DSC 680
Project Two: Creating a Marketing Forecast with Neural Prophet
Milestone 2: White Paper
Milan Sherman

Topic: I am going to build a model to predict funnel metrics for the next 30 days for each of our marketing channels, including:

- Site visits
- Started applications
- Submitted applications
- Approved applications
- Bound (purchased) policies

Business Problem: to anticipate and plan for revenue, expenses, and cost per acquisition (CPA), we need to be able to generate the metrics noted above by channel. As our pay structure varies by channel, e.g., cost per click, started app, bound policy, etc., we need to be able to forecast these metrics by channel to get an accurate CPA. For revenue, it is sufficient to get the number of policies across all channels and use the average revenue per policy. Right now, the growth team is generating this forecast in a very manual manner, using historical data and anticipated changes in marketing spend for the following month. They would like to develop a more sophisticated model that can be automated.

For revenue it's generally sufficient for the aggregated numbers across all marketing channels for the month to be accurate, so some error/variation for individual channels is OK if the aggregated numbers are accurate. But the pay structure varies from channel to channel, as noted above, so we still need to try to minimize the error per channel to accurately predict our costs.

Research Questions:

1. Can we create a model that is as good or better at predicting funnel metrics by marketing channel?
2. Which parameters are most likely to improve the model accuracy?
3. What data will allow us to make the most accurate predictions (see below)?

Data

As our Marketing Growth team has been manually creating the forecasts that I was using a machine learning model to replicate, the data for this project was much more straightforward than it was for my first project. The data consists of the number of users who visited the site, opened and/or submitted an application, were approved for a policy, and purchased, i.e., bound, a policy. We have data for 18 marketing channels, but currently three are not active and thus were excluded from the model.

Data Dictionary

- Year: Year of the date
- Month: Month of the date
- Dt: Date at the day level
- Users: Number of users

- Program: marketing program
- Channel: marketing channel
- Status: status of the visit/application (visit, opened app, submitted app, approved app, bound app)
- Carrier name: insurance carrier underwriting the policy
- Device type: the type of device of the user (Windows desktop, android, iphone, etc.)

Although this was the data from the query that the marketing team uses for forecasting, the Neural Prophet package takes only two inputs: 'ds' which is a datetime object, and 'y' which is a numeric target. Thus, the data that I started with needed to be aggregated to the daily level by marketing channel and status (visit, open, submit, approved, bound). The question regarding what to use as the target was still open at this point and will be discussed in the next section.

Methods

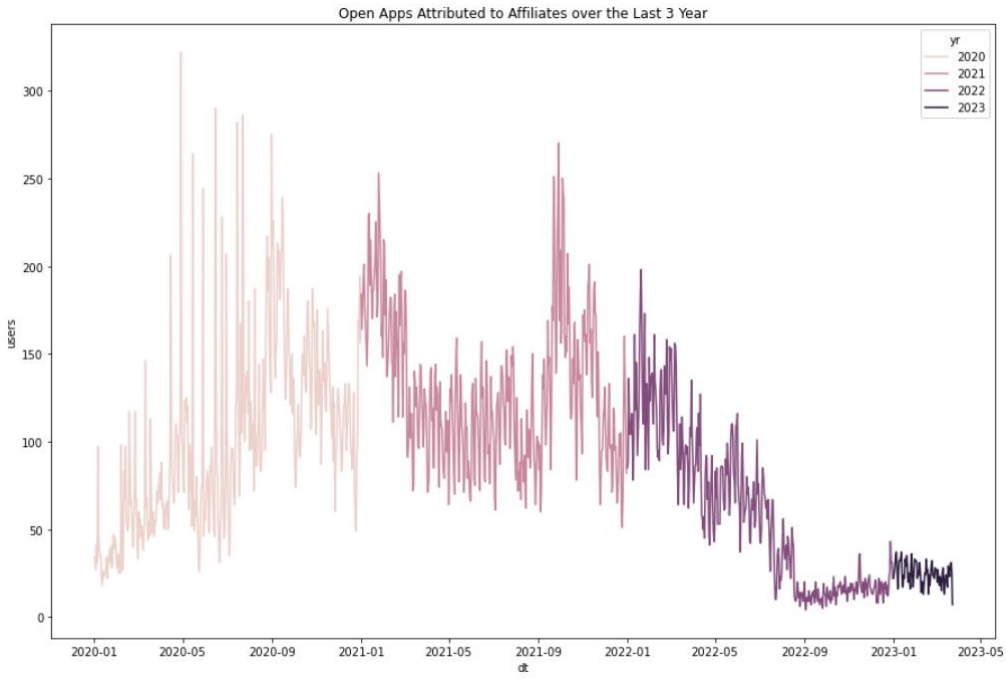
In terms of methods, the first question to answer is how accurate we could make the model and what parameters we could control to achieve better accuracy. Three factors were identified at the outset:

1. Although data was available on a daily basis, we are creating a forecast for the next 30 days, so one question was whether to make predictions on a daily basis and aggregate, or just to generate a single prediction for the next 30 days.
2. Another question was what data to train the model on. As noted above, due to changes in marketing spend and other factors, there has been a significant shift in our site traffic.
3. A third issue was what part of the funnel to use as the target, i.e., visits, opened applications, submitted applications, etc., as we can use recent conversion metrics for each channel to extrapolate the rest of the funnel. For example, if we predict the number of visits by channel, we can use what we know about how many of those visits results in an opened application to predict open applications from the predicted number of visits. Conversely, if we predicted the number of opened applications by channel, we could use the same conversion rate to predict the number of visits.

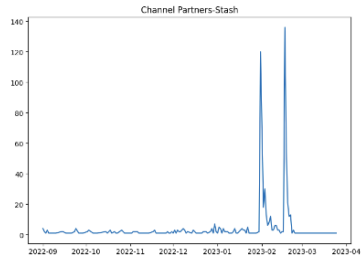
For the first question, I decided to start with making daily predictions, for a couple of reasons. First, make predictions at the same granularity as the data made the most sense. More importantly, however, is that making predictions at the daily level gave me more leverage in terms of tweaking the model to get accurate numbers for the month. It shifts the problem from one of accuracy to one of distribution of errors. As long as the model both over- and under-predicted by about the same amount on a daily basis, the aggregated predictions should be close. I still wanted to monitor daily accuracy, but I also knew that with lower volume, and making predictions at the individual channel level, the data would be noisy and making predictions that landed in the "middle" of the daily trend over the course of the month would be our best chance of being accurate over a 30-day period.

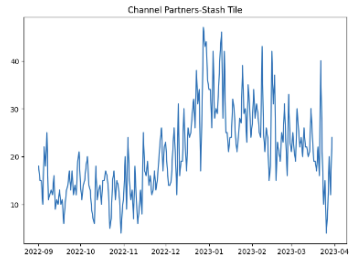
In terms of identifying the period to use to train the model, I started by looking at traffic over the last 3 years. Below is the trend for opened apps over the last 3 years, and there is clearly a shift to a new "normal" last September. I did try generating predictions using all of the data, but the model learned the downward trend from prior to this time and the predictions were a gross underestimate, and many of them were even negative. Thus, I decided to train the model using data from last September onward.

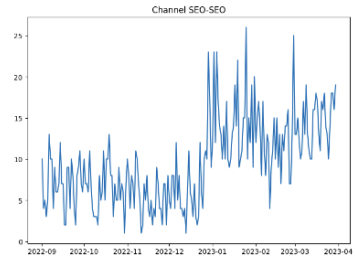
The downside to this approach is that the model cannot learn annual seasonality with less than a year's worth of data.

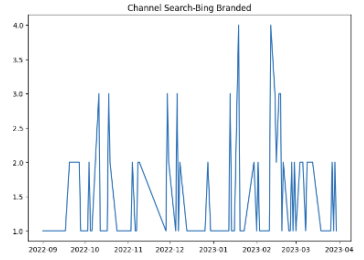


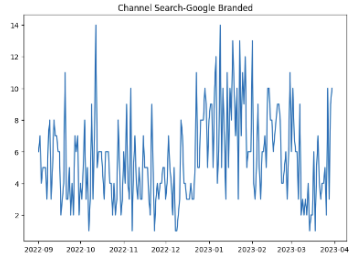
Finally, I started with using visits as the target since between the decrease in volume on our site and trying to make predictions at the individual channel level I wanted to choose a metric that would have enough data for the model to learn from. Some of the channels are currently generating fewer than 5 bound policies per month, so having enough data was definitely a concern. Below are trendlines for daily visits for six of our marketing channels since September 2022:

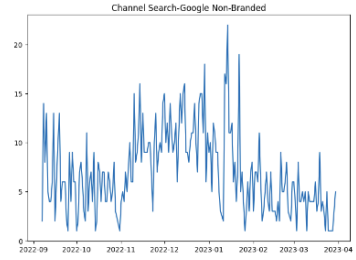












One concern raised by these trendlines was how noisy the data was, and that concern was confirmed when generating predictions using visits as the target, where 15-20% error was the lowest I was able to achieve. While I had yet to put a lot of effort into tuning the model, I decided to try using opened applications at the target in the hopes that this metric would have enough data to train the model but with less noise than visits. In fact, this turned out to be the case, and the accuracy of the predictions we less than 5% in some cases.

Hyperparameter Tuning

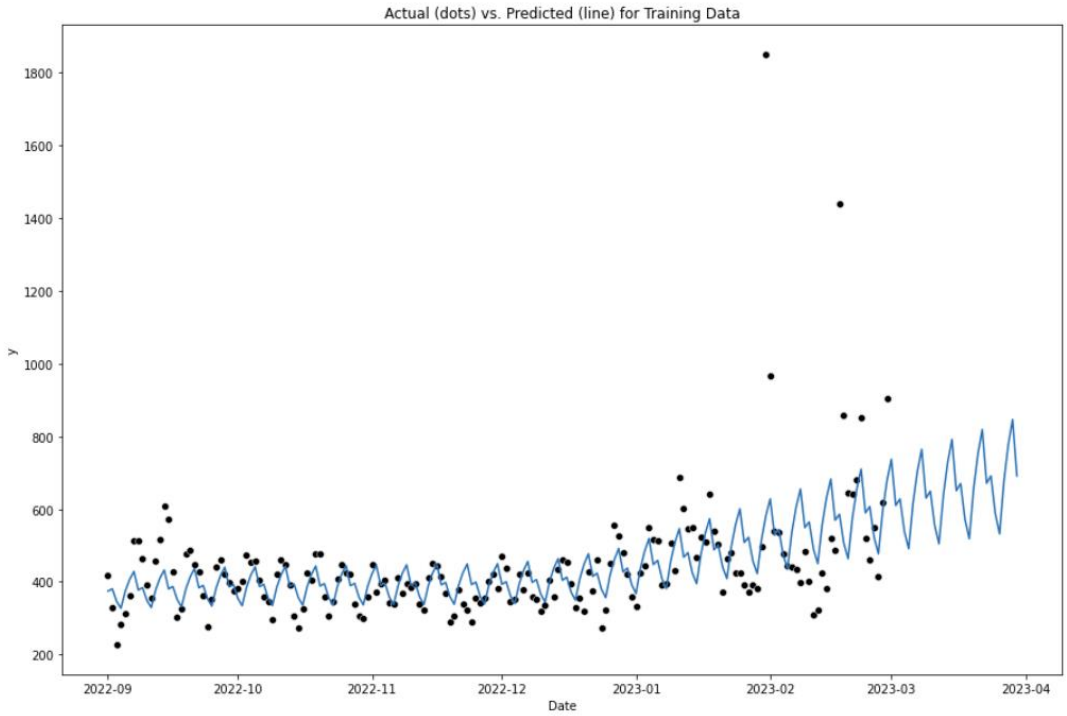
There are a large number of parameters that can be adjusted in the Neural Prophet package, but I was able to tweak just a few to get good performance. One parameter that needed to be specified but not tuned was the 'freq' parameter during model fitting, which tells the model the frequency of the data that the model is consuming for training, which in this case was daily.

In general, there are two components to the Neural Prophet model: a trend component, which discerns an overall trend in the target over time, and seasonality component, which identifies repeated patterns over a period of time, i.e., day, week, or year. The effectiveness of Neural Prophet in particular, and time series models in general, depends on their ability to tease apart these different components and model them separately. As noted above, since I was only working with data since September, the model that I was creating would not have the benefit of modeling a yearly seasonality, as at least two years' worth of data would be needed to separate a yearly seasonality from other trends in the data. Daily seasonality is only applicable for data collected multiple times per day. While seasonality at each of the levels can be manually set to 'False', leaving them at the default value of 'True' allows the model to learn what it can – if the granularity or amount of data is not present to learn seasonality at a particular level, the model will ignore it.

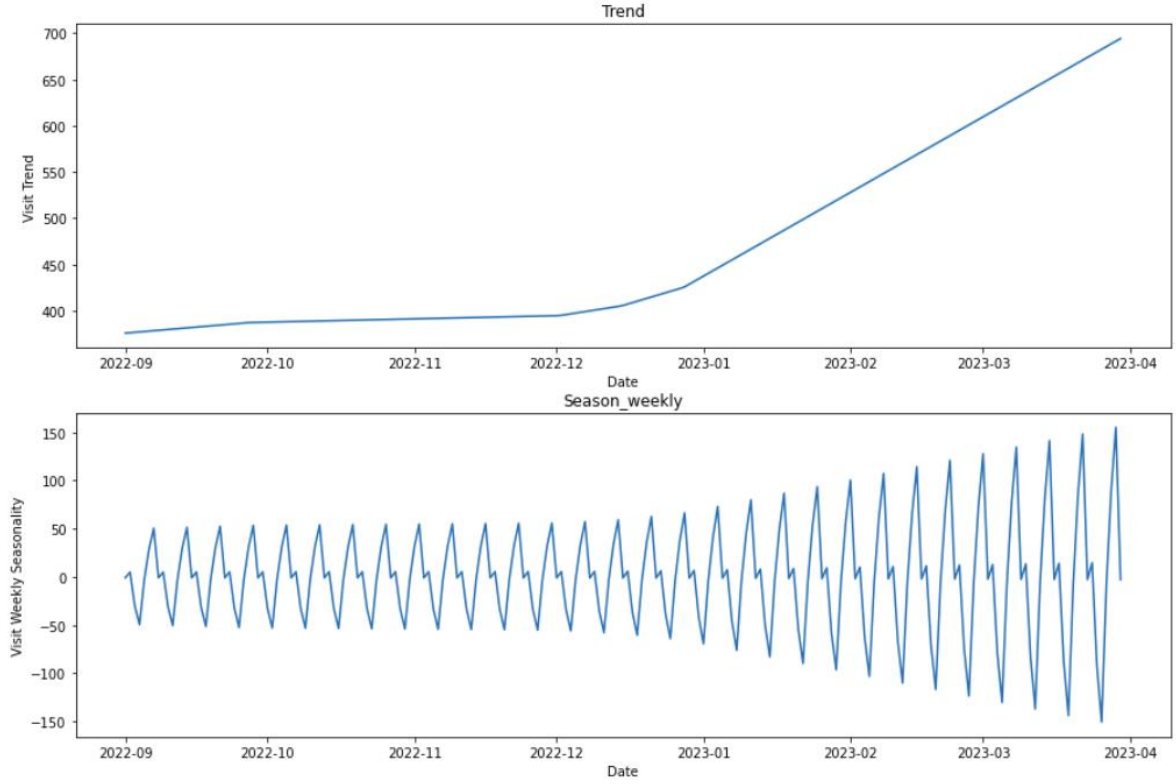
The three parameters that I need to tweak to get a good model were:

- Seasonality mode: this controls how the seasonal component of the model interacts with the trend component. Possible values:
    - Additive seasonality (default): the seasonal component is added to the trend, so that it is assumed to have fixed effects across the time series. This means that if the trend increases, the seasonal variation stays the same.
    - Multiplicative seasonality: the seasonal component is multiplied by the trend, so that the it is assumed that effects change with the trend across the time series. This means that if the trend increases, the seasonal variation also increases, and vice versa.
    - None: seasonality is not included in the model.
- Trend reg: this controls the strength of the regularization of the trend component of the model. A higher value of this parameter results in a smoother trend with fewer fluctuations and more gradual changes over time. The default value is 0.05.
- Seasonality reg: similar to trend reg, this controls the strength of the regularization of the seasonal component of the model, resulting in smoother and more gradual adjustments to the seasonal component of the model. The default value is 0.05.

The way that I tuned these parameters was to use the graphs that the Neural Prophet package generates with the model. Below is a graph that show the actual data points and the predicted values over time:

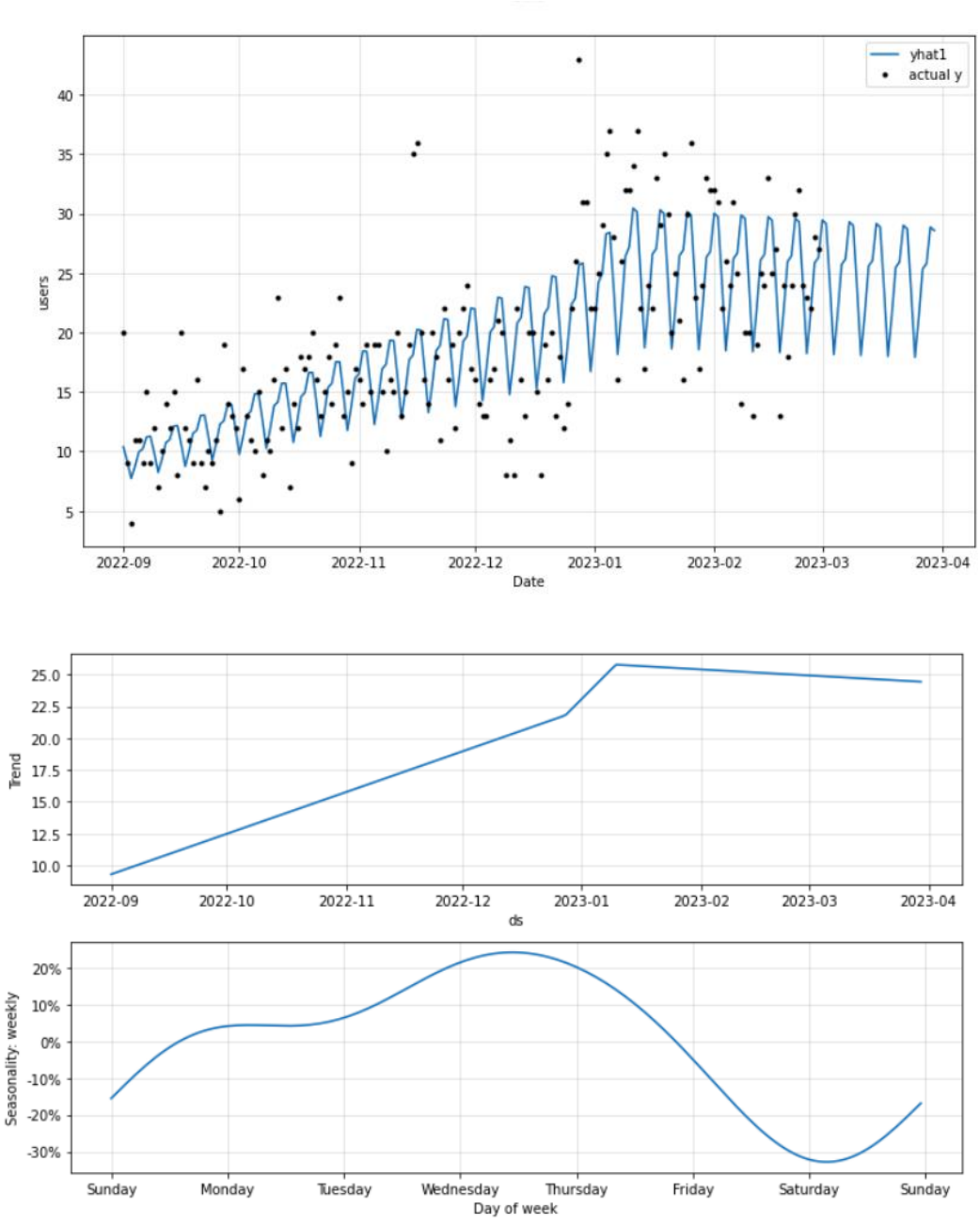


A nice feature of Neural Prophet is that you can also view the trend and seasonal components separately, as shown below:



This is a good example of what happens to the seasonal component when the seasonal mode is set to 'multiplicative': as the trend increases, the variation in the seasonal component does as well. Looking at the data by channel, this seemed appropriate as when the number of started applications increases, so does the variation, and likewise when it decreases.

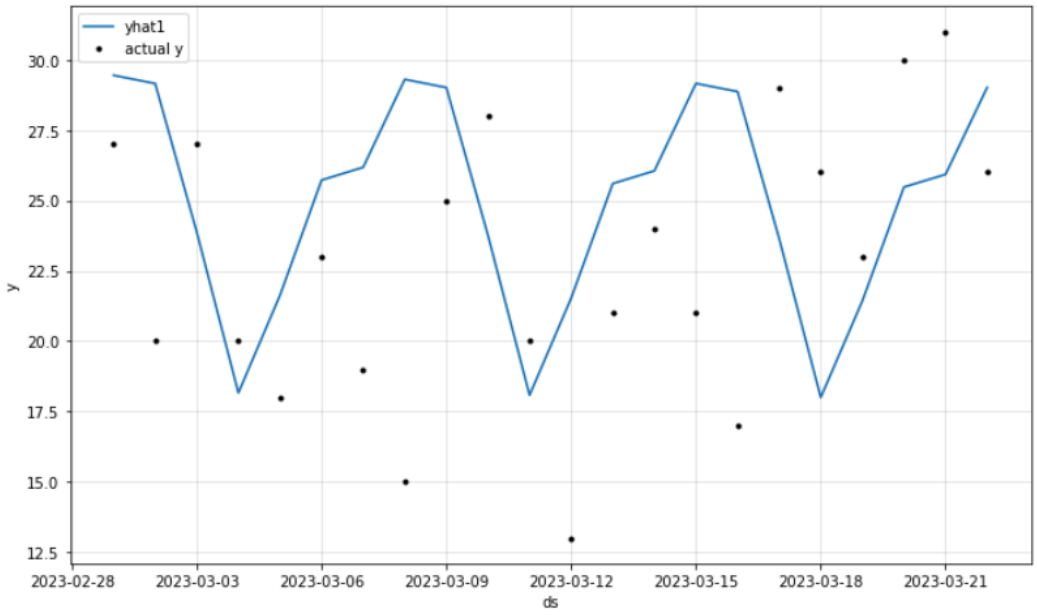
To determine the appropriate values for the trend and seasonal regularization, I used these visualizations for both the training and testing predictions. Below is the visualization for training predictions for a channel, followed by the trend and seasonal components:



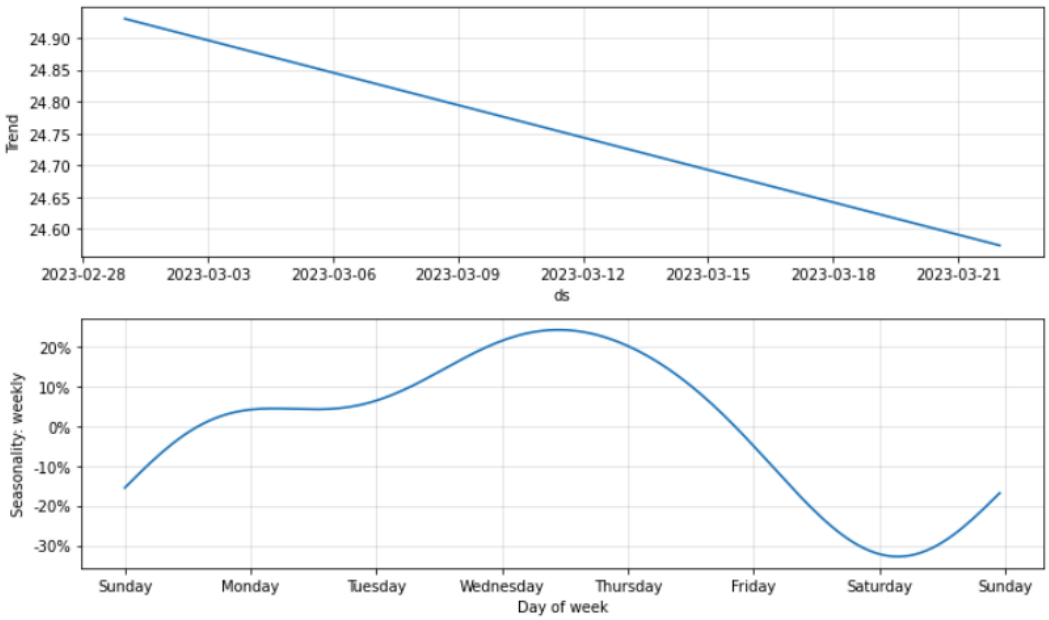
There were a few things that I noticed:

1. The trend component discerned a fairly strong increasing trend, followed by a slight tailing off
2. The model's predictions ('yhat1') generally went through the middle of the actual data points, which was I was aiming for so that the aggregated predictions for the month would be more accurate
3. The variation in the actual data values did seem to increase as the trend grew, confirming the use of the 'multiplicative' value for the seasonality mode parameter
4. While the seasonal mode was set to multiplicative, the actual data points are still often above or below the predicted values

Next I looked at the performance of the model on the test set (actual vs. predicted overall):



Trend and seasonal components:



What I noticed in these visualizations was:

1. More of the data points were below the predicted values
2. The seasonal component seemed to match the variation in the data fairly well in terms of the magnitude

Based on these insights, and what I knew about our data, I decided to try smaller values for the trend and seasonal regularization. The fact that the model picked up on a decreasing trend but didn't follow it closely enough suggested to me that the model trend was too smooth and gradual. This makes sense for our data, where we are trying to make predictions for individual channels in a context with significantly reduced volume. This leads to data that is both noisy, and sensitive to changes. For example, any increase or decrease in spending is immediately reflected in our data. This would apply to other changes as well, such as yearly seasonality. Although we don't have enough data for the model to pick this up, we know that life insurance sales peak at the beginning of the year and decline as the year goes on. Given the volume and level of granularity of our data, we see these changes over the course of a month. Given this sensitivity in our data, it made sense to use less regularization in order to allow the model to follow more recent trends more strongly.

After trying a few values, I found that setting these to 0 yielded the best results. While this introduces the possibility of overfitting, my analysis showed that next month's numbers are influenced much more by this month's trend than anything that precedes this month. By training the model on all the data preceding the last 30 days, and checking the accuracy on the last 30 days, I found the parameter values that work best, and used those to generate the forecast for next month.

Analysis

I computed RMSE, MAPE, and max error for each channel at the daily level for the last 30 days, but ultimately these metrics were not nearly as important as the accuracy of each channel over the month, and even more importantly, the aggregate forecast for the month across all channels. Both are shown in the table below:

Site visits and opened applications:

	Visits				Opened			
Channel	Actual	Predicted	Difference	Percent Difference	Actual	Predicted	Difference	Percent Difference
SEO-SEO	5634	5097	-537	-9.5%	323	263	-60	-18.6%
Partners-Chime	503	657	154	30.6%	103	125	22	21.4%
Direct-Other	17734	16442	-1292	-7.3%	514	513	-1	-0.2%
Affiliates-Other	624	602	-22	-3.5%	59	74	15	25.4%
Direct Mail-Other	70	61	-9	-12.9%	7	11	4	57.1%
Partners-Stash	42	754	712	1695.2%	5	80	75	1500.0%
Partners-Experian	4443	4851	408	9.2%	498	488	-10	-2.0%
Display-Facebook	788	-203	-991	-125.8%	70	-19	-89	-127.1%
Partners-Stash Tile	4805	4961	156	3.2%	527	504	-23	-4.4%
Search-Bing Branded	218	187	-31	-14.2%	11	12	1	9.1%
Partners-Credit Karma	2896	5933	3037	104.9%	626	989	363	58.0%
Search-Google Branded	1102	1433	331	30.0%	98	130	32	32.7%
Affiliates-Ad Practitioners	2645	2507	-138	-5.2%	467	425	-42	-9.0%
CLM-Customer Referrals	59	57	-2	-3.4%	10	9	-1	-10.0%
Search-Google Non-Branded	2019	423	-1596	-79.0%	96	35	-61	-63.5%
Total	**43582**	**43762**	**180**	**0.4%**	**3414**	**3639**	**225**	**6.6%**

Submitted and Approved

	Submitted				Approved			
Channel	Actual	Predicted	Difference	Percent Difference	Actual	Predicted	Difference	Percent Difference
SEO-SEO	253	215	-38	-15.0%	86	74	-12	-14.0%
Partners-Chime	72	90	18	25.0%	6	11	5	83.3%
Direct-Other	343	360	17	5.0%	98	110	12	12.2%
Affiliates-Other	49	62	13	26.5%	6	12	6	100.0%
Direct Mail-Other	5	13	8	160.0%	1	2	1	100.0%
Partners-Stash	4	61	57	1425.0%	0	8	8	#DIV/0!
Partners-Experian	352	357	5	1.4%	62	70	8	12.9%
Display-Facebook	48	-12	-60	-125.0%	6	-2	-8	-133.3%
Partners-Stash Tile	350	349	-1	-0.3%	50	58	8	16.0%
Search-Bing Branded	7	10	3	42.9%	2	2	0	0.0%
Partners-Credit Karma	427	699	272	63.7%	80	164	84	105.0%
Search-Google Branded	79	108	29	36.7%	22	40	18	81.8%
Affiliates-Ad Practitioners	352	327	-25	-7.1%	118	106	-12	-10.2%
CLM-Customer Referrals	9	8	-1	-11.1%	2	3	1	50.0%
Search-Google Non-Branded	61	24	-37	-60.7%	10	6	-4	-40.0%
Total	**2411**	**2671**	**260**	**10.8%**	**549**	**664**	**115**	**20.9%**

Bound Policies:

	Bound			
Channel	Actual	Predicted	Difference	Percent Difference
SEO-SEO	34	31	-3	-8.8%
Partners-Chime	0	1	1	#DIV/0!
Direct-Other	43	43	0	0.0%
Affiliates-Other	1	3	2	200.0%
Direct Mail-Other	0	0	0	#DIV/0!
Partners-Stash	2	0	-2	-100.0%
Partners-Experian	12	12	0	0.0%
Display-Facebook	1	0	-1	-100.0%
Partners-Stash Tile	12	12	0	0.0%
Search-Bing Branded	1	1	0	0.0%
Partners-Credit Karma	11	30	19	172.7%
Search-Google Branded	4	15	11	275.0%
Affiliates-Ad Practitioners	45	40	-5	-11.1%
CLM-Customer Referrals	3	3	0	0.0%
Search-Google Non-Branded	3	2	-1	-33.3%
Total	**172**	**193**	**21**	**12.2%**

A few comments about the results:

1. The target (opened applications) was off by less than 7%. This is the most important metric for two reasons:
    a. It's the basis for all the other metrics, which are derived using conversion rates.
    b. It's the only one the model can really be held accountable for. For example, the forecasted number of opened applications was off by less than 7%, but the number of approved applications was off by over 20%. That error is due to the inaccuracy of the funnel conversion rates compounded from opened to approved application, i.e., submit rate and approval rate.
2. Visits were much more accurate using open apps and dividing by the start rate (opened applications/visits) than trying to predict them directly. This is due to both the accuracy of the opened applications predictions and the historical start rate.
3. Approved applications had the worst accuracy, but this worked out well since this is the only metric that is not the basis for any marketing channel payout. We just need it to generate the bound policies forecast.
4. Predictions at the individual channel level we much less accurate than at the aggregate level. For the purposes of this forecast, that was fine, but this represents an area to improve in future iterations as improvement at this level will help to tighten up the aggregated forecast and better predict marketing costs since the pay structure for each channel varies, and thus a more accurate forecast at the channel level will help us to better anticipate cost per acquisition (CPA).

Conclusion

The model shown above was used to generate our marketing forecast for April, in conjunction with the marketing team, i.e., they generated their forecast as they'd done in the past and we discussed and reconciled differences. Next week we will look at the model predictions against our actual April performance and do some analysis of factors that the model did not account for, and if/how any of those could be incorporated into the May forecast.

Assumptions

I think the main assumption, which generally seems justified, is that recent behavior is most predictive of future trends, vs. previous historical trends. This is perhaps less true since September, but even during the last 8 months, there has been a good amount of fluctuation in our funnel metrics.

Limitations

A major limitation is not being able to use more than 6-8 months of data to train the model, which prevents us from being able to leverage a yearly seasonality component in the Neural Prophet model. One thought I had was to try to normalize the 2-3 years of worth data that we have access to by identifying change points in the data, and finding a scale factor for periods between change point that would put the historical data on a similar scale to our current metrics. This could allow me to use more data in training our model and include a yearly seasonality component and potentially improve the forecast.

Challenges

A big challenge for this use case is that we are a small company and we have so many changes going on simultaneously. In addition to changes in marketing spend, we are constantly running experiments to test improvements to the site and user experience, making changes to pricing or eligibility, and identifying and fixing bugs in the system. And these are just changes that we have control over. External changes include economic factors such as inflation or recession, changes in consumer behavior based on the pandemic, and the fact that we are a tech start up in an environment where securing investor funding has become extremely difficult.

Another challenge was generating forecasts for each channel using the same model parameters. While the channel-level predictions could be improved, this would likely require that the model be tuned for each channel. Given that the aggregated forecast was accurate enough, this was not something we felt was needed at this time, but there was certainly a trade off between how hands-on we needed to be vs. accuracy at the channel level.

Another challenge was the coding part of this project. Importing data from various sources, structuring the data to be able to loop through the channels and generate a separate forecast for each, and then aggregating the forecasts and measuring the accuracy of each as well as the aggregate, and saving and sharing the forecast with colleagues was a good experience of writing code for an end-to-end model.

Future uses/additional applications

- **Marketing costs:** three of our channels are paid on a per click basis vs. any of our funnel metrics, and a similar model was used to generate a cost forecast for these channels based on daily cost data for these channels.
- **Conversion rates:** while the model target is opened applications, the rest of the forecasts is based on funnel conversion rates for each of the channels. So even if the model is accurate with respect to the target, the rest of the forecast could be off if the conversion rates are off. One thought is to use a time series model to generate the conversion rates by channel.

Recommendations

My recommendation was to use the model for the April forecast, and continue to develop the model to get more accurate forecasts for individual channels.

Implementation Plan

As noted above, this forecast was already used to generate the April forecast, and will be used again for May. On an ongoing basis, some time will be needed each month to train the model and generate the forecast. A few additional improvements have been made since the first version of the model:

- **Generating a cost forecast:** three of our channels are paid on a per click basis vs. any of our funnel metrics, and a similar model was used to generate a cost forecast for these channels based on daily cost data for these channels.
- **Integrating funnel conversion rates:** since conversion rates per channel are updated each month by the marketing team and used to make forecasts for all metrics besides opened applications, we wanted to get away from hard coding them into the model, which is what we did in the first version. I was able to set up a connection to a google sheet that the marketing team uses and import the conversion rates when the marketing team updates them.
- **Generating scenarios:** based on anticipated changes, we generate some scenarios in terms of how we think performance over the next 30 days might be impacted. For example, for April we anticipated some changes to our underwriting rules to increase approval rates, and potentially decrease our bind rates. Thus, we generated a base forecast, and then created forecasts with all combinations of the following changes:
    - Approval rate: increase by 2%, 4%, or 6%
    - Close rate: decrease by 2%, flat, or increase by 2%

  Finally, as a team we looked at forecasts based on each of these scenarios, and then decided which one to use. Like integrating the conversion rates from a google sheet that the marketing team can update as described above, I will do the same with a list of anticipated scenarios for the following month. These integrations allow us to automate the forecasts while still providing flexibility in integrating short-term inputs that will affect the forecast.

Ethical considerations: the only thing that I can think of in terms of ethical considerations is that various marketing channels will reach different types of people, and thus the forecast could guide the company to focus on marketing to certain groups of people to the exclusion of others. I think the broader issue is balancing making good business decisions without systematically excluding certain groups of people, especially when the product we're selling is financial protection.

Ten Questions an audience could ask:

1. Can we improve the channel level predictions?

   I believe that we can, but to do so I'll need to tune the model for each channel rather than running the model with the same parameters for all channels. This approach was fine for v1 as it yielded good results at the aggregate level. But more accurate forecasts at the channel level will allow us to better forecast costs and will make the aggregated predictions better as well.

2. What other parameters can be tweaked to try to improve the channel level predictions?

   To begin with, I'll try tuning the parameters that I've already identified at the channel level. The challenge will be finding an efficient way to do this. Using the forecast graphs for each channel is not a sustainable solution as it is too hands-on. Using the error for predicted vs. actual opened applications and identifying the best combination of trend and seasonal regularization parameters is probably the best way to start.

3. Can we create a better model without Neural Prophet?

   I think it's quite possible to create a better model using a Deep Learning time series model. While this option would likely take more time up front, optimizing forecasts at the channel level would likely be much more automated as we can use early stopping or other regularization techniques to stop training the model when the validation accuracy starts to degenerate.

4. How much can this process be automated?

   Automating the channel level optimization is probably the most important piece as automating this forecast is the goal. A process for this that requires a lot of time each month simply shifts the burden from the marketing to analytics team.

   Other parts of the process have already been automated by pulling inputs from a google sheet maintained by marketing. This means that marketing can provide guidance and input to the forecast without additional coding.

5. Can the full data set be normalized to current levels in order to leverage yearly seasonality?

   This has been discussed, and I think it could be done. We'd need to find changepoints where the trend changes and divide all the data into separate time periods based on those changepoints. For each time period, we'd need to identify a scale factor that relates that time period to the current period, and then normalize each time period separately and create a single data set from the normalized time periods. The part that I think would be tricky about this strategy is knowing the base to normalize to. Right now, we are using all data since 9/1/22 in our model, but there appears to be a change point around the first of the year even for this data. It would likely take some trial and error to balance normalizing to be able to discern yearly seasonality with leaving enough variation in the data for the model to learn from. While this approach is possible, due to the time it would take and uncertainty about the results it would yield, it would not be my first option for trying to improve the model.

6. Can we account for holidays?

   There is a parameter in Neural Prophet to account for holidays, but some research would need to be done to understand which holidays the package accounts for and how holidays affect the sales of life insurance policies. In addition, we'd need to understand how Neural Prophet treats the time period before and after holidays, which can also be affected, and what's most appropriate for online life insurance.

7. Can we model revenue using time series?

   Yes, the goal would be to be able to not just forecast bound policies, but the actual amount of revenue generated by policy sales by channel. The cost of a policy changes based on the user's risk class, the term of the policy in years, and the coverage amount, and ideally we'd be able to predict revenue at the channel level vs. simply using the average policy revenue across all channels.

   However, this gets trickier because most channels don't have daily data for bound policies. For example, in March only three of our channels had more than 30 bound policies for the month, while 9 channels had fewer than 10. This is why we're using opened applications as the current target. One possibility is to aggregate the data to weekly or monthly level, but this significantly reduces the amount of data that the model has to learn from. I will likely start with using data at the daily, weekly, and monthly level and see which seems to work best and how good the forecast is. It might also work to make predictions at different levels for different channels, since the overarching goal is to make a forecast at the month level.

8. Can we model conversion rates using time series?

   Right now, someone on the marketing team generates conversion rates for each channel for the upcoming month, I think mostly based on the conversion rates from the most recent month, combined with other inputs such as whether we are increasing or decreasing spend for a given channel in the upcoming month. The way we're currently generating the forecasting depends heavily on these numbers as the model only uses opened applications as the target and generates the rest of the funnel numbers from this prediction using the conversion rates. This is another way in which the forecast could be more automated, but also potentially where it could be more accurate.

   However, predicting conversion rates is just a roundabout way of getting to what we really want, which is the number of applications at each level of the funnel. We are currently using opened applications as the target for the model, and then using conversion rates to get the other parts of the funnel (visits, submitted, approved, bound). While I could use, for example, the submit rate (submitted applications/opened applications) as the target and generate a forecast for submit rate for the next month, and then use the predicted submit rate to determine how many predicted opened applications will be submitted, it would be more straightforward to simply create a model with submitted applications as the target.

   In terms of next steps, it seems that tuning the model at the channel level for opened applications, as described above, is the place to start. Once that problem is solved, scaling the solution to other parts of the funnel would be the natural next step.

References

Neural Prophet:

- https://towardsdatascience.com/in-depth-understanding-of-neuralprophet-through-acomplete-example-2474f675bc96
- https://medium.com/@cuongduong_35162/facebook-prophet-in-2023-and-beyondc5086151c138

Time series:

- https://towardsdatascience.com/time-series-forecasting-deep-learning-vs-statistics-who-winsc568389d02df