 1). The dataset I used was a Darts-provided dataset that mapped both ice cream sales and heater sales, monthly over sixteen years. This was still a very small dataset, but it allowed me to test the multivariate capabilities of the models. The Group 1 models that supported multivariate datasets, like VARIMA and the Kalman Forecaster, produced poor results. Shown below is the graph of the VARIMA predictions.

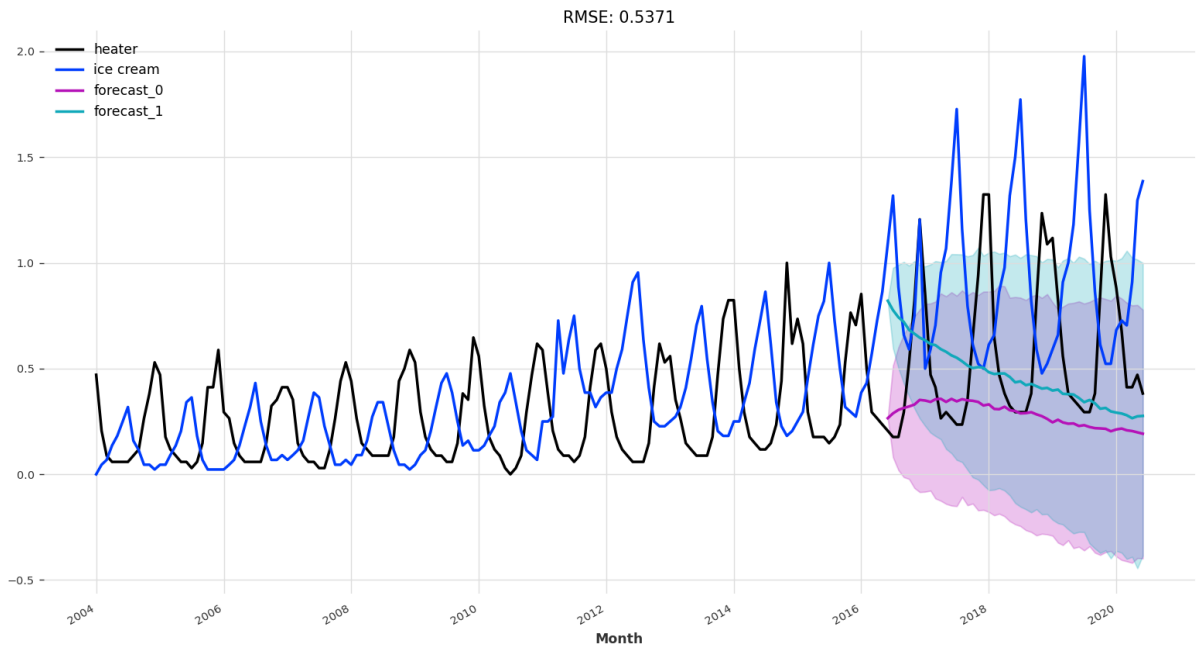
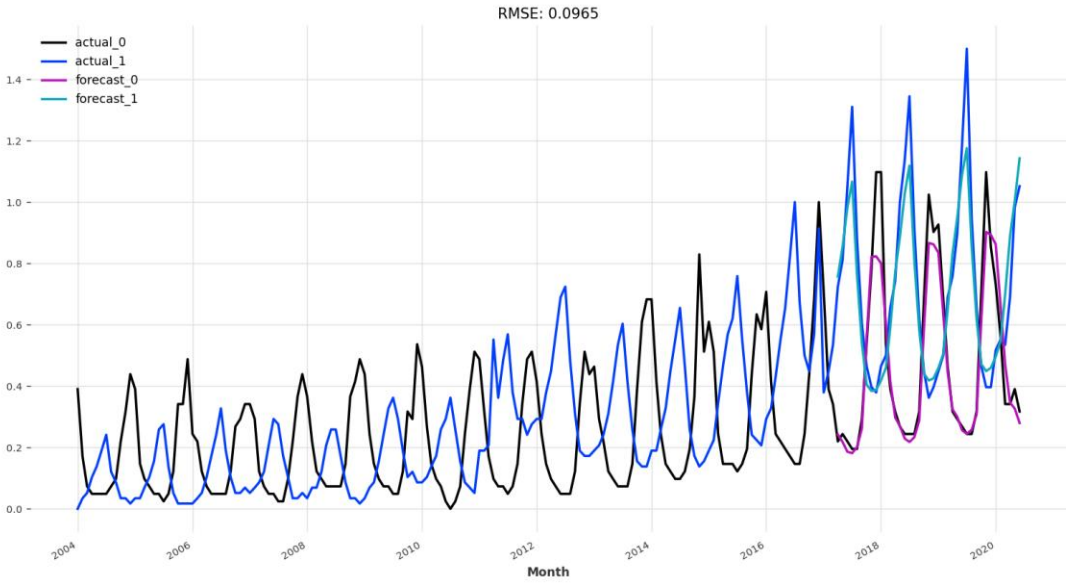


Figure 5

Much better results were obtained from Group 2 models, including the recurrent neural network, regression, temporal fusion transformer (TFT), and NBEATS models. The best results achieved from these models were from the recurrent neural network, as shown below in Figure 4, which contains the graph for the RNN and the table of all results.



Model	Root Mean Square Error	Time to Train (seconds)
VARIMA	0.5398	5
Kalman Forecaster	0.3968	4
Regression	0.2118	4
Random Forest	0.4519	5
Recurrent Neural Network	0.0965	26
NBEATS	0.1732	11
NHiTS	0.1806	11
Temporal Fusion Transformer	0.3707	135

Figure 6

Results- Dataset 3- Multi Covariate

The third dataset I tested kept track of energy usage in a household. This dataset was a huge increase in size, keeping track of energy usage per hour for several years, for a total of almost 50,000 rows. Aside from the time and energy usage, this dataset also recorded the temperature and precipitation at each hour. These variables acted as covariates, which are variables that are known throughout the time series and assist in making predictions. In this case, the temperature and precipitation would naturally have an impact on energy usage, making them important variables for the model to consider. Again, the usage of covariates limited the choice of models (see Figure 1). When working with this much larger dataset, I began to draw new conclusions about the effectiveness of the models. Below are the results of the ARIMA and TFT models.

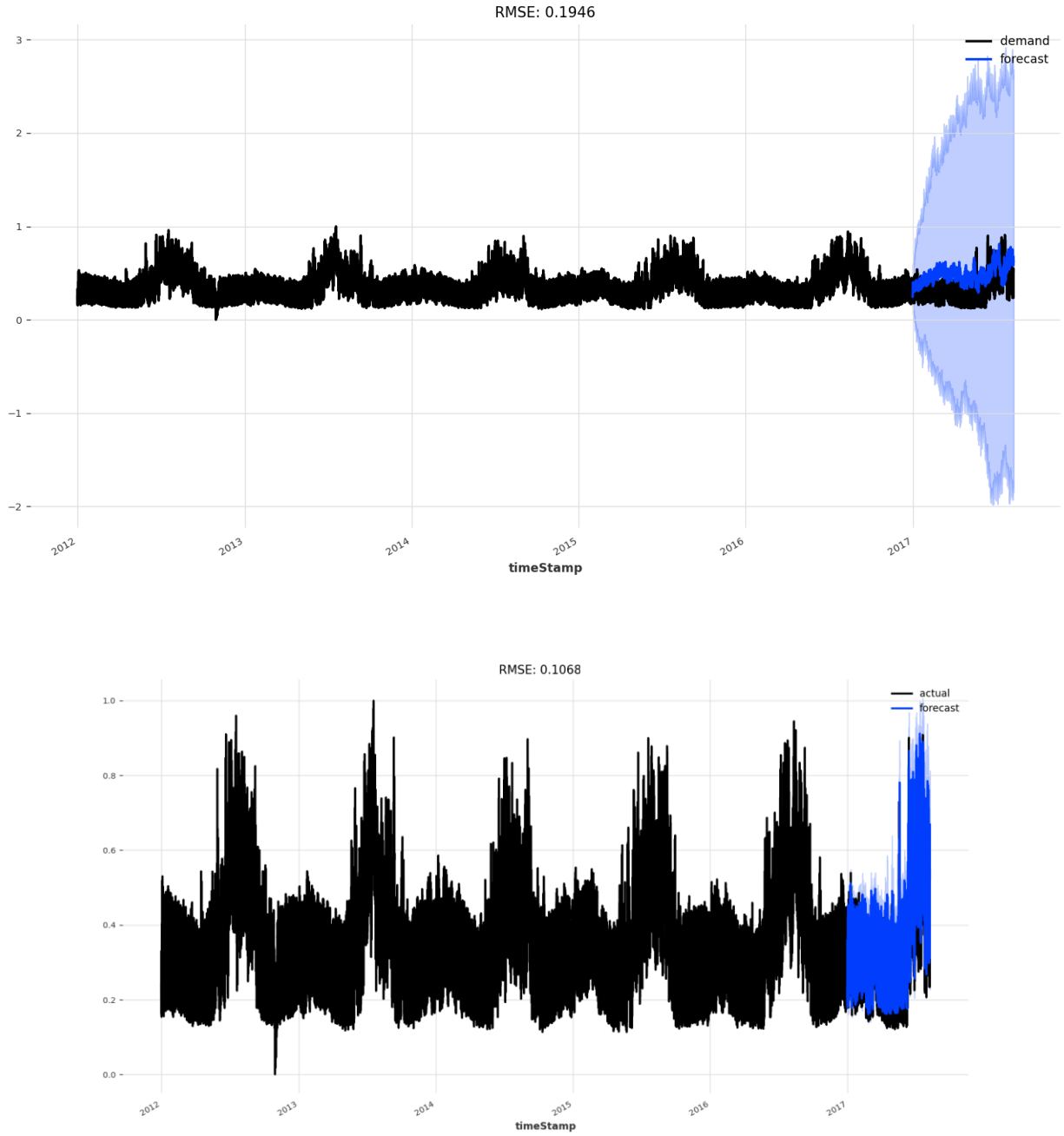


Figure 7

I believe that these more complex, Group 2 models can generalize to bigger and more complicated datasets better than the smaller, Group 1 models like ARIMA, which may not be complex enough to model these types of datasets. Interestingly, the regression model was able to

achieve a very low RMSE score by simply modeling the trend of the data rather than trying to account for all data points. The graph for the regression model is shown below.

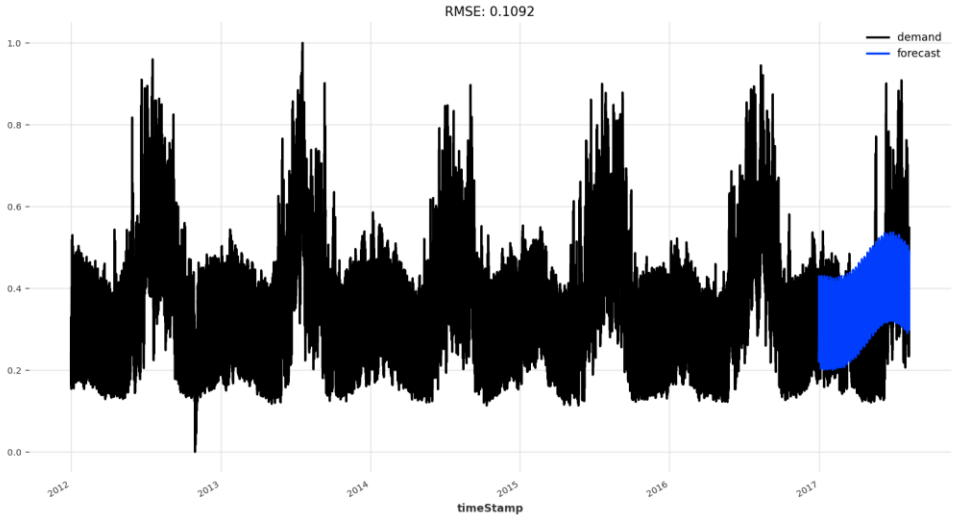


Figure 8

Below is a table with all results for this dataset. Note that the time to train has been changed from seconds to minutes.

Model	Root Mean Square Error	Time to Train (minutes)
ARIMA	0.1946	2
Regression	0.1092	0.5
Recurrent Neural Network	0.1693	8.95
Temporal Fusion Transformer	0.1068	65.37

Figure 9

Accuracy Metrics

One challenge I faced in this project was finding a good, consistent accuracy metric. I utilized many different accuracy metrics, including mean absolute percentage error[24], mean absolute scaled error[25], r2 score[28], coefficient of correlation[27], and root mean square error[26],

which were detailed earlier in this paper. The first one I consistently worked with was mean absolute percentage error, but this metric has a noticeable bias. It will consistently favor predictions that are too low over predictions that are too high, which skewed the results of the initial dataset. This observation was backed up by existing research[25]. Over the next datasets, I tried each other metric, and I found the most consistent success with root mean square error. The other metrics were either more difficult to implement or interpret, or they produced less consistent results. The only downside to the RMSE is that it is dependent on the scale of the data, so it required normalization of the series before training.

Conclusions

One large, initial challenge I faced was acquiring good datasets. Publicly available datasets of good quality are hard to find online, especially when specifically searching for datasets of a particular size or complexity. Some time series that I used proved too unpredictable to be analyzed by any of the models, while others were so massive that my computer ran out of memory when trying to train on them. Most of the datasets that I ended up successfully training on were included as part of the Darts library. Darts has extensive documentation[6], which allowed me to search through the provided datasets to find ones that matched the type I needed.

My initial hypothesis after completing training on the first, simple dataset was that the Group 1 models would generally perform better than the Group 2 models. ARIMA and TBATS produced better results than the RNN or Temporal Fusion Transformer at a much quicker training time (see Figure 3). The more complex models may have simply not had enough data to be able to produce as good of predictions. Some models performed very poorly even on the simple dataset, such as the Fast Fourier Transform or the random forest. I believe the random sampling of data points in the random forest method renders the model more inefficient for time

series datasets, where the order of the observations is very important. Another factor separating the Group 1 and Group 2 models that could affect results is the structure of the data itself. Time series models like ARIMA and Exponential Smoothing use the timestamps as well as lagged values of the predicting variable to analyze trends, while models like the RNN only use the lagged values. This difference could be important in more time-dependent series versus datasets with serial dependence. If the actual time that the data is recorded is more important than the past values of that data, then Group 1 models will likely be more effective.

However, I also discovered through my work that at times, the more complex machine learning models of Group 2 are much more effective at predictions. This was mainly shown through the energy dataset, where the ARIMA model performed very poorly compared to the RNN and the TFT. While these models require a lot of parameter tuning, their complexity allows them to develop very accurate predictions when there is a lot of data. Some simpler models, such as ARIMA, are unable to account for seasonal trends, which causes the predictions to be less accurate. Therefore, it seems that there is a large benefit to using neural network architectures for bigger time series.

In the end, if I was given a random dataset and told to choose a model to generate predictions for that time series, my choice would be the recurrent neural network. This model consistently performed well for every dataset and allowed for a wide variety of dataset structures and customizability. This conclusion was supported by the number of modern forecasting projects I found that used an LSTM recurrent neural network[3,4,5]. The temporal fusion transformer is another good choice of model, achieving similar results to the recurrent neural network in most cases, although it takes the longest to train of any model. NBEATS also proved very successful for every dataset. Depending on the series length and forecast window, it may

take a long time to build its training window and repeatedly make predictions, but the use of an expanding window allows it to make very accurate predictions by the time it has looked at the entire series. Finally, while the TBATS model performed the best on the first dataset, this model does not allow for covariates or multivariate datasets, limiting its potential uses. These results surprised me, especially after running the tests on the first dataset, as my hypothesis that Group 1 models performed better was proven untrue with testing on more complex datasets. Instead, one of the most ubiquitous and commonly used models in machine learning, the RNN, proved to be the most effective in my testing, justifying its popularity.

Data science is the field I am hoping to go into, so I appreciated the opportunity to gain practical experience with time series forecasting. I have learned a great deal about the models, processes, metrics, and challenges, which helps me to feel equipped to handle a real-world time series analysis task.

Acknowledgments

I acknowledge the supervision and assistance of Dr. V.K. Bumgardner in completing this work.

References

1. Time Series Forecasting methods. InfluxData. (2022, May 4). Retrieved December 5, 2022, from https://www.influxdata.com/time-series-forecasting-methods/
2. De Gooijer, J. G., & Hyndman, R. J. (2006). 25 years of Time Series forecasting. International Journal of Forecasting, 22(3), 443–473. https://doi.org/10.1016/j.ijforecast.2006.01.001

3. Chimmula, V. K., & Zhang, L. (2020). Time series forecasting of covid-19 transmission in Canada using LSTM Networks. Chaos, Solitons & Fractals, 135, 109864. https://doi.org/10.1016/j.chaos.2020.109864
4. Sharma, A., & Jain, S. K. (2021). Deep learning approaches to time series forecasting. Recent Advances in Time Series Forecasting, 91–97. https://doi.org/10.1201/9781003102281-6
5. Guan, Y. J. (2022). Financial Time Series Forecasting Model based on CEEMDAN-LSTM. 2022 4th International Conference on Advances in Computer Technology, Information Science and Communications (CTISC). https://doi.org/10.1109/ctisc54888.2022.9849780
6. Time series made easy in python. Time Series Made Easy in Python - darts documentation. (n.d.). Retrieved December 5, 2022, from https://unit8co.github.io/darts/
7. Welcome to sktime#. Welcome to sktime - sktime documentation. (n.d.). Retrieved December 5, 2022, from https://www.sktime.org/en/stable/
8. Introduction. Introduction - PyFlux 0.4.7 documentation. (n.d.). Retrieved December 5, 2022, from https://pyflux.readthedocs.io/en/latest/
9. Quick start. Prophet. (2022, September 21). Retrieved December 5, 2022, from https://facebook.github.io/prophet/docs/quick_start.html
10. Tsfresh. tsfresh. (n.d.). Retrieved December 5, 2022, from https://tsfresh.readthedocs.io/en/latest/
11. Janacek, G. (2009). Time Series Analysis Forecasting and Control. Journal of Time Series Analysis. https://doi.org/10.1111/j.1467-9892.2009.00643.x

12. Rosenblatt, M., & Quenouille, M. N. (1959). The analysis of multiple time series. Econometrica, 27(3), 509. https://doi.org/10.2307/1909486
13. Winters, P. R. (1960). Forecasting sales by exponentially weighted moving averages. Management Science, 6(3), 324–342. https://doi.org/10.1287/mnsc.6.3.324
14. Cochran, W. T., Cooley, J. W., Favin, D. L., Helms, H. D., Kaenel, R. A., Lang, W. W., Maling, G. C., Nelson, D. E., Rader, C. M., & Welch, P. D. (1967). What is the fast fourier transform? Proceedings of the IEEE, 55(10), 1664–1674. https://doi.org/10.1109/proc.1967.5957
15. Assimakopoulos, V., & Nikolopoulos, K. (2000). The Theta model: A decomposition approach to forecasting. International Journal of Forecasting, 16(4), 521–530. https://doi.org/10.1016/s0169-2070(00)00066-2
16. Morrison, G. W., & Pike, D. H. (1977). Kalman filtering applied to statistical forecasting. Management Science, 23(7), 768–774. https://doi.org/10.1287/mnsc.23.7.768
17. Gos, M., Krzyszczak, J., Baranowski, P., Murat, M., & Malinowska, I. (2020). Combined TBATS and SVM model of minimum and maximum air temperatures applied to wheat yield prediction at different locations in Europe. Agricultural and Forest Meteorology, 281, 107827. https://doi.org/10.1016/j.agrformet.2019.107827
18. HURVICH, CLIFFORD M., & TSAI, CHIH-LING (1989). Regression and time series model selection in small samples. Biometrika, 76(2), 297–307. https://doi.org/10.1093/biomet/76.2.297
19. Qiu, X., Zhang, L., Nagaratnam Suganthan, P., & Amaratunga, G. A. J. (2017). Oblique random forest ensemble via least square estimation for time series forecasting. Information Sciences, 420, 249–262. https://doi.org/10.1016/j.ins.2017.08.060

20. Connor, J. T., Martin, R. D., & Atlas, L. E. (1994). Recurrent neural networks and robust time series prediction. IEEE Transactions on Neural Networks, 5(2), 240–254. https://doi.org/10.1109/72.279188

21. Oreshkin, B. N., Carpov, D., Chapados, N., & Bengio, Y. (2019). N-BEATS: Neural basis expansion analysis for interpretable time series forecasting. https://doi.org/ https://doi.org/10.48550/arXiv.1905.10437

22. Challu, C., Olivares, K. G., Oreshkin, B. N., Garza, F., Mergenthaler-Canseco, M., & Dubrawski, A. (2022). N-HiTS: Neural Hierarchical Interpolation for Time Series Forecasting. https://doi.org/10.48550/arXiv.2201.12886

23. Lim, B., Arık, S. Ö., Loeff, N., & Pfister, T. (2021). Temporal Fusion Transformers for interpretable multi-horizon time series forecasting. International Journal of Forecasting, 37(4), 1748–1764. https://doi.org/10.1016/j.ijforecast.2021.03.012

24. de Myttenaere, A., Golden, B., Le Grand, B., & Rossi, F. (2016). Mean absolute percentage error for regression models. Neurocomputing, 192, 38–48. https://doi.org/10.1016/j.neucom.2015.12.114

25. Hyndman, R. J., & Koehler, A. B. (2006). Another look at measures of forecast accuracy. International Journal of Forecasting, 22(4), 679–688. https://doi.org/10.1016/j.ijforecast.2006.03.001

26. Hodson, T. O. (2022). Root-mean-square error (RMSE) or mean absolute error (mae): When to use them or not. Geoscientific Model Development, 15(14), 5481–5487. https://doi.org/10.5194/gmd-15-5481-2022

27. DERRICK, TIMOTHY R., BATES, BARRY T., & DUFEK, JANET S. (1994). Evaluation of time-series data sets using the Pearson product-moment correlation

coefficient. Medicine & Science in Sports & Exercise, 26(7). https://doi.org/10.1249/00005768-199407000-00018

28. Pierce, D. A. (1977). R 2 Measures for Time Series., Journal of the American Statistical Association, 74:368, 901-910. https://doi.org/https://doi.org/10.1080/01621459.1979.10481052