anodot

THE ESSENTIAL GUIDE TO **TIME SERIES FORECASTING**

Part II: Design Principles of an Autonomous Forecasting System

INTRODUCTION

In setting out to develop a machine learning-based forecasting system that could ultimately be delivered to customers as a turnkey SaaS application, we at Anodot realized that there was no existing blueprint for how to build such a sophisticated system. There were no other systems to study or reverse engineer; this was a greenfield solution area. Therefore, we pulled together a multi-disciplinary team of experts on machine learning, DevOps, application development, and more to create our own blueprint.

The most important aspect of forecasting is accuracy. Any forecasting algorithm, project or product is measured by how accurate it is. Very simply, the more accurate it is, the better it is. There are a lot of design principles that can help a forecasting system be more accurate, and especially forecasting based on machine learning. For example, forecasting is usually based on historical data of the metric being measured, but there are other relevant influencing factors that can increase the accuracy of a forecast. Anodot has designed how to include these factors into our system. And while additional factors often help produce a better forecast, not all factors have that effect. Our product considers how to quickly and efficiently evaluate whether a factor is relevant and weed out the ones that provide no value.

Other important design considerations take into account how to work with seasonal data patterns, how to identify and account for anomalies in the historical data as well as different times series behavior, and why and how to create an ensemble of learning models. We also consider why and how to build business-driven bias into a learning model in order to match a business use case.

This document, Part II of a three-part series on time series forecasting, outlines the design principles we have incorporated into Anodot Autonomous Forecast™. Part I of this series delves into the value of autonomous machine learning-based forecasting, and Part III outlines the architecture of the system that Anodot has built to deliver a sophisticated yet turnkey forecasting solution.

There are many principles in designing an autonomous system that can affect the accuracy of a forecast. Anodot has thoughtfully considered these design principles and incorporated them into our product, Autonomous Forecast.

DESIGN PRINCIPLES

At a high level, the illustration in **Figure 1** shows the process and components of a system that will generate the most accurate as possible forecast for growth or demand

Generically speaking, the components of a machine learning-based forecasting system will do the following:

- **CONNECT THE DATA –** choose internal data to use, plus external data and/or events, if desired
- **SELECT THE FORECAST METRICS –** specify what to forecast, over what time period
- **DO AUTOMATIC DATA PREPARATION –** correlate the data sets, look for anomalies that can affect the forecast
- **TRAIN THE MACHINE LEARNING MODELS –** use a variety of algorithms to derive a forecast and then test for accuracy, select the best models for the use case
- **CREATE A CUSTOMIZED MODEL –** using an ensemble of models, create a persistent model that can be used to forecast the metric on demand or continuously
- **REVIEW THE CUSTOMIZED MODEL –** conduct frequent reviews of the custom model to ensure it is optimized
- **MAKE A FORECAST –** using the customized model, create an actual forecast either on demand or continuously
- **CONSUME THE FORECAST INSIGHTS –** send output of the forecasting process to a dashboard, reports, alerts, or to other information systems

That's the blueprint at a high level. Let's go into some detail of the considerations behind those steps, including what Anodot does to derive the best possible forecast for a specified use case.

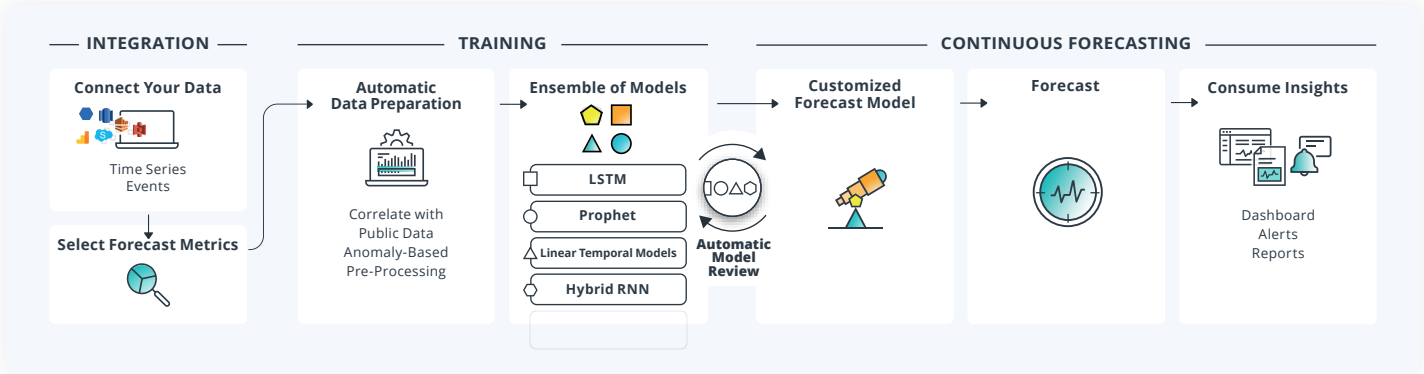


Figure 1. *A blueprint for a ML-based forecasting system.*

RELEVANT DATA: METRICS AND EVENTS

Time series data is used to forecast the future values pertaining to business growth or demand. A company has historical time series data that comes up to the present time and wants to know what the values of that time series will be in the future, given some horizon. To derive a machine learning model that can do that forecasting accurately, models (algorithms) are trained based on that historical data.

Say a company has five years of revenue data and the CFO wants to predict the upcoming year's revenue. A model is trained to look at the patterns that exist in that five years of data, and then once it learns whether there is a repetitive pattern that happens there, the model can try to forecast the future based on that learned pattern.

An advantage of machine learning algorithms is that they can take in as inputs the time series that is being forecast, as well as other internal or external data that might be significant in forecasting that time series. Suppose an eCommerce company wants to forecast sales of shoes. Historical patterns of sales of shoes will certainly help in training the forecast model. The company also can input data about sales of adjacent products like socks which might be related to shoe sales. And since this is an eCommerce company, another important metric might be the number of users coming to the website. The algorithm can determine if there is a correlation of these factors to the sales of shoes or not.

External factors could be important too, such as general economic data like the Consumer Confidence Index. If the economy is growing and people feel good about spending money, they might buy more shoes. Likewise, if their consumer confidence is slipping, they might pull back on buying new shoes. Another example of a potentially relevant external factor is how many people did a web search for shoes over a certain time period. There's a lot of additional information that can help forecast that one thing that the company cares about, and the advantage of machine learning models is that they can take as many inputs like that as desired and use them to understand how to forecast more accurately.

Events can have an impact on training the model as well. For example, a Black Friday sales campaign is likely to cause a spike in sales. The more discrete events like this that can be entered into the model, the more accurate the forecast of the future will be.

To understand whether Black Friday has an impact on sales, the company obviously needs to have historical data where Black Friday was there. The model needs to see past years that include Black Friday sales metrics. The algorithm must be told that this is a day when something special happened. One year of data containing a Black Friday sale isn't enough. The model can't learn from an event that only happens once, but it can learn from an event that repeats itself at least twice or three times. The result is that the model projects a spike in sales for the upcoming timeframe of this thing called Black Friday, as shown in **Figure 2**.

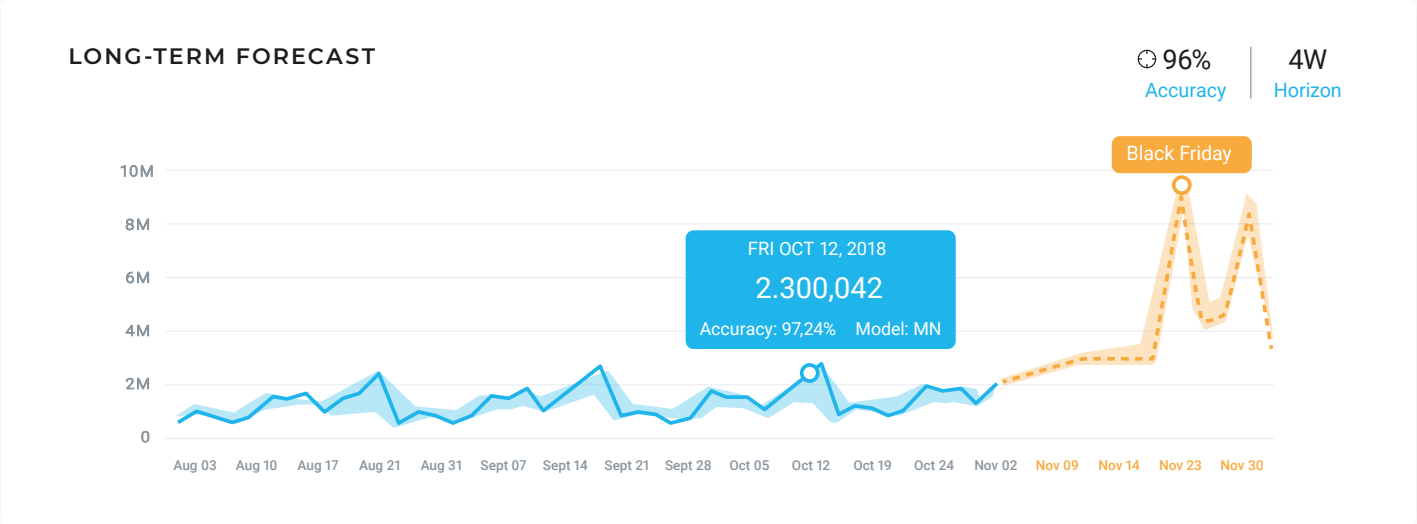


Figure 2. *Forecasting a spike in sales for Black Friday.*

Now consider that the company has a sale event that might happen five times a year. The algorithm understands there is an event within the time series it is trying to forecast and will learn whether that event tended to have an impact on sales. And then when trying to forecast 30 days in the future, the assumption is that it's known beforehand that this sale event will occur. The algorithm can be told that 30 days from now there is a sale event that's going to happen and it should take that into account when doing the forecast.

For an event to be statistically significant, the algorithm has to see it at least several times and be aware that it is an impactful event. The more times the algorithm sees the event, the more confidence there is in saying what will probably happen in the future when this event happens again.

Some events that have a big impact on a time series are actually "one-offs" that are not learnable because 1) it only happened once, and 2) there probably wouldn't be any advanced notice if it happens again. An example of such an event would be a massive network failure that knocks the eCommerce site offline for several hours. While the impact of the event is large, it can't be forecasted. Therefore, such an event needs to be identified as an anomaly to ensure that the algorithm isn't thrown off by it when it is being trained to forecast.

The illustrations below show the effect on accuracy when we do take other factors into account, in this case, an event. In **Figure 3**, the solid line is the actual measurement. The dash line is the forecast that was made for each one of these hours of the day before. This is testing the algorithm. The model is trained up to a point and then it forecasts the future points. Once we know what actually happens, we can measure the accuracy, which is basically the difference between the forecasted value and the actual value, normalized to be a percentage, and then averaged over the entire period[1].

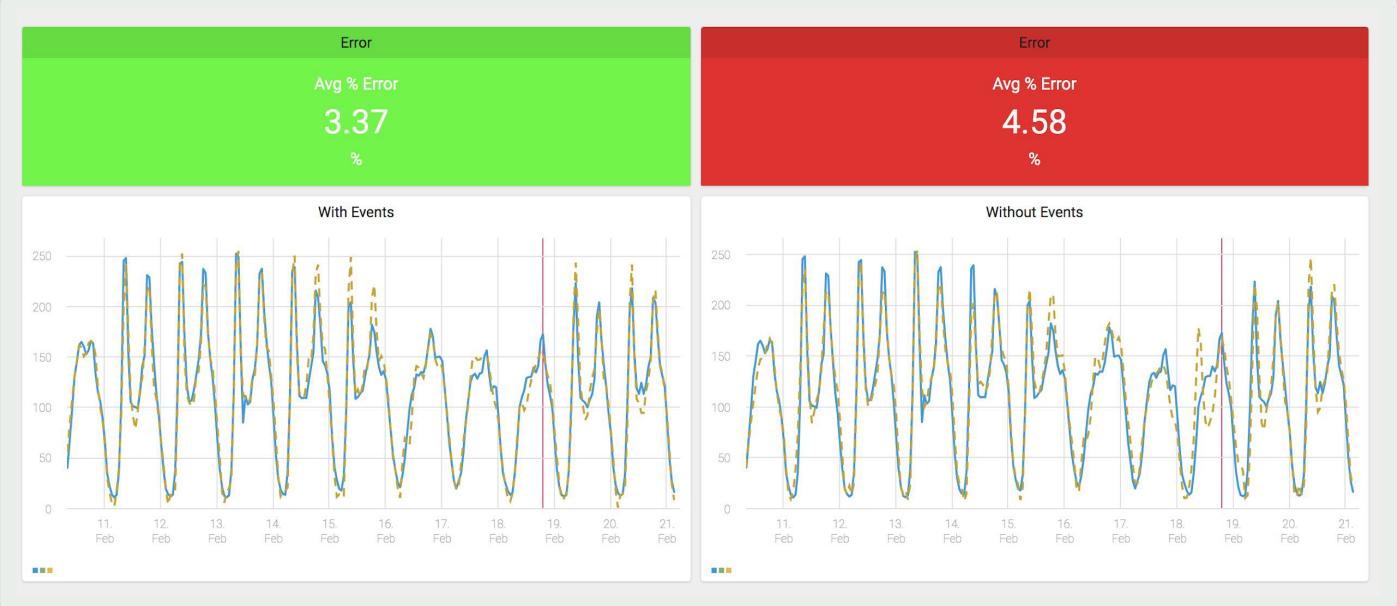


Figure 3. *Measuring accuracy of a forecast model.*

1. There are several accuracy measures for forecasting. The one described here is called MAPE (Mean Average Percentage Error), and it is one of the most commonly used error for forecasting.

In the right side of the illustration of **Figure 3**, the forecast model was not given information about an influencing event that occurred in past history—in this case, Washington's birthday. Because the model didn't know about the event, it couldn't understand the effect of the event on the measurement and therefore didn't train to understand it. The result is that it makes the wrong forecast on that day and the error increases, certainly for that particular day as well as overall.

In the left side of the illustration of **Figure 3**, the model has been informed of the influencing event (the holiday) so it could account for it in the learning process. The result is a much more accurate forecast, both for that particular day and overall.

Let's summarize our thoughts on the impact of events on time series forecasting. There are certain events that a company knows are going to happen because it is in control of them, like an internal marketing campaign, or the events are well known, like Black Friday, Cyber Monday and other holidays. There are events that come from external sources that help a company understand its own forecast; for example, a major hurricane in a weather forecast that will surely impact sales for a short while. And there are events that can't be anticipated and nobody knows they will happen until they do, like a massive network outage. All these factors can impact a forecast and they must be considered when training the forecast model.

TESTING THE ACCURACY OF A FORECASTING MODEL

We at Anodot place a high priority on deriving an accurate forecast. This begs the question, how is accuracy of a forecasting model determined? This is done by taking the available historical data and splitting it into two parts. Let's say we have five years of historical sales data. We would use the first four years as the training data and the fifth year as the validation data.

The model is trained with only the first four years of data and then the model is used to predict the next year of sales. This prediction is compared to the actual sales from the last year of the historical data, which was held back from the training process. Then we measure the error—how close the prediction came to the actual numbers. The smaller the error value (expressed as a percentage), the more accurate the forecast model.

When splitting the data into the training set and the validation set, it's critical to split it based on the appropriate time interval. This accounts for seasonality of the data as well as events that occur at the same time each year, like Black Friday. In our example above, we wouldn't split the data as four years and six months as the training data and the final six months as the validation data; this simply isn't symmetrical.

In addition, time series data can't be taken out of sequence. We can't train on the sales data from 2015, 2017, 2018 and 2019 and reserve the data from 2016 as the validation data. When dealing with time series data, the models are trying to capture the temporal dynamics of that time series. The patterns are temporal and they have dynamics associated with them, and it's usually continuous. So, something can't be taken from the middle of the series and used as the validation or test data.

DETERMINING RELEVANT INFLUENCING FACTORS: COMBATING THE "CURSE OF DIMENSIONALITY"

It's true that many potential measurements and events can influence the forecast of something. In forecasting, say, sales of shoes, there can be a thousand different things that might influence the forecast. One of the challenges of building a machine learning-based forecasting model is to understand which factors are influencing factors and which ones are not. The goal is to design something in the system that will prune out or only keep the factors that have a potential for helping make an accurate forecast.

Theoretically there could be a million things that might possibly influence a forecast, but if too many factors that truly have no influence are input into the machine learning models, the models have difficulty in the training process. The fact is, it doesn't work to input an unlimited number of potential factors and let the algorithms figure out on their own what is relevant and what is not. Something must be designed to do this more efficiently, and here's why.

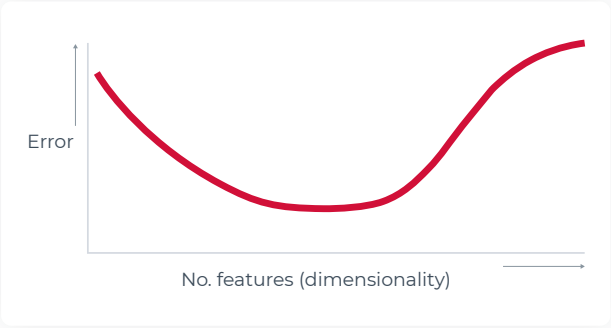


Figure 4. *The Curse of Dimensionality – as the number of Features increases, with a fixed set of samples, the error starts increasing.*

In machine learning for time series data, there is always a finite amount of data to learn from. After all, historical data has a starting point and it's not possible to simply "create more data" to make it infinite. When models in training are presented with a lot of dimensions and a lot of potential factors, they encounter something called the Curse of Dimensionality. It's a condition with a weird name but nonetheless it's a legitimate concern in data science. The curse of dimensionality says that as we have a finite amount of training data and we add more dimensions to that data, we start having diminishing returns in terms of accuracy. There is a point where errors in the forecast results start to increase because we are asking the models to learn about too many factors with a finite amount of training data.

Unfortunately, there's no formula or rule of thumb for knowing how many factors are too many before reaching the upward trajectory of the Curse of Dimensionality. Thus, Anodot has designed a procedure into our system – another algorithm – that takes the burden of thinking about the Curse of Dimensionality away from the user. The process involves adding a dimension to the learning data and then training the model and measuring the error. We look at the factors one by one instead of looking at them all together, and we discard the factors that seem to have no influence on the dataset we are trying to forecast.

For example, let's say a taxicab company in New York City wants to forecast its number of fares for a week to determine how many cabs and drivers to schedule for work each day. The weather forecast for New York City in the week ahead might have an influence on cab usage. If rain is in the forecast for NYC, more people might be expected to hail a cab rather than walk a few blocks. We can use historical data about cab rides and actual weather patterns to train a forecast model and see if that correlation is very weak or very strong. If weak, throw away the NYC weather data; if strong, keep it.

Another factor might be the value of the stock market, so we test the metrics of the Dow Jones Industrial Average with historical data of cab rides. Again, if the correlation is weak, toss out this factor, but keep it if the correlation is strong. This process continues testing factors one by one until all are tested—potentially millions of elements. What is sacrificed in the name of efficiency is the ability to test factors together, such as whether stock market data plus weather data together affect the number of cab rides. This procedure may lose some potential accuracy in the forecast model but it's an efficient way to narrow down the factors that might have the most impact on the forecast without hitting the Curse of Dimensionality. This is all done automatically so there is nothing for the user to think about.

USING LOCALITY SENSITIVE HASHING TO INCREASE EFFICIENCY

Even with the process described just above, there could be challenges with training the forecast model. Suppose we have millions of these comparisons. These correlations might be very expensive computationally if we don't have an infinite amount of resources. Anodot uses a technique in machine learning called Locality Sensitive Hashing, which helps make our technology more efficient in computing this correlation between the target and the measurement time series.

This approach takes each time series by itself and creates a computation on it to derive a single value—a hash. If we compute the hash for two different time series and the hash values are numerically close to each other, we assume that the series are somehow correlated. The advantage of this approach is that it quickly prunes out the factors that are probably not similar, and if we are comparing millions to the time series we want to forecast, we can probably prune out 99% of them very quickly. Then we are left with the remaining 1% of the factors where we can go ahead and do an exact computation to make sure that they are really close. In this case, the false positive rate is typically small and it's something that can be controlled.

WORKING WITH SEASONAL DATA PATTERNS

Seasonal patterns often can be found in time series data. For example, if we are looking at sales data for some sort of consumer product, let's say video games, we would expect to see an annual increase in sales each November and December to account for the Christmas holiday, with sales dropping off again in January. This pattern is repeated year after year and is deemed to be seasonal in nature.

If there is seasonality in time series data, multiple cycles that include that seasonal pattern are required to make a proper forecast. Otherwise, there is no way for the model to learn the pattern. This is one element that makes working with time series data harder than other typical machine learning models in which the order of samples doesn't matter. With time series data, the order really, really matters.

But it is not just about having enough training data that has multiple instances of the seasonal pattern. It also impacts the settings of the learning algorithms being used. For example, a deep learning algorithm known as LSTM would produce an accurate forecast if one of its parameters (called "lookback") is set to be larger or equal to than the length of seasonal pattern, and very poor results if not. Another well known classical forecasting algorithm is ARIMA. It too requires knowledge of the seasonal pattern length in order to account for it in the estimated forecasting model. Without this knowledge, an ARIMA model may produce very poor results.

IDENTIFYING AND ACCOUNTING FOR DATA ANOMALIES

Anomalies, by definition, are things that we can't forecast. If we can forecast them, they're not anomalies. For example, in 2017 there was a big outage of AWS that knocked many large Internet sites offline for several hours. An outage of this nature is unplanned and unpredictable, but nevertheless impactful on the businesses whose websites or applications were down during that time. Therefore, we must identify and account for data anomalies because if they are overlooked and ignored, the forecast results can be really bad.

We give our machine learning models historical data to train from, and there may have been anomalies in that data. Continuing with the eCommerce example of selling shoes, it's possible that the shoe company website was down during a network outage. It was unexpected, and certainly we can't forecast when or if it will happen again, but we have to deal with that exception in the data history. If we let the forecasting models train without taking the anomalies into account in some meaningful way, they can actually hurt the accuracy of these forecasting models in a dramatic way.

Figure 5 shows an example of a time series where there are three anomalies in the data. The model was trained with these anomalies in place with no identification or explanation, so the model assumed this is the normal data pattern and tries to fit a mathematical function to all the data. Unidentified anomalies can really distort what we try to learn about the entire time series. Consequently, the model was not able to forecast the anomaly nor numerous other data points along the timeline that were not anomalies and were perfectly okay. This series should have been forecastable.

The blue line shows the actual data with the anomalies (i.e., the data spikes) and the orange line shows the forecast made with the anomalies in place. It's obvious to see how inaccurate the overall forecast is because of the unexplained anomalies.

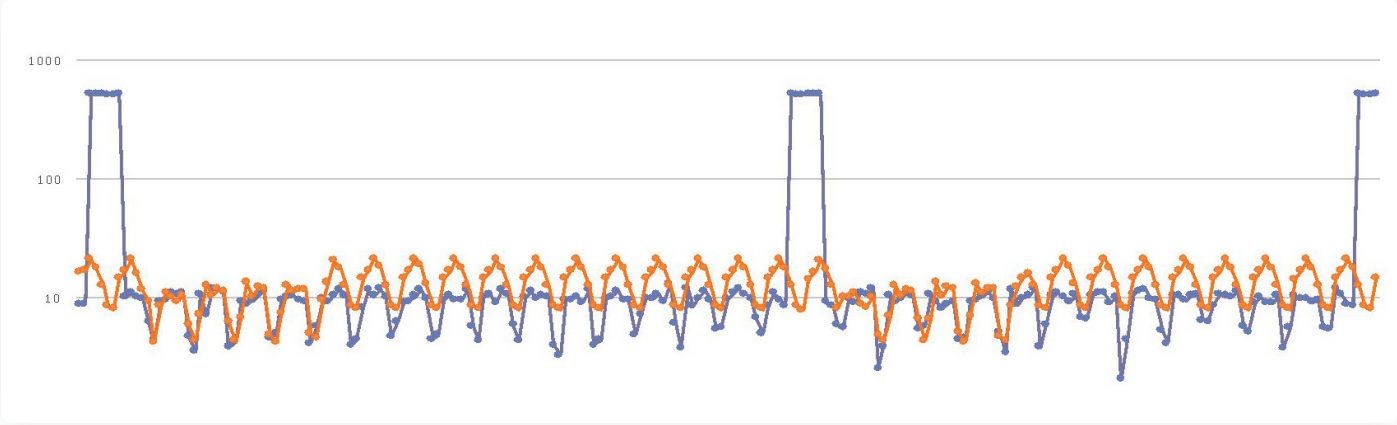


Figure 5. *The results of forecasting with unexplained anomalies in the data.*

Figure 6 below shows the same time series with the same anomalies, but this time the training model was made aware of the anomalies and did not assume they were part of the normal data pattern. The resulting forecast is much more accurate.

Once we understand that we have to handle these anomalies in the data history, the question is how to do it. Completely ignoring them might not be the right thing to do because the anomalies might have signal in them, so we have to get clever in how to treat the anomalies. The first order of business is to go through the historical data and discover any anomalies. Anodot does this by running our anomaly detection algorithm to discover any anomalies in the training data as a first step—before we let the algorithm start learning how to forecast that data. Once anomalies are discovered, we partition them into two cases.

In the first case, we look through external data for influencing events or measurements, or some pattern that can potentially explain the anomalous data. For example, if we see an anomaly in the data, and we see that on that day there was an extreme weather event like a blizzard, we can assume that event accounts for the anomalous data and then use that factor in the training data in order to learn from that factor. We also have to ask, is that factor something that we have beforehand? When we are forecasting tomorrow's data, can we get an external weather report to see if another blizzard is predicted to occur? If so, we have an explanation factor that we have available whenever we are doing the forecast, and not after the fact. If we can forecast these factors, they are actually super useful because they look like an anomaly but really aren't an anomaly.

Weather is a hard factor to predict more than a few days out. If a company wants to forecast something 30 days out that could easily be affected by weather – say the number of cab riders – the weather forecast really shouldn't be used in training the model. In this case, there is another way to treat anomalous behavior. We weigh it down to dampen the effect it has on the forecast, so that the overall forecast isn't thrown out of whack by one unexplained anomaly.

Another way to treat a scenario like this is to make the forecast operational by continuously forecasting. Say we want to forecast not just 30 days in advance, but days 1 through 30. Every day we look at the adjustment of the forecast for the next day, two days from now and three days from now. We might not know if there will be disruptive weather 30 days from now, but the weather forecast for two days from now should be good enough to project weather as a factor in the time series forecast. Therefore, the forecast can be continuously changing to a higher accuracy level.

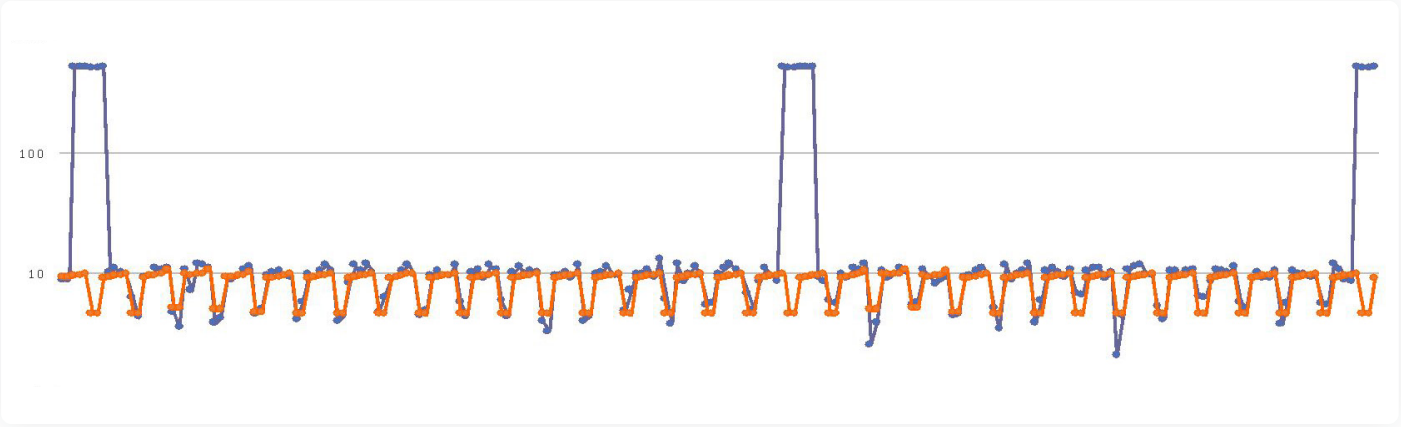
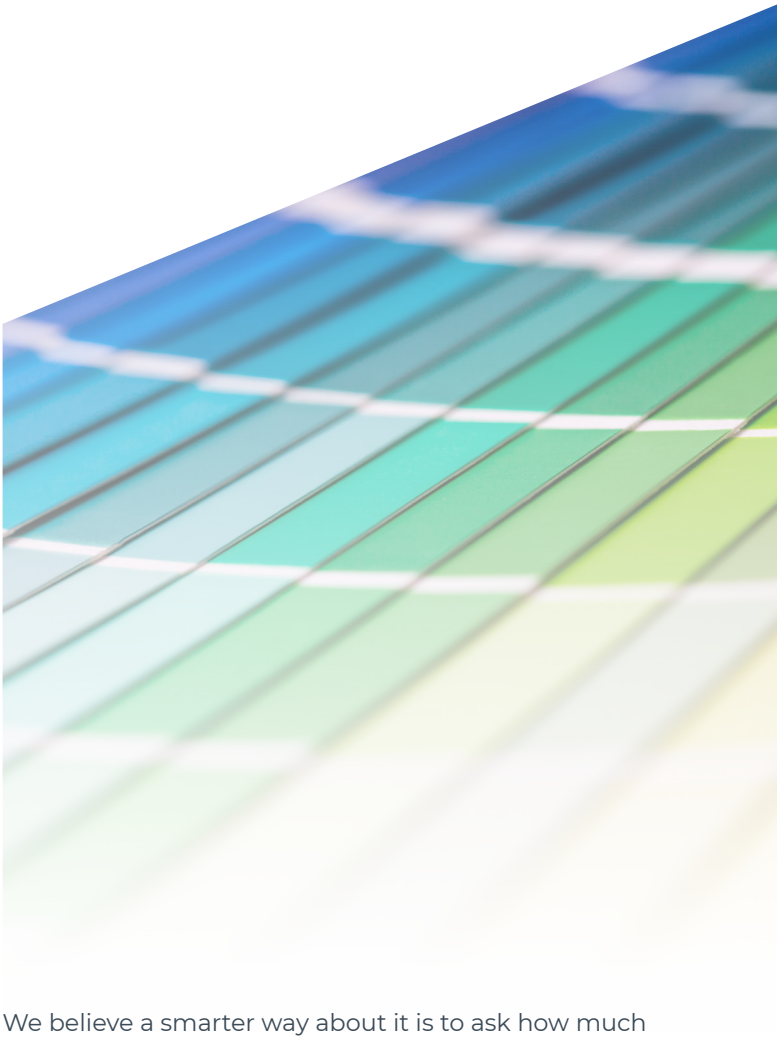


Figure 6. *The results of forecasting when anomalies have been identified to the training model.*

Any event can be a tricky thing, but especially weather. Factors that have relative consistency, like holidays or other special events, tend to be known in advance that they are going to occur. The model can take them into account in a good way and enhance the treatment of anomalies. Factors that are totally unpredictable are probably unforecastable so we just have to deal with them. We can forecast with the potential of something happening by designing the system such that it doesn't completely ignore the anomaly. Done right, we get very accurate results, and that's the point of forecasting.

CREATING AN ENSEMBLE OF MODELS



In time series forecasting with machine learning, there isn't just one algorithm that can do the forecasting. A lot of different algorithms were designed for time series forecasting and there isn't any specific one that is "always the best." They all have a different mathematical way to describe the data and to do this forecasting to fit different functions to the data effectively.

We take an approach with the algorithms that is similar to listening to expert advice. Oftentimes when there are a lot of experts on a subject, the most accurate thing to do is combine the expert advice intelligently into one piece of advice. This is essential what people do when we look at online reviews—we internalize everything that people have written and come up with one composite opinion. We do this with the forecasting task and we call it an ensemble approach.

There are multiple potential algorithms that can do time series forecasting: LSTM, Prophet, Linear Temporal Models, Hybrid RNN, and others. They all take as input time series and output the forecast for that time series. The best approach that we found is to train them all in parallel, independently of each other, and then when we want to forecast tomorrow, we let all of them forecast tomorrow. Each one will output a number, and then we can combine all their outputs into a single forecast that is the voting of what they believe tomorrow will be.

There are a lot of techniques to ensemble the algorithms. The simple approach is to take their average. If one is saying tomorrow's number is going to be 1000, the other one says it will be 800 and the third says 1200, we can take their average and just use that as the forecast of what will be tomorrow.

We believe a smarter way about it is to ask how much trust we have in each individual result, with trust being tied to accuracy. We train the models on historical data and we validate the accuracy of the forecast based on test data. If we know that LSTM for the sales prediction was 98% accurate and Prophet was 90% accurate over that test set, maybe we should trust LSTM a bit more than we do Prophet. If a linear temporal model was 70% accurate, we would trust it less. This enables us to do a weighted average as another approach.

There are many ways to ensemble, but the point is, ensembling the models is the way to get the best, most accurate model. Moreover, we continuously measure the algorithms' accuracy because the fact that an algorithm proved to be good on one set of test data doesn't mean it will continue to be good forever. We continue to recompute their accuracy and reassess how much to trust each model when we are doing the ensemble.

IDENTIFYING AND ACCOUNTING FOR DIFFERENT TIMES SERIES BEHAVIOR

Earlier we discussed the challenge of having a limited amount of historical time series data. More data that goes back further in history can't be invented. If we want to forecast sales day by day, then we can only get 365 data points per year. That's the limit and it's a hard limit. But we ask, is there a way to get around that limit to create a more accurate forecast?

As it happens, there is a potential way to get around that limit. Let's say we want to forecast sales of shoes by an eCommerce business. Not only does this company sell shoes, but it also sells socks and other items, so there is historical sales data for those items as well. And maybe this company has sales data from other companies that are selling shoes. In all, let's say we have five years of day by day sales data for all of those elements. There are probably a lot of shared patterns between the sales of shoes and sales of socks in this company, and sales of shoes of the other companies. If we can collect all that data, although we're limited in the data history, maybe we can learn a lot more about the patterns that occur within the year or within the day or within a month, with less history but more data overall.

The thinking is, if we have multiple time series, it might be possible to train one model that will learn all the patterns, and then we can get by with having less history even though it's not that same time series. For any given time series, we don't have an infinite history, but now we have a lot of time series that may allow us to learn the patterns of all of them together.

The central question becomes whether we can take a lot of them together and train one forecasting model that can be accurate for each one of them individually, while not having infinite history for any one of them.
Will that work? Can a single model be more accurate than individual ones, and how can we know? If not, if we can't just mix everything together, are there some behaviors of time series that we can compute beforehand that will tell us which ones we can mix for a single model and which ones we can't? For example, maybe if they share the same type of seasonal pattern, or they have the same type of pattern throughout the week or throughout the month or throughout the year. Maybe if they do share that then we can group them and train a single model for them together. But if they don't share such similarities, then we can't train them together.

There are many ways to describe the behaviors of time series. Seasonality is one of them. Does the time series have a seasonal pattern, and if so, what is the length of that season? Yearly, hourly, daily, monthly, biweekly, quarterly—anything could be there. How strong is that pattern? Does it tend to have a trend? Is it trending data up or down? Is it very spiky? Is it sparse? Is it stationary? There is actually a very long list of potential behaviors that can be extracted from any time series that describe it mathematically.

After asking so many questions, we conducted extensive research and came to some conclusions—and really there were two main factors that we found to have relevance. If two time series are different in those factors, we cannot train models together with them. The first is seasonal effect. If two time series have very different seasonal patterns, and very different strength for a seasonal pattern in terms of season length, then we cannot train them together. The second is homoscedasticity, which has to do with the variability of the noise level of the data. If the two series' patterns are significantly different, the data cannot be trained together. And the other factors we tested were not as important, meaning we can probably put them together and not worry about that single model being less accurate.

BUSINESS-DRIVEN BIAS — THE HUMAN ELEMENT OF FORECASTING

Throughout this paper we have talked about the need for accuracy—to make the forecast as close as possible to what the actual values will be. There's one exception to the rule of "accuracy is critical" and that is when the human element – CEOs, CFOs, business managers – intentionally wants to introduce a business-driven bias into the forecast. The CFO wants a growth forecast that is slightly less optimistic than reality to ensure the company stays within its budget. The store manager wants his demand forecast to be slightly more optimistic to ensure that inventory is sufficient to cover customer demand. Anodot's system can consider those biases to build in a small percentage of error to deliver the right type of forecast for the business use case.

Consider the case of a company that is doing growth forecasting of revenues for the next year. The CFO is doing the forecasting for budget planning purposes. Now, obviously he wants the forecast to be as accurate as possible, so the measure of how close the test forecast came to what was actually there in the historical data is an important aspect to gain confidence in the forecasting—but there are additional factors to that accuracy measure. We have to consider how this forecast is going to be used, and that's going to influence how we measure the goodness of the forecast.

Suppose tomorrow we give the company a forecast that is optimistic – that is, higher than what will actually happen by 10% – and the organization is going to do all its budget planning based on that forecast. If it's a public company, they will give guidance to the market based on that forecast. And then when the actual number is 10% lower than what was predicted, the CFO is going to be very unhappy. They are short 10% on their budget because they spent too much, and their investors will be very unhappy with them because of the shortfall in revenues. All things considered, the CFO would prefer that we give him a forecast that is less optimistic than what will actually happen, because the impact of giving him an optimistic forecast is more detrimental than if we give him an underestimate of that. In other words, the CFO prefers a bias toward an underestimate rather than us providing an overestimate.

In demand forecasting, oftentimes it's the other way around—we want to slightly overestimate demand to avoid ending up with short supply of something, whether it's inventory in the store, capacity in the network, drivers on the road, money in the ATMs, etc. If the forecast underestimates the demand, the company might end up with short supply, which can be worse than keeping more inventory or having a surplus of supply.

During the training process, models are symmetric in that they give equal weight to an error that goes up and an error that goes down. It will try to minimize the error, get as close as possible to the actual value, but without the business context, it will try to minimize the error with no bias towards one direction or another. However, based on the business use case, we want to incorporate bias into the model when it's being trained. We do this by putting more weight on errors that are higher or lower than the actual value, depending on the intended use of the forecast.

Another typical example is a bias towards forecast accuracy on special times. Consider Black Friday as an example. Many retailers and eCommerce sites gain most of their revenues on that day. The accuracy of the demand forecast for that day is more important than all other days during the year. Biasing the demand forecasting model to be more accurate on that day (and potentially others) is important, as normal accuracy measures average out the error for the entire data and do not give parts of the data special importance.

SUMMARY

Accurate growth or demand forecasts are very important aspects of corporate planning. Machine learning-based forecasting offers the most efficient and effective way to derive business forecasts. There are many principles in designing an autonomous system that can affect the accuracy of a forecast—adding factors that are known to influence the forecast, identifying and accounting for anomalies, ensembling the models, and so on. Anodot has thoughtfully considered these design principles and incorporated them into our product, Autonomous Forecast.

Looking ahead to Part III in this white paper series, we'll cover the system architecture that allows us to "productize" this solution such that it becomes a turnkey SaaS application that any business can use.

In part III of this series, learn more about time series forecasting

Download Part III

For more information, please contact Anodot:

North America
669-600-3120
info.us@anodot.com

International
+972-9-7718707
info@anodot.com

Anodot is a turnkey AI analytics platform whose anomaly detection and forecasting is helping market leaders such as Waze, Microsoft and Wix to seize opportunities and avoid revenue loss.

Anodot Autonomous Detection drives scalable, adaptive business and operational monitoring. Patented machine learning algorithms weed out superficial outliers and alert storms to reveal critical anomalies and correlate them to similar anomalies and events. Leading fintech, eCommerce, telco, gaming, adtech and digital businesses are using Anodot to cut remediation time by 50-80 percent.

Anodot Autonomous Forecast continuously forecasts growth and demand – no data science experience needed. Algorithms are independently selected, trained and tuned to produce highly accurate forecasts. By identifying your data trends in real time, Anodot enables companies to quickly anticipate changing conditions and avoid unnecessary costs.

Learn more at **www.anodot.com**