Secure and Automated Enterprise Revenue Forecasting

Jocelyn Barker
Microsoft Corp.,
CA, USA.
jocelyn.barker@microsoft.com

Amita Gajewar
Microsoft Corp.,
CA, USA.
amitag@microsoft.com

Konstantin Golyaev
Microsoft Corp.,
WA, USA.
konstantin.golyaev@microsoft.com

Gagan Bansal *
Google Inc.,
CA, USA.
gagan.bansal@gmail.com

Matt Conners
Microsoft Corp.,
WA, USA.
matt.conners@microsoft.com

Abstract

Revenue forecasting is required by most enterprises for strategic business planning and for providing expected future results to investors. However, revenue forecasting processes in most companies are time-consuming and error-prone as they are performed manually by hundreds of financial analysts. In this paper, we present a novel machine learning based revenue forecasting solution that we developed to forecast 100% of Microsoft's revenue (around $85 Billion in 2016), and is now deployed into production as an end-to-end automated and secure pipeline in Azure. Our solution combines historical trend and seasonal patterns with additional information, e.g., sales pipeline data, within a unified modeling framework. In this paper, we describe our framework including the features, method for hyperparameters tuning of ML models using time series cross-validation, and generation of prediction intervals. We also describe how we architected an end-to-end secure and automated revenue forecasting solution on Azure using Cortana Intelligence Suite. Over consecutive quarters, our machine learning models have continuously produced forecasts with an average accuracy of 98-99 percent for various divisions within Microsoft's Finance organization. As a result, our models have been widely adopted by them and are now an integral part of Microsoft's most important forecasting processes, from providing Wall Street guidance to managing global sales performance.

1 Introduction

Revenue forecasting is a key function of any finance organization in an enterprise. Most companies continuously need to plan future spending, such as marketing budgets or employee hiring. Accurate revenue forecasts allow companies to estimate their future spending ability thereby significantly improving their strategic business planning. Accurate revenue forecasts also allow companies to provide a precise view of their financials to shareholders and convince investors to fund necessary investments. Despite the importance of the forecasting process, finance organizations in large enterprises face significant challenges today as the amount and availability of relevant data has grown exponentially. Most organizations currently manage their forecasts using a variety of off-the-shelf packages that rely on basic statistical analysis. In addition to being error prone, the forecasts are biased by human judgment, are time-consuming to create, and cannot easily incorporate new data streams as they become available nor easily adapt to changing business environments.

In this paper, we present our secure and automated revenue forecasting solution developed for Microsoft's Finance organization. Our solution generates automated forecasts for 100% of Microsoft revenue across all geographies and products in a few hours, and has implemented security controls needed to protect the high business impact (HBI) data such as revenue. Owing to the consistent accuracy over the past few quarters, the machine learning forecasts have been widely adopted by various Finance teams within Microsoft.

As an overview, we first review classical time series models in Section 2. We then propose our machine learning based modeling framework in Section 3 which extends classical time series models into a generic regression framework and allows us to easily combine external sources of data with historical information. We also include details on the methods we used for tuning of hyperparameters of machine learning models, selection of best model, and computation of calibrated confidence intervals. We then provide detail architecture of our automated solution using services from the Cortana Intelligence Suite (e.g., Azure Data Factory, Azure Machine Learning, etc.) in Section 4. We built this solution on Azure since it is a scalable and cost-efficient platform for building and operationalizing data-intensive advanced analytics workloads. However, since it is a public cloud with many multi-tenant services, strict controls are required to ensure the security of sensitive data like revenue. Hence, we mention the steps taken to meet the corresponding stringent security standards recommended by our enterprise IT team. The key contributions of our paper are twofold.

1. We propose a machine learning based modeling framework for time series data that uniquely combines external data sources like macroeconomic factors, sales pipeline data, social data, product launches with the historical revenue data to generate accurate long range (multi-quarter or multi-year) forecasts. This gives more visibility to business leaders about their business allowing them to manage

*Work was done while author was at Microsoft.

it better. We also provide a method for computing prediction intervals which provides statistical estimates on the confidence of the forecast to the financial executives allowing them to better manage the risk.

2. We propose an architecture for running advanced analytics workloads over highly sensitive enterprise data like revenue on a public cloud such as Azure. This architecture allows us to build a secure and automated revenue forecasting solution that can run on desired frequency, such as daily or weekly, something which is not possible with manual forecasts. This allows finance executives to track their likelihood to achieve the quarterly goals within the quarter, thereby making the forecasts more actionable. Further, this also reduces the time to generate the forecasts to hours as compared to weeks if the forecasts were generated manually.

2 Related Work

Classical time series forecasting models have been well studied by researchers in the statistics community for the past few decades (Winters 1960; Holt 2004; Box et al. 2015). Many of these methods rely on decomposing the time series into three primary components:

1. **Trend**: The long term increase or decrease in the data after higher frequency patterns and noise are removed.
2. **Seasonality**: The repeating pattern of a fixed interval that is seen in the data.
3. **Noise**: Irregular component in the data.

Some of the popular classical time series forecasting models include Auto-Regressive Integrated Moving Average (ARIMA) and Exponential Time Smoothing (ETS). ARIMA model has three main components, the auto-regressive part which represents how much past values of a series directly influence its current value; the integrated part which represents the order of differencing required to make the time series stationary over time; and the moving average part which represents how much inter-period dependence there is in the noise component of the time series. ETS is another popular time series model which has many specifications depending on how one chooses to model the trend and seasonal components, and whether errors are additive or multiplicative. These classical time series models work well on capturing historical trends and seasonality. However, these models have trouble incorporating external information, which can be key to accurate forecasts. For example, it is known that temperature has a strong effect on electricity demand (Weron 2007) as with a rise in temperature more people use energy intensive appliances such as air conditioning units. Day of week can also have an effect as on weekdays there is higher demand in industrial areas and lower demand in residential areas compared to the demand on weekends. These factors are key in producing accurate energy forecasts to meet energy demand, and therefore it is essential that they are incorporated into the models.

Leveraging this information in classical time series models is not trivial. Incorporating external data into ETS models is particularly difficult. While some researchers have been able to do it (Athanasopoulos and Hyndman 2008; Osman, King, and others 2015), it has been at the cost of forecast-ability and interpret-ability of the models. ARIMA-based models can accommodate features (Young and Whitehead 1977), but most of the existing implementations do not include any regularization mechanisms, and therefore they have a strong tendency to overfit on high dimensional data. For this reason, regression frameworks are appealing as they allow to easily incorporate external factors. Regression models like support vector machines (Ruping and Morik 2003; Kim 2003) and neural networks (Dorffner 1996) have been applied to time series forecasting in the past. Such models have also been applied for business forecasting problems like supply chain demand forecasting (Carbonneau, Laframboise, and Vahidov 2008), energy load forecasting (Chen, Chang, and others 2004), and to a limited extent for revenue forecasting (Lin et al. 2013).

Fitting regression models on time series data has additional concerns which are not relevant to most of the machine learning applications, e.g., data are not independent. This causes problems in selecting model hyperparameters, e.g., regularization parameters α and λ for an elastic net model (Zou and Hastie 2005). Standard cross-validation where samples are in some way randomized and divided into groups for training and testing relies on the assumption that data are independent and identically distributed (Opsomer, Wang, and Yang 2001). This results in overfitting on dependent data (Hart and Wehrly 1986). As a result, many new methods for cross-validation in time series have been developed and evaluated to reduce overfitting (Arlot, Celisse, and others 2010; Bergmeir and Benítez 2012; Bergmeir et al. 2015).

In addition to the accuracy, it is also important to consider how these machine learning models can be deployed into the production environment. With an availability of cloud computing (Armbrust et al. 2010), many enterprises have now started to migrate their existing applications and build new applications on the cloud. Cloud allows these applications to be cost-efficient and elastically scale storage and processing without requiring significant upfront hardware investments. However, with public clouds like Azure and Amazon Web Services, enterprises need to carefully architect their solutions in order to avoid the risk of leaking any sensitive data (Mather, Kumaraswamy, and Latif 2009). In this paper, we describe the security controls that can be followed to build a secure enterprise application on the public cloud.

3 Revenue Forecasting Models

We now describe a modeling framework that extends time series forecasting to a generic regression problem.

3.1 Feature Extraction for Time Series Data

The idea at the core of feature extraction for forecasting is to create predictive features for forecast values using only information that would have been known at the time of forecast. Different data will be available to forecast the same data point from different horizons, or lengths of time into the past from when the forecast was made. This means

that in order to train a model to forecast data at horizons $h \in 1, \ldots, H$, each time point in the training set will have features extracted H times; each time using only data known h time points prior to the time being forecast. For example, if we desire to forecast the value of a series in 2016-Q4 at horizon 1, it would be fair to use all data known up to 2016-Q3. However if we wanted to make the 2016-Q4 forecast at horizon 2, it would only be fair to use information known up to 2016-Q2. Including two sets of features for the 2016-Q4 data point, one at horizon 1 and one at horizon 2, allows the models to make future forecasts at both horizons.

Features Derived from Historical Values As discussed in the Section 2, traditional statistical methods for time series forecasting strongly leverage patterns in the historical values for a time series when creating a model. Our method builds on this by creating features both from the historical data, as well as from the forecasts made by other time series forecasting methods (Gajewar and Bansal 2016). The three main classes of features derived from historical data are:

1. **Core Data**: Features characterizing core elements of the forecast (i.e. time, seasonality, horizon).
2. **Forecast Features**: Forecasts derived from the classical methods.
3. **Lag Features**: Previous data points from the time series.

The Core Data features are simple features and provide information about the time point that is being forecast, rather than the values of the data. Forecast features leverage the previously developed statistical models to inform the machine learning forecasts. These features are the forecasts computed using the classical models, e.g. ARIMA or ETS forecasts, made for the series at time t and horizon h. Parameters for the classical models are selected using the methodology described in (Hyndman, Khandakar, and others 2007). Lag features are previous values of the series starting at the time of forecast (time $t - h$) and moving backwards. This is done to both incorporate the most recent information possible into the forecast, and avoid using data which would be unknown at the time of forecast. These features allow the model to find patterns in the data which do not conform to the structure imposed by the classical models. Finally, the response variable is simply the value of the series (y_t) at time t. Taken together, these features allow the models to build on the success of previous models while also potentially discover patterns the previous models do not allow.

Features Derived from External Data One of the advantages of transforming a forecasting problem into a regression problem is the ease of adding external data. Such data can become features in the model and existing solutions can be employed to avoid over-fitting. Much as with the features derived from historical values of the series (Section 3.1), care must be taken to avoid incorporating information that would be unknown at the time of forecast into the features set. At the same time, it may be possible to know a future value of the series, e.g. in product retail forecasting, the sale price of the product next month. When data are known in advance it is possible to incorporate lags relative to the time of forecast, and even create lags representing future values. This can be advantageous as it provides information temporally closer to the time of forecast, and therefore, more likely to be informative to the forecast.

3.2 Modeling with Time Series Cross-Validation

Our modeling framework uses a form of time series cross-validation based on that described in (Hyndman and Athanasopoulos 2014) to avoid overfitting in time series data. This method, also known as "rolling forecasting origin" validation, tries to replicate how forecasting would be performed in practice. The number of validation folds K is chosen for the time series of length N. In each fold, the training and validation sets are defined such that:

$$T_k = \{x_t | t < N - K + k\}$$

$$V_k = \{x_t | N - K + k \leq t < N - K + k + H\}$$

Where T_k is the training set for fold $k \in 1, \ldots, K$, V_k is the validation set for fold k, x_t is the data corresponding to time t, and H is the maximum forecast horizon desired. Since our data have H instances for each time point t, a slight modification of this method was required. In the modified form of cross validation, we define our training and validation sets such that:

$$T_k = \{x_{t,h} | t < N - K + k; h \in 1, \ldots, H\}$$

$$V_k = \{x_{t,h} | t = N - K + k + h - 1 \text{ for } h \in 1, \ldots, H\}$$

Where $x_{t,h}$ are the data corresponding to time point t as calculated at horizon $h \in H$.

A concrete example of two folds of cross-validation (Figure 1) demonstrates how the process works. In the first fold of cross-validation the training set includes data from 2010-Q1 to 2010-Q4 and the test set would be the data from 2011-Q1 at horizon 1, 2011-Q2 at horizon 2, 2011-Q3 at horizon 3, and 2011-Q4 at horizon 4. The next fold of cross-validation would move one step forward in time and the training set would contain data from 2010-Q1 to 2011-Q1 while the test set contains 2011-Q2 at horizon 1, 2011-Q3 at horizon 2, etc.

This cross-validation framework is model agnostic, so any regression model can employ it for forecasting purposes. In order to maintain this flexibility, the framework allows a user to specify multiple models. For the results presented in this paper, we used an Elastic Net model, a Random Forest model, a K-Nearest Neighbors (kNN) model, and a Support Vector Machine (SVM) with a radial kernel. All of these models have mechanisms to avoid overfitting in high dimensionality data, e.g., regularization in Elastic Net and SVM. However, when multiple models can be implemented, the problem of model selection arises. Section 3.3 discusses how our proposed framework addresses this problem.

3.3 Model Selection Methodology

Our method allows for easy back testing by leveraging the time series cross-validation used to train the models (Section 3.2). For each fold of cross-validation k, we use the data from cross-validation folds $1, \ldots, (k - 1)$ to select hyperparameters for the model with the lowest root mean square error (RMSE). We only use the errors for data whose

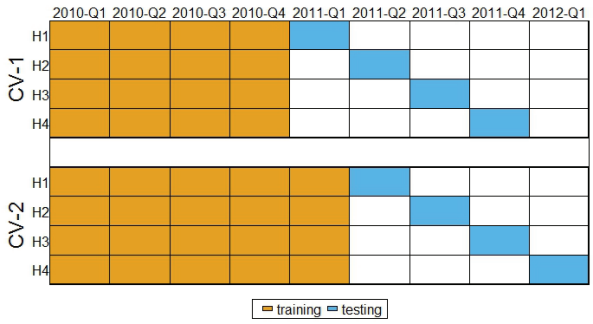

Figure 1: An illustration of how data would be allocated into training and testing data for two folds of cross-validation. In the diagram, each of the two groups of boxes represent a fold of cross-validation. Within each cross-validation fold each box represents one set of features to be used to make a forecast, with the box located in the same location in each fold representing the same set of data. The columns represent the time point of the forecasts, and the rows represent the horizon of the forecast.

error would have been known at time of forecast (where $t \leq N - K + k$). We then collect the error for the held out testing data in fold k using those hyperparameters selected using the error of the previous folds.

Once these back testing errors are collected, it is possible to compare model errors on historical data. A "best" model can be selected using a metric of choice. We have empirically found best results using a weighted RMSE where the weight of the error decays exponentially with an increase in time since present forecast. For a forecast made at time t

$$wRMSE = \sqrt{\sum_{v=N-K+1}^{t} \sum_{h=1}^{H} \frac{e_{v,h}^2}{w_v}}$$

where

$$w_v \propto \frac{1}{2^{t-v}} \text{and} \sum w_v = 1,$$

where $e_{v,h}$ is the validation error at index v and horizon h and w_v is the weight for the validation data at index v.

3.4 Prediction Interval Generation

Most common methods for prediction interval quantification rely on the hypothesis that the actual value of a forecast is the sum of the modeled value and a normally distributed error term (the noise in the system) (Chatfield 1993). This error distribution is generally estimated using the in-sample errors of the model (Hyndman and Athanasopoulos 2014). However, it is a known phenomenon that prediction intervals for forecasting using this assumption tend to be too narrow. For example, one study found that for ETS models, the the estimated 95% prediction intervals only in fact covered $71-87\%$ of the actual forecasts (Hyndman et al. 2002). This is likely because this assumption only takes into account error due to noise in the system, and not other sources of error such as error in model parameter estimates and changes in the data from historical norms (Chatfield 1993).

In the traditional method the prediction interval is estimated as a Gaussian distribution with a mean of zero and a standard deviation equal to the standard deviation of the residuals around the assumed mean of zero. Our method is very similar to the first except in one key aspect; the errors used to approximate the Gaussian distribution are out-of-sample errors. In most methods, this would be computationally expensive, and this is likely the reason in-sample errors are used. In our method, we can leverage the back testing errors described in Section 3.3 with almost no additional cost. This avoids reduction of the prediction interval size due to over fitting on the in-sample data, and it helps to account for some of the effect of uncertainty in the model hyperparameters. Additionally, it allows for the computation of prediction intervals on models which would not support them with in-sample errors, e.g. a kNN forecast with $k = 1$.

One thing to note is that unlike in classical time series methods, the prediction interval computed here may not grow wider at longer horizons. Classical methods use a recursive strategy for long horizon forecasts where forecasts for earlier time points are used as inputs for later ones. As a result uncertainty in the model compounds. Here we use a direct strategy where longer horizon forecasts are not dependent on shorter horizon ones. This means that uncertainty may or may not grow with time depending on observed patterns in the data.

4 Cloud-based automated and secure forecasting solution

We have developed an automated solution to generate forecasts for 100% of Microsoft's revenue on Azure using Cortana Intelligence Suite (CIS) services like Azure Data Factory (ADF), Azure Machine Learning (AML), Azure database (Azure DB), PowerBI (an interactive data visualization tool from Microsoft) and ExpressRoute. We provide the end-to-end solution architecture in Subsection 4.1. Further, since most enterprises would consider their revenue data as highly sensitive, they would require strict security controls for storing and processing these data using multi-tenant services on a public cloud. Hence we needed to implement desired security controls as part of the solution architecture. We illustrate the security controls for each of the components in our architecture in Subsection 4.2. Although we provide a reference architecture on Azure, the security controls can be generalized to other public clouds.

4.1 Automated Workflow on Azure

The end-to-end revenue forecasting workflow as depicted in Figure 2, can be divided into four main activities:
(1) Moving historical revenue data from the SQL server within Microsoft's corporate network to Azure SQL database. (2) Generating forecasts by invoking AML-based web-service. (3) Storing the forecasts back in the Azure DB. (4) Updating PowerBI dashboard to reflect the most up-to-date actual and forecasted revenue along with historical fore-

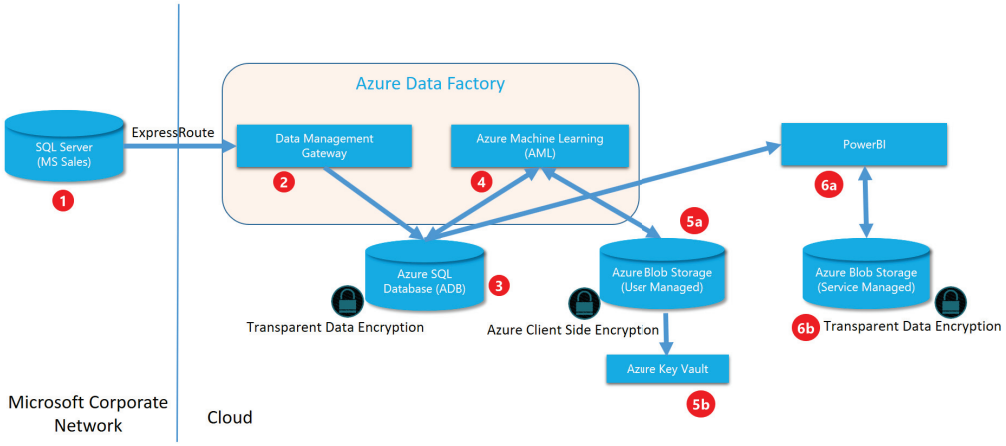

Figure 2: Revenue Forecasting Solution Architecture.

cast accuracies.The following services from the Cortana Intelligence Suite were used to build this workflow:

1. **Azure Data Factory:** ADF is a managed service that allows for orchestration of advanced analytics workflows on Azure. We used ADF to automate the activities described above and configured it to run at the required cadence of the respective forecasts (e.g. weekly, monthly, etc.). The data management gateway (component 2 in Figure 2) was also required in this ADF to migrate data from Microsoft's corporate network to Azure.
2. **Azure SQL Database:** We used Azure DB (component (3)) to store the actual historical data along with machine learning based forecasts. Specifically, we used temporal tables to store the data. Using temporal tables allowed us to automatically archive historical data and only retain the most recent data in the main table.
3. **Azure Machine Learning:** The training experiments corresponding to the modeling workflow described in Section 3 were done in AML. The scoring experiments used to generate the forecasts were published as REST APIs (web services). These web services were integrated in ADF to enable the end-to-end workflow.
4. **PowerBI:** We created a PowerBI dashboard to visualize the historical and forecasted data, confidence intervals and historical forecast accuracy metrics. This dashboard was automatically updated when either the actual data or the forecasts in Azure DB were refreshed.

4.2 Security Controls

In this subsection, we provide details on the security controls used in our end-to-end automated workflow for revenue forecasting on Azure. These security controls can be summarized as below

1. **Encryption**: Data in transit should be over a secure and encrypted channel (SSL/TLS etc.). Data at rest should be encrypted. Any keys used for encryption should be securely managed and regularly rotated to defend against potential future attacks.
2. **Logging and Monitoring**: User activity logs should be continuously captured and monitored for all cloud services.
3. **Access control**: Multi-factor authentication should be used for logging into production solution. All users should have minimum required access rights to data sources and cloud resources. Strong password generators should be used to generate passwords and no password should be stored in clear text anywhere. Further, any connection strings containing sensitive passwords should be encrypted.
4. **Threat Modeling and Penetration Testing**: A detailed threat model including the solution architecture, attack surface and data flows should be periodically updated and reviewed. Penetration testing should be done periodically or when significant architectural changes are performed.
5. **Security Updates**: Security updates should be applied immediately after they are available for any of the components in the solution.
6. **Incident Response Plan**: A detailed incident response plan including methods for detecting, response, deterrence, post-mortem of security events along with roles and responsibilities of teams should be prepared.

We now describe the security controls used for each of the components in our pipeline below.

1. **On-premise SQL Server:** Microsoft's historical revenue data resides inside its corporate firewall in an on-premise SQL Server as indicated by component (1) in the Figure 2. We created a unique service account to access this SQL Server from our automated solution. This service account was not shared with any other application and its password was generated by a strong password generator to prevent misuse of its credentials.
2. **Azure Data Factory:** All connection strings configured in ADF were encrypted and Data Management Gateway was installed on a Microsoft domain joined virtual machine (VM) on Azure. ExpressRoute was used to create a

private connection between this VM and Microsoft's corporate network. Further, this VM had strict access controls and continuous monitoring to track any misuse.

3. **Azure SQL Database:** We enabled Transparent Data Encryption (TDE) for Azure DB to ensure that all of the data at rest is encrypted. TDE allowed the database service to transparently use secret keys to encrypt any data while storing on the disk and decrypt while retrieving from disk. Further, we also enabled logging for Azure DB and setup regular monitoring to detect any malicious user action.
4. **Azure Machine Learning:** During the execution of an experiment, AML stores intermediate outputs in Azure Blob storage. To ensure security of these intermediate data outputs, we configured AML to use encryption keys to encrypt and decrypt the data at rest stored in Blob storage. These keys were generated by us and uploaded to Azure Key Vault, which is a service that helps safeguard cryptographic keys and secrets used by cloud applications and services.We also enabled a feature in AML which prevented any user of our AML experiments to share these experiments with a user outside the company. Finally, we enabled continuous collection and monitoring of user audit logs.
5. **PowerBI:** The Power BI tenant used Azure Active Directory (AAD) for authentication. Further, the Power BI service managed an encrypted Azure Blob Storage account where datasets like the uploaded metadata files (e.g. pbix) were securely stored.

5 Results

Forecasts for 100% of Microsoft's revenue were generated using the previously described framework and shared with Microsoft's Finance Team for the last four quarters. Forecasts were created for about 30 individual products which make up the three segments which are reported to Wall Street. These segments are 1) Intelligent Cloud, which includes a portfolio of server products, Azure and Enterprise Services, 2) Productivity and Business Processes, which is comprised of commercial and consumer Office, and Dynamics, and 3) More Personal Computing, which includes Windows, Devices, and Search. These time series represent a wide variety of properties ranging from slowly evolving contract-based series to highly dynamic consumer products.

5.1 Forecast Accuracy

In order to compare the forecasts at higher granularity, the mean absolute scaled error (MASE) statistic (Hyndman and Koehler 2006) was computed for each product. The MASE is defined as:

$$MASE = \frac{\frac{1}{H}\sum_{t=N+1}^{N+H}|e_t|}{\frac{1}{N-s}\sum_{t=s+1}^{N}|y_t - y_{t-s}|},$$

where s is the seasonality of the data. The denominator of the MASE computes the in-sample mean absolute error (MAE) for the seasonal naive model, which is generally considered to be a good reference forecast. The numerator computes the out-of-sample MAE for the forecasting method.

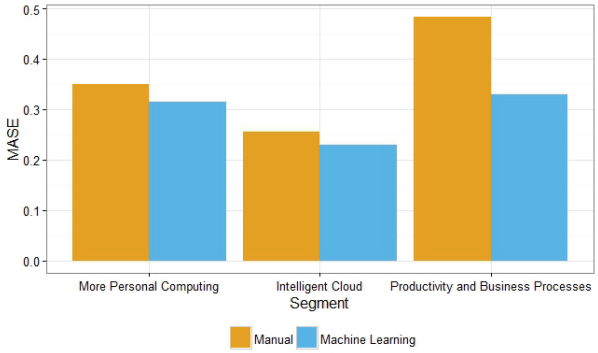

Figure 3: Barplot comparing the MASE value for ML and manual forecasts in all three External Segments computed over the last four quarters.

MASE is the preferred metric for forecast accuracy as it is independent of data scale and helps to normalize across time series forecasting difficulty, making comparing forecast results easier. It has an additional benefit in our data as it makes it possible to quantitatively report errors without disclosing highly sensitive data.

Approximately 70% of the time the machine learning forecasts were more accurate at the segment level than the manual forecasts, which are those computed by traditional methods. Additionally the MASE showed an overall improvement in all three segment level forecasts when computed across the four quarters (Figure 3).

Figure 4 compares the error distributions of the ML and manual forecast MASEs at the product level using a density plots for the errors. ML errors for More Personal Computing and Intelligent Cloud both peak at a lower value and skew more to the left, indicating more accurate forecasts. In particular, Intelligent Cloud shows a much greater skew towards lower forecast errors. The MASE density for Productivity and Business Processes peaks at a similar value for both manual and machine learning forecasts, however, the machine learning forecasts have a much higher density at the lower values, indicating higher accuracy.

5.2 Prediction Interval Accuracy

In order to fully leverage the power of machine learning forecasts, it is important to have prediction intervals that the Finance team can use to quantify the risk in the forecasts. However none of the machine learning methods used for forecasting have prediction interval methods available as an intrinsic part of the model. Our pipeline creates prediction intervals using back testing of the models (Section 3.4) to create an error distribution based on out-of-sample errors.

Prediction intervals were evaluated by computing the quantile of the predicted error distribution for the actual revenue value for each of the forecasts created in each quarter. If the prediction intervals accurately reflect the actual error distribution, we would expect the distribution of quantile errors to be uniform on the interval of $[0, 1]$. Put more simply, we would expect 10% of the errors to be in the bottom 10%

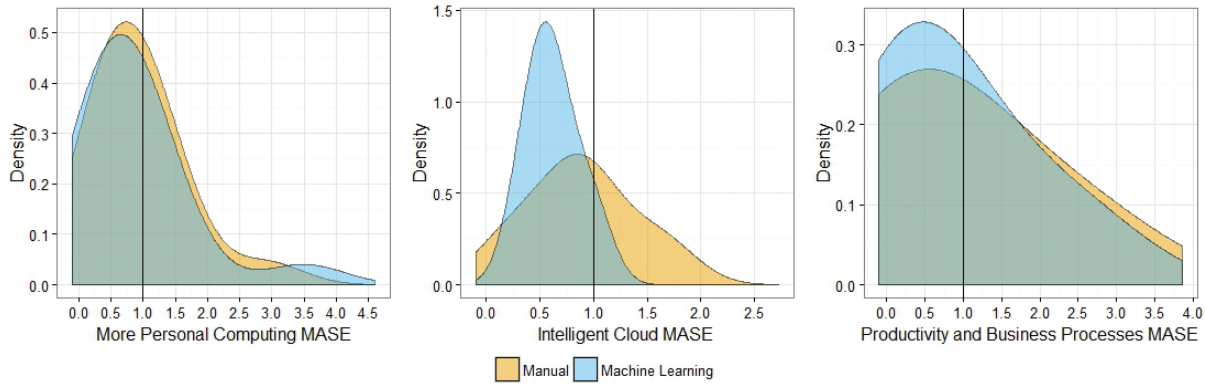

Figure 4: Density plots for the product breakdown MASEs for each of the External Segments. For both More Personal Computing and Intelligent Cloud the ML forecasts are skewed more tightly to the low end of the graph, indicating lower errors.

of the distribution, 20% in the bottom 20%, etc. The distribution of error quantiles can easily be visualized by plotting a CDF of forecast error quantiles similar in concept to a Q-Q plot. The ideal distribution would be a line through the origin with a slope of 1. Since it is important that accurate prediction intervals are created for all machine learning methods evaluated, error distributions were evaluated for all forecasts made by each method, not only for the "best" forecasts (Figure 5). While there was some deviation from the expected distribution in the tails of the distribution, the errors closely followed the expected distribution, indicating the pipeline's prediction intervals characterize the error distribution well.

Together these data show our framework's ability to leverage both regression modeling and classical methods to produce accurate forecasts and calibrated prediction intervals.

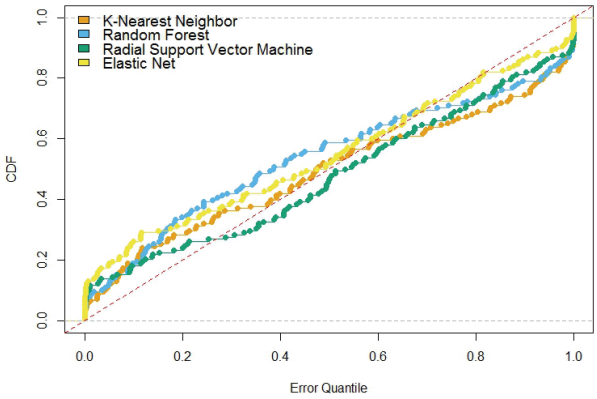

Figure 5: CDF plot for the error quantiles of the actual revenue for the last 2 quarters based on the forecast prediction interval. Prediction intervals were not produced for the earlier forecasts. The red dashed line represents the ideal uniform distribution of error quantiles. Data for all four machine learning methods evaluated fall very near this line.

6 Discussion

Our revenue forecasting solution described in this paper has evolved over the period of two years wherein we ran several experiments to fine tune the models and cater the requirements of Finance team, worked with IT team to ensure security of the HBI data in the cloud, developed automated pipelines, and created dashboards. The core team consisted of four data scientists and one program manager, and had help from IT team for production support.

Our fully automated pipeline has been used to securely provide revenue forecasts for Microsoft's Finance team for the past four quarters. The proven track record of high accuracy combined with the fast turnaround time of the forecasts has led to a wide spread adoption throughout the Finance organization. Unbiased, automated forecasts in an easily digestible format (i.e. Power BI dashboard) provided the Finance team a powerful tool to improve their own forecast accuracy. In particular, the prediction intervals have proven useful to the team as these intervals allow them to estimate the risk in these forecasts. Additionally, since the framework is generic, with minor adaption it can be applied to other companies or even other types of forecasting problems.

Our work has some limitations. While the framework has been tested on a wide variety of Microsoft's financial data, these time series extend back at most 10 years. It is possible there may be some computational challenges to overcome for longer time series. These problems can likely be solved by reducing the search space for the univariate models used as features and limiting the machine learning methods to those that are computationally more efficient. Additionally, when the data are very short (i.e. less than 3 seasons), there may not be enough historical data to generate time series features and validate models. However, many of the framework elements can still be used on such short time series. For example, the methods for creating univariate forecast features (Section 3.1) can be used to create historical forecasts, which can then be used by the methods for model selection (Section 3.3) and prediction interval generation (Section 3.2) to create forecasts on these short data.

The success of this solution has generated high demand for expanded pipeline features. These forecasts can provide a starting point for the manual forecasts to build on, making

the manual process less labor intensive. One work already in progress is the creation of updateable in-quarter forecasts, which will allow the Finance team to evaluate whether they are likely to hit their targets. Since manual forecasts are labor intensive, high quality in-quarter forecasts are not available, but automated forecasts can easily fill this gap.

7 Conclusion

The secure, automated revenue forecasting pipeline described in this paper shows a strong track record of high accuracy and proven value in a high business impact application. This work creates a general platform which can be deployed for many forecasting applications.

8 Acknowledgments

The authors would like to thank the Microsoft's Finance team for their constant partnership with us and business insights to improve the model performance. We are also grateful to the Microsoft's IT team for their help to build a secure pipeline on Azure and for validating our pipelines through thorough security reviews and penetration tests.

References

Arlot, S.; Celisse, A.; et al. 2010. A survey of cross-validation procedures for model selection. *Statistics surveys* 4:40–79.

Armbrust, M.; Fox, A.; Griffith, R.; Joseph, A. D.; Katz, R.; Konwinski, A.; Lee, G.; Patterson, D.; Rabkin, A.; Stoica, I.; et al. 2010. A view of cloud computing. *Communications of the ACM* 53(4):50–58.

Athanasopoulos, G., and Hyndman, R. J. 2008. Modelling and forecasting australian domestic tourism. *Tourism Management* 29(1):19–31.

Bergmeir, C., and Benítez, J. M. 2012. On the use of cross-validation for time series predictor evaluation. *Information Sciences* 191:192–213.

Bergmeir, C.; Hyndman, R. J.; Koo, B.; et al. 2015. A note on the validity of cross-validation for evaluating time series prediction. *Monash University, Department of Econometrics and Business Statistics, Tech. Rep.*

Box, G. E.; Jenkins, G. M.; Reinsel, G. C.; and Ljung, G. M. 2015. *Time series analysis: forecasting and control*. John Wiley & Sons.

Carbonneau, R.; Laframboise, K.; and Vahidov, R. 2008. Application of machine learning techniques for supply chain demand forecasting. *European Journal of Operational Research* 184(3):1140–1154.

Chatfield, C. 1993. Calculating interval forecasts. *Journal of Business & Economic Statistics* 11(2):121–135.

Chen, B.-J.; Chang, M.-W.; et al. 2004. Load forecasting using support vector machines: A study on eunite competition 2001. *IEEE transactions on power systems* 19(4):1821–1830.

Dorffner, G. 1996. Neural networks for time series processing. In *Neural network world*. Citeseer.

Gajewar, A., and Bansal, G. 2016. Revenue forecasting for enterprise products. *International Symposium of Forecasting*. ISSN 1997-4124.

Hart, J. D., and Wehrly, T. E. 1986. Kernel regression estimation using repeated measurements data. *Journal of the American Statistical Association* 81(396):1080–1088.

Holt, C. C. 2004. Forecasting seasonals and trends by exponentially weighted moving averages. *International journal of forecasting* 20(1):5–10.

Hyndman, R. J., and Athanasopoulos, G. 2014. *Forecasting: principles and practice*. OTexts.

Hyndman, R. J., and Koehler, A. B. 2006. Another look at measures of forecast accuracy. *International journal of forecasting* 22(4):679–688.

Hyndman, R. J.; Koehler, A. B.; Snyder, R. D.; and Grose, S. 2002. A state space framework for automatic forecasting using exponential smoothing methods. *International Journal of Forecasting* 18(3):439–454.

Hyndman, R. J.; Khandakar, Y.; et al. 2007. Automatic time series for forecasting: the forecast package for r. Technical report, Monash University, Department of Econometrics and Business Statistics.

Kim, K.-j. 2003. Financial time series forecasting using support vector machines. *Neurocomputing* 55(1):307–319.

Lin, K.-P.; Pai, P.-F.; Lu, Y.-M.; and Chang, P.-T. 2013. Revenue forecasting using a least-squares support vector regression model in a fuzzy environment. *Information Sciences* 220:196–209.

Mather, T.; Kumaraswamy, S.; and Latif, S. 2009. *Cloud security and privacy: an enterprise perspective on risks and compliance*. " O'Reilly Media, Inc.".

Opsomer, J.; Wang, Y.; and Yang, Y. 2001. Nonparametric regression with correlated errors. *Statistical Science* 134–153.

Osman, A. F.; King, M. L.; et al. 2015. A new approach to forecasting based on exponential smoothing with independent regressors. Technical report, Monash University, Department of Econometrics and Business Statistics.

Ruping, S., and Morik, K. 2003. Support vector machines and learning about time. In *Acoustics, Speech, and Signal Processing, 2003. Proceedings.(ICASSP'03). 2003 IEEE International Conference on*, volume 4, IV–864. IEEE.

Weron, R. 2007. *Modeling and forecasting electricity loads and prices: A statistical approach*, volume 403. John Wiley & Sons.

Winters, P. R. 1960. Forecasting sales by exponentially weighted moving averages. *Management science* 6(3):324–342.

Young, P., and Whitehead, P. 1977. A recursive approach to time-series analysis for multi-variable systems. *International Journal of Control* 25(3):457–482.

Zou, H., and Hastie, T. 2005. Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 67(2):301–320.