Forecasting Strategies

Nicholas Bradley

# Forecasting Strategies

The goals of this analysis are:

* Train and test approaches for a forceasting model
* Performance evaluation methods and error measurement matrices
* Benchmark methods
* Quantifying forecast uncertainty with confidence intervals and simulation

## The forecasting workflow

Time series forecasting follows the same workflow as most predictive analysis and includes the following stages:

* Data preparation: Here, the data is prepared for training and testing process. This involves splitting the data into training and testing partitions, creating new variables (when applicable) and applying a transformation if needed (log transofrmation, scaling etc)
* Train the model: Use the training partition to train a statistical model. The objective of this step is to utilise the training set to train, tune and estimate the model coefficients that minimize the selected error criteria. The fitted values and the model estimation of the training partition observations will be used later on to evaluate the overall performance of the model
* Test the model: Utilise the trained model to forecast the corresponding observations of the model with a new dataset (that the model did not see in the training process)
* Model evaluation: After the model has been trained and tested, the model can now be evaluated in terms of overall performance on both the training and test data.

Based on model evaluation, if the model meets a certain threshold, then it the model can be used on the full series to generate the final forecast or select new training parameters/different model and repeat the training process.

It is important to remember that time series forecasting has its own unique characteristics, which distinguish it from other predictive fields:

* The training and testing partitions must be in chronological order, as opposed to random sampling.
* Typically, once the model has been trained and tested using the training and testing data, it is then necessary to retrain the model on all the series data, which does cause a problem with overfitting, so steps need to be taken to address this.

## Training approaches

The quality of the model’s training will have a direct impact on the forecast output. The main goals of this process is as follows:

* Formalise the relationship of the series with other factors such as seasonal and trend patterns, correlation with past lags and external variables in a predictive manner.
* Tune the model parameters (when applicable)
* The model is scalable on new data, in other words, it avoids overfitting.

## Training with single training and testing partitions

It is important to ensure the length of the test partition is upto 30% of the total length of data in order to have enough observations for the test data. For example, if we have a monthly series with 72 observations and the goal is to forecast the next year, it makes sense to use the first 60 observations for training and test the performance using the last 12 observations. Let’s use the window function to split the series into the training and testing partitions.

Alternatively, we can also use the following partition structure where the sample.out argument set the size of the testing partition (and therefore the training partition):

## Jan Feb Mar Apr May Jun Jul Aug Sep Oct  
## 2018   
## 2019 3399.9 2999.2 2899.9 2201.1 2121.0 2115.2 2407.5 2437.2 2215.6 2472.3  
## Nov Dec  
## 2018 2709.9 2993.1  
## 2019

A weakness with such patitions is that it is feasible that a model by chance, will have relatively good performance on the test set, but do poorly on the actual forecast as it isn’t stable over time. One way to address this is to use a back testing approach, based on training a model with multiple training and testing partitions.

## Forecasting with backtesting

This back testing approach is based on the use of a rollin window to split the series into multiple pairs of training and testing partitions. A basic training process involves the following steps:

* Data preparation: create multiple pairs of training and testing partitions
* Train a model: this is done on each one of the training partitions
* Test the model: score its performance on the corresponding testing partitions.
* Evaluate the model: evaluate the model’s accuracy, scalibility and stability based on the testing score. Based on the evaluation, you could do either:
  + Generate the final forecast to check whether the model score meets a specific threshold or criteria
  + Apply additional tuning and optimization for the model and repeat the training and evaluation steps

Scoring methodology enables the researcher to be able to assess the model’s stability by examining the model’s error rate on the different testing sets. A model could be considered stable whenever the model’s error distribution on the testing set is fairly narrow. In this case, the error rate of the actual forecast should be within the same range of the testing sets (assuming there are no abnormal events that impact the forecast error rate)

## Forecast evaluation

The primary goal of the evaluation is to assess the ability of the trained model to forecast (or based on another criteria) the future observations of the series accurately. This includes:

* Residual analysis: focuses on model quality, with fitted values in the training paritition
* Scoring the forecast: this is based on the ability of the model to forecast the actual values of the testing set.

## Residual analysis

This tests how well the model captured and identified time series patterns and additionaly it provides information about the residual distributions, which are required to build confidence intervals for the forecast. To demonstrate this process, we will train an ARIMA model on the training parition created earlier for the USgas series. An ARIMA model, which stands for AutoRegressive Integrated Moving Average, is a popular and widely used statistical method for time series forecasting. It combines three components:

* Autoregression (AR)
* Integration (I)
* Moving Average (MA).

## Components of ARIMA:

### Autoregression (AR) part:

This part of the model involves regressing the time series on its own lagged (past) values. It captures the relationship between an observation and a number of lagged observations (p lags). The parameter denotes the number of lag observations included in the model (the order of the AR part).

### Integration (I) part:

This component involves differencing the time series to make it stationary. A stationary time series has constant statistical properties over time, such as mean and variance. The parameter represents the number of differences required to achieve stationarity.

### Moving Average (MA) part:

This part of the model involves modeling the error term as a linear combination of error terms occurring at various lagged times. It captures the relationship between an observation and a residual error from a moving average model applied to lagged observations (q lags). The parameter denotes the number of lagged forecast errors in the prediction equation (the order of the MA part).

## ARIMA Model Notation:

The ARIMA model is typically expressed as ARIMA(p, d, q), where:

* is the number of lag observations in the autoregressive model.
* is the degree of differencing required to make the series stationary.
* is the size of the moving average window.

## Series: train   
## ARIMA(2,1,1)(2,1,1)[12]   
##   
## Coefficients:  
## ar1 ar2 ma1 sar1 sar2 sma1  
## 0.4301 -0.0372 -0.9098 0.0117 -0.2673 -0.7431  
## s.e. 0.0794 0.0741 0.0452 0.0887 0.0830 0.0751  
##   
## sigma^2 = 10446: log likelihood = -1292.83  
## AIC=2599.67 AICc=2600.22 BIC=2623.2

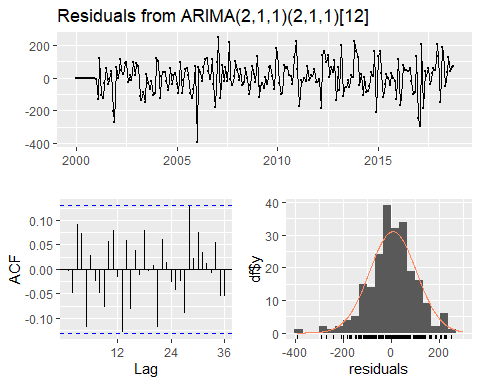
To examine the residuals, we can use the checkresiduals function, which returns the following four outputs:

* Time series plots of the residuals
* ACF plot of the residuals
* Distribution plot of the residuals
* The output of the Ljung-Box test

The Ljung Box test is a statistical method for testing whether the autocorrelation of a series (in this case the residuals) is different from zero and uses the following hypothesis:

H0: The level of autocorrelation between the series and its lag is equal to zero and therefore the series observations are independent H1: The level of autocorrelation between the series and its lag is different from zero and therefore, the series observations are not independent.

Let’s use checkresiduals function to evaluate the trained model’s performance on the training partition:



##   
## Ljung-Box test  
##   
## data: Residuals from ARIMA(2,1,1)(2,1,1)[12]  
## Q\* = 24.95, df = 18, p-value = 0.1263  
##   
## Model df: 6. Total lags used: 24

We can’t reject the null hypothesis with a level of significance of 0.1263, which indicates that the correlation between the residual series and it’s lags are not different from zero. The ACF plot confirms this as non of the lags go beyond the blue dotted line of statistical significance. In the residual time series plot, there are some negative which are around - 400, which could be outliers.

## Scoring the forecast

Once the model tuning has been forecast, we can test the model’s ability to predict observations that the model didn’t see before (as opposed to the fitted values that the model saw in the training process). The most common method is to use accuracy or error metrics and the most common method for evaluating the forecast success is to predict the actual values with an error metric to quantify forecast overall accuracy. The selection of a specific error metric depends on the forecast accuracy goals. The most common error metrics are as follows:

## Error Metrics

### Mean Squared Error (MSE)

The Mean Squared Error (MSE) is a common measure used to evaluate the accuracy of a model. It is the average of the squared differences between the actual and predicted values. A lower MSE indicates a better fit of the model to the data. The algebraic formula for MSE is:

where:

* is the number of observations
* is the actual value
* is the predicted value

### Root Mean Squared Error (RMSE)

The Root Mean Squared Error (RMSE) is the square root of the Mean Squared Error (MSE). It provides an error metric that is on the same scale as the original data, making it more interpretable. Like MSE, a lower RMSE indicates a better fit of the model. The algebraic formula for RMSE is:

where:

* is the number of observations
* is the actual value
* is the predicted value

### Mean Absolute Error (MAE)

The Mean Absolute Error (MAE) is another common measure used to evaluate the accuracy of a model. It is the average of the absolute differences between the actual and predicted values. MAE is less sensitive to outliers than MSE and RMSE. The algebraic formula for MAE is:

where:

* is the number of observations
* is the actual value
* is the predicted value

### Mean Absolute Percentage Error (MAPE)

The Mean Absolute Percentage Error (MAPE) expresses the prediction accuracy as a percentage. It is the average of the absolute percentage differences between the actual and predicted values. MAPE is useful for understanding the relative error magnitude. The algebraic formula for MAPE is:

where:

* is the number of observations
* is the actual value
* is the predicted value

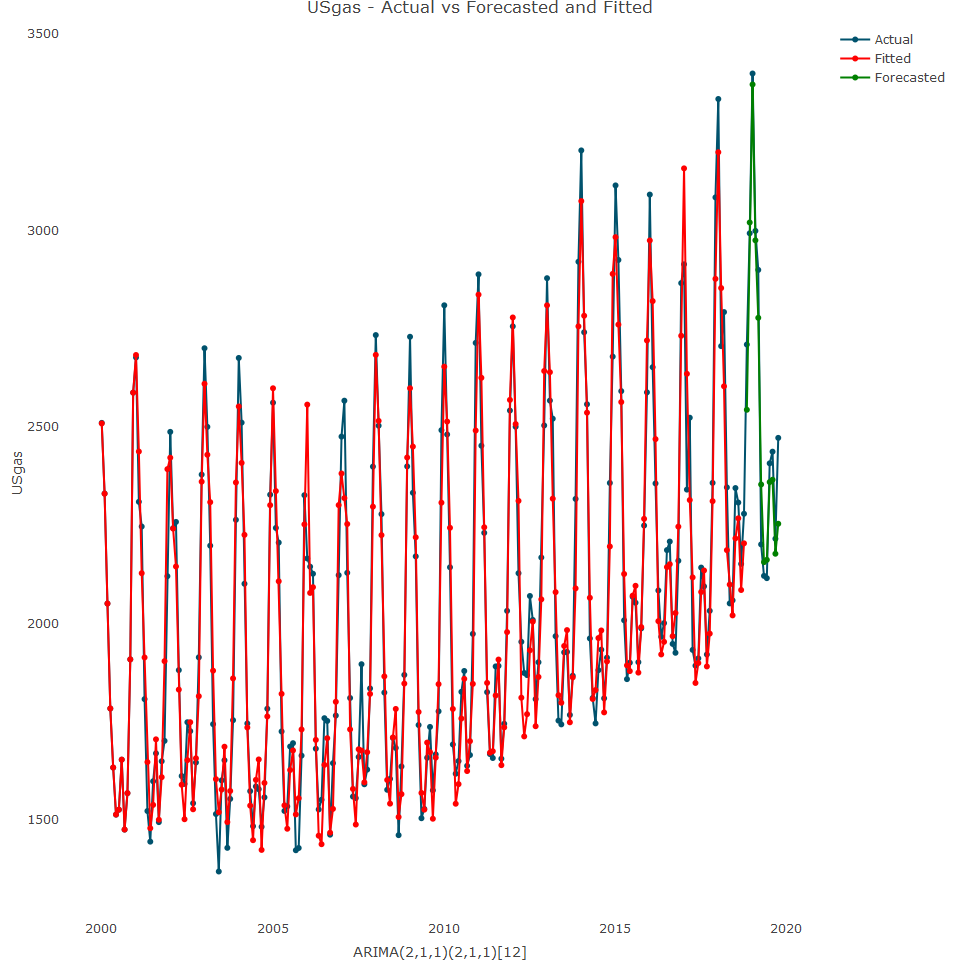
Lets use the model we trained earlier to forecast the 12 observations we left for testing and score its performance.

## Point Forecast Lo 80 Hi 80 Lo 95 Hi 95  
## Nov 2018 2543.848 2412.863 2674.832 2343.524 2744.171  
## Dec 2018 3020.624 2872.969 3168.278 2794.805 3246.442  
## Jan 2019 3372.018 3219.976 3524.059 3139.491 3604.545  
## Feb 2019 2975.209 2821.146 3129.272 2739.590 3210.828  
## Mar 2019 2777.974 2622.463 2933.485 2540.141 3015.807  
## Apr 2019 2353.685 2196.894 2510.476 2113.894 2593.476  
## May 2019 2155.955 1997.939 2313.970 1914.291 2397.618  
## Jun 2019 2161.846 2002.630 2321.062 1918.346 2405.346  
## Jul 2019 2359.808 2199.405 2520.212 2114.492 2605.125  
## Aug 2019 2365.725 2204.144 2527.306 2118.608 2612.842  
## Sep 2019 2177.364 2014.615 2340.114 1928.460 2426.268  
## Oct 2019 2253.671 2089.761 2417.581 2002.993 2504.350

## ME RMSE MAE MPE MAPE MASE  
## Training set 5.844136 97.81626 73.42657 0.1170672 3.522348 0.6376860  
## Test set 37.847885 103.22848 81.46603 1.3107987 3.261643 0.7075062  
## ACF1 Theil's U  
## Training set -0.004183172 NA  
## Test set -0.046708926 0.3404092

The MAPE percentages are quite even for train and test, which indicates over fitting hasn’t really occurred.

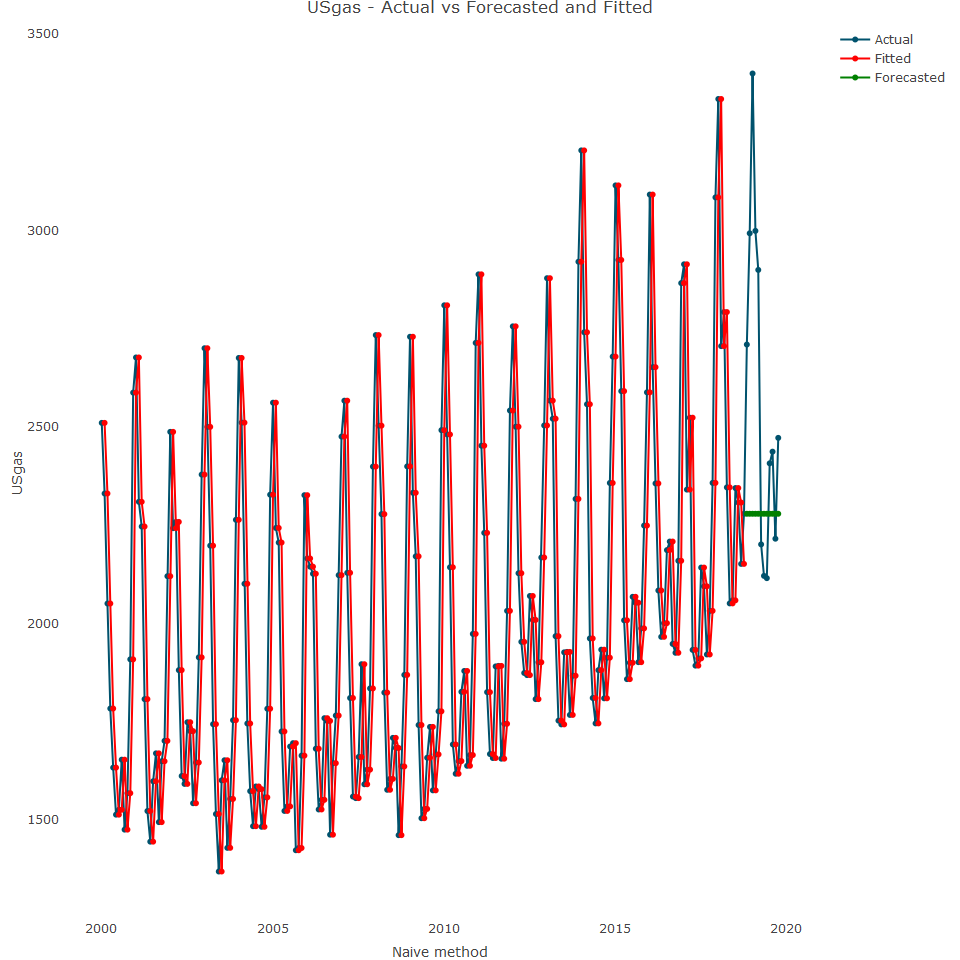
An alternative approach is to use the test\_forecast to visualise the actual series, the fitted values from the training partition and the forecasted values on the test set. We can see that the residual peak in 2006 is caused by outliers and the actual forecast missed the 2018 yearly peak. These insights cannot be observed with error metrics



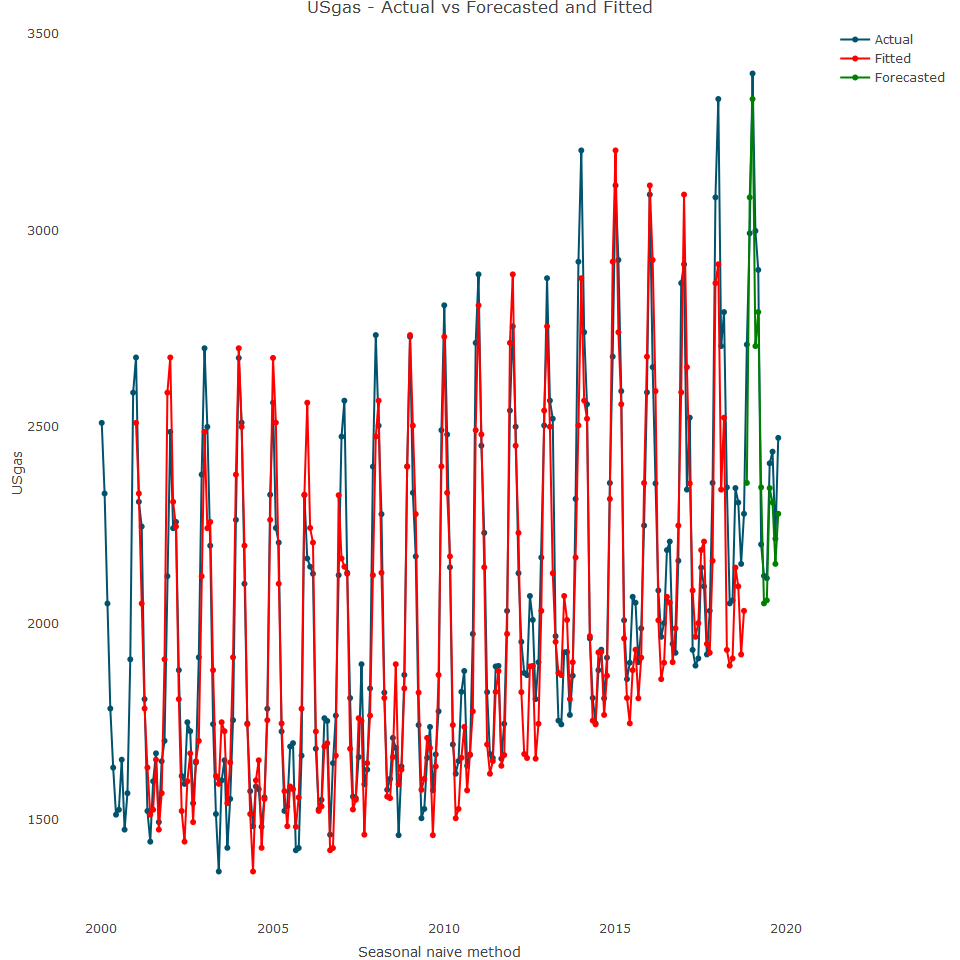
## Forecast benchmark

How can we judge if the error scores are too high or too low? A common method is to benchmark against some baseline forecast, such as a simplistic forecasting approach as benchmark. We can run a naive model, which assumes the most recently observed value to be the true representative of the future.

## ME RMSE MAE MPE MAPE MASE  
## Training set -1.028444 285.6607 228.5084 -0.9218463 10.97123 1.984522  
## Test set 301.891667 499.6914 379.1417 9.6798015 13.28187 3.292723  
## ACF1 Theil's U  
## Training set 0.3761105 NA  
## Test set 0.7002486 1.499679



Additionally, since USgas has a strong seasonal pattern, we can run a model that takes seasonlity into account, by using seasonal naive model, which uses the last seasonal point as a forecast of all future seasonal observations. For example, if you use a monthly series, the value of the most recent January in the series will be used as the point forecast for all future January months.



Lets use the accuracy function to review the seasonal naive performance

## ME RMSE MAE MPE MAPE MASE ACF1  
## Training set 33.99953 148.7049 115.1453 1.379869 5.494048 1.000000 0.4859501  
## Test set 96.45000 164.6967 135.8833 3.612060 5.220458 1.180103 -0.2120929  
## Theil's U  
## Training set NA  
## Test set 0.4289964

It seems that the snaive model has a better fit to the US gas series, due to its strong seasonal pattern than the naive model

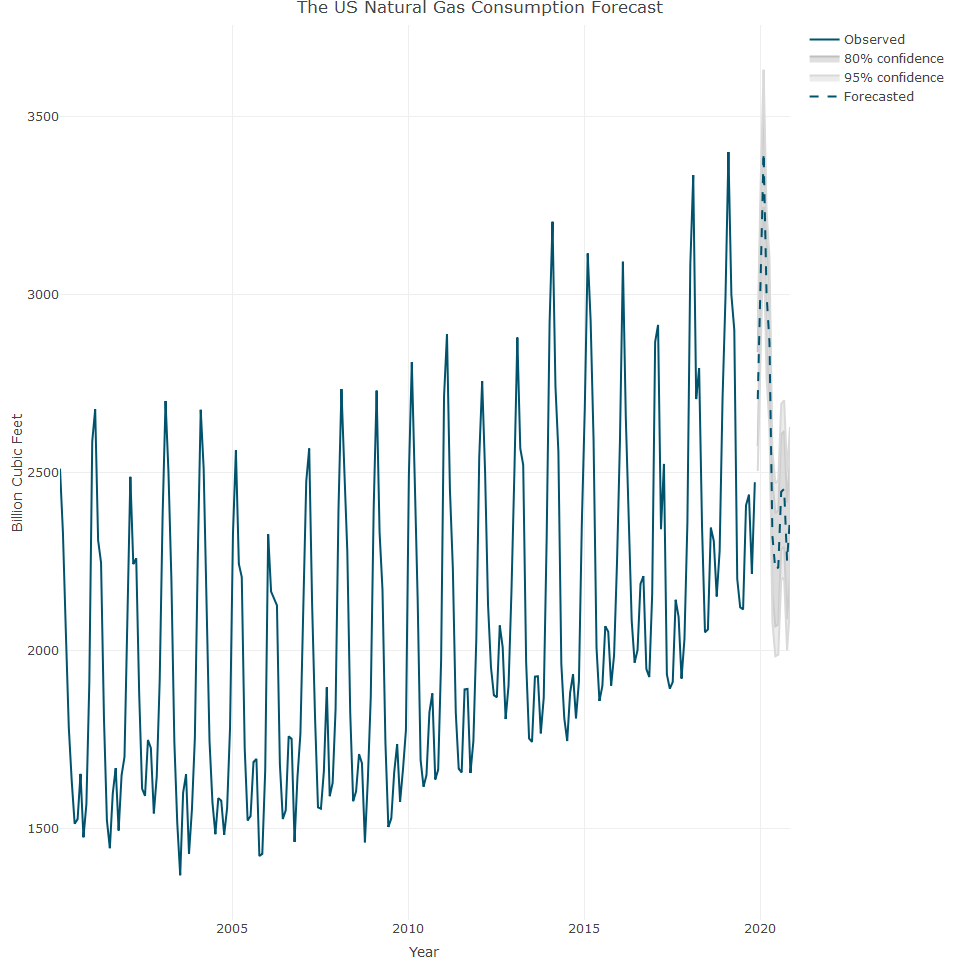
## Finalizing the forecast

Once a model has been trained, tested, tuned (if required) and evaluated, then it is possible to finalise the forecast. This step is based on recalibrating the model’s weights or coefficients with the full series. There are two approaches to using the model parameter setting:

* If a model was trained manually, then the exact tuning parameters should be used that were used on the trained model
* If the model was tuned automatically by an algorithm, then either of the following can be done:
  + Extract the parameter setting that was used with the training partition
  + Let the algorithym retune the model parameters using the full series, under the assumption that the algorithym has the ability to adjust the model parameters correctly when training the model with new data

Using algorithyms to automate the model tuning process is recommended when the model’s ability to tune the model is tested with backtesting. This allows you to review whether the algorithym has the ability to adjust the model parameters correctly based on backtesting results.

We can plot it:



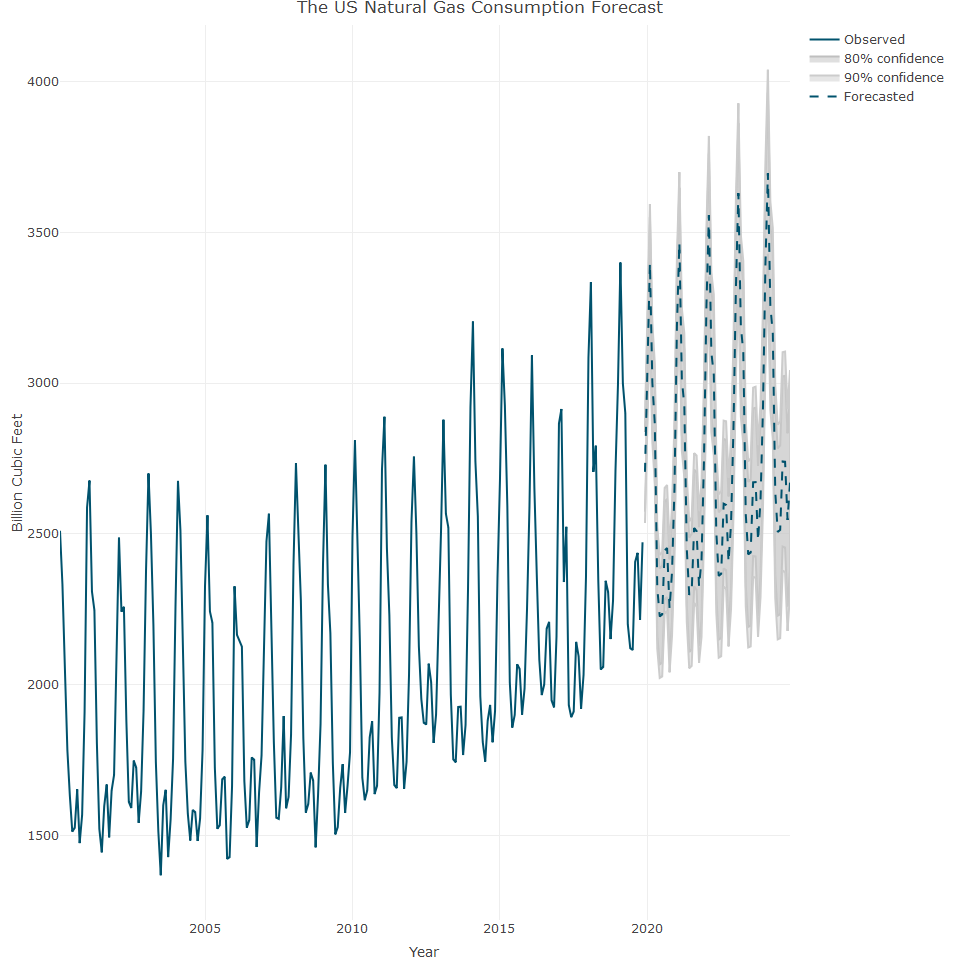
## Handling forecast uncertainty

The main purpose of forecasting is to minimize the level of uncertainty around the future values of the series. Although we cannot completely eliminate uncertainty, we can quanify it and provide some range around the point estimate of the forecast (which is nothing but the model’s expected value around each point in the future). This can be done by using either the confidence interval (or a credible interval, when using Bayesian model), or by using simulation.

## Confidence Interval

This is a statistical approximation of the range of possible values that contain the true value with some degree of confidence or probability. There are two parameters that determine the confidence interval range:

* The level of confidence or probability that the true value will be in that range. The higher the confidence level is, the wider the interval range.
* The estimated standard deviation of the forecast where the lower the error rate, the shorter the range of the prediction interval. By default, the forecast function has confidence interval range of 80% to 95%, but we could change it to 80% and 90%



## Simulation

An alternative aproach is to use the model distribution to simualate paths for the forecast. We can simulate using 100 iterations

## Horse race approach

This approach is based on training, testing and evaluating multiple forecasting models and selecting the model that performs the best on the testing partitions. We can apply this approach to seven models, as follows:

* auto.arima: Automated ARIMA model
* bsts: Bayesian structural time series model
* ets: Exponential smoothing state space model
* hybrid: An ensemble of multiple models
* nnetar: Neural network time series model
* tbats: Exponential smoothing state space model, along with Box-Cox transformation, trend, ARIMA errors and seasonal components
* HoltWinters: Hotl-Winters filtering

This function, compare\_time\_series\_models, performs the following tasks:

**Data Splitting:** It splits the input time series data into training and testing sets. The training set comprises 80% of the data, and the testing set comprises the remaining 20%.

**Model Fitting:** It fits four different time series models to the training data:

* ARIMA model (auto.arima)
* Exponential Smoothing (ETS) model (ets)
* Neural Network Autoregression model (nnetar)
* TBATS model (tbats)

**Forecast Generation:** It generates forecasts for each of the fitted models using the forecast function.

**Forecast Combination:** It combines the forecasts from the four models by taking the average of their predicted values.

**Error Calculation:** It calculates various error metrics to evaluate the performance of each model and the combined forecast. The error metrics calculated include:

* Mean Absolute Error (MAE)
* Root Mean Squared Error (RMSE)
* Mean Squared Error (MSE)
* Mean Absolute Percentage Error (MAPE)
* Dataframe Creation: It creates a dataframe (error\_rates) to store the error metrics for each model and the combined forecast.

**Output:** It prints a formatted table of the error rates using the kable function.

**Return:** Finally, it returns the error\_rates dataframe containing the error rates for each model and the combined forecast.

Overall, this function is designed to compare the performance of different time series forecasting models and provide a summary of their error rates.

##   
##   
## Table: Error Rates of Time Series Models  
##   
## | Model | MAE | RMSE | MSE | MAPE |  
## |:--------:|:--------:|:--------:|:--------:|:---------:|  
## | ARIMA | 330.8615 | 462.6402 | 214036.0 | 12.375796 |  
## | ETS | 331.6097 | 455.5222 | 207500.4 | 12.546454 |  
## | nnetar | 171.1658 | 217.2848 | 47212.7 | 6.881045 |  
## | TBATS | 305.0882 | 427.4046 | 182674.7 | 11.499919 |  
## | Combined | 256.6781 | 360.2577 | 129785.6 | 9.650431 |

## Model MAE RMSE MSE MAPE  
## 1 ARIMA 330.8615 462.6402 214036.0 12.375797  
## 2 ETS 331.6097 455.5222 207500.4 12.546454  
## 3 nnetar 171.1658 217.2848 47212.7 6.881045  
## 4 TBATS 305.0882 427.4046 182674.7 11.499919  
## 5 Combined 256.6781 360.2577 129785.6 9.650430

This output presents error rates for different time series models:

**ARIMA:** The mean absolute error (MAE) is 330.8615, indicating that on average, the ARIMA model’s forecasts deviate from the actual values by approximately 331 units. The root mean squared error (RMSE) is 462.6402, which means that the average error of the ARIMA forecasts is about 463 units. The mean squared error (MSE) is 214035.96, providing another measure of the average squared deviation between the forecasts and the actual values. The mean absolute percentage error (MAPE) is 12.375797%, indicating that, on average, the ARIMA model’s forecasts are off by approximately 12.38% from the actual values.

**ETS:** Similar to ARIMA, ETS has its error rates presented. MAE is 331.6097, RMSE is 455.5222, MSE is 207500.45, and MAPE is 12.546454%. ETS and ARIMA are comparable in terms of their performance based on these metrics.

**nnetar:** This model shows lower error rates compared to ARIMA and ETS. It has an MAE of 190.6301, RMSE of 224.0644, MSE of 50204.84, and MAPE of 8.018525%. These values suggest that the nnetar model performs better in terms of forecasting accuracy compared to ARIMA and ETS.

**TBATS:** TBATS model’s error rates are also presented. It has an MAE of 305.0882, RMSE of 427.4046, MSE of 182674.69, and MAPE of 11.499919%. TBATS performs slightly worse than the nnetar model but better than ARIMA and ETS.

**Combined:** This refers to the combined forecast obtained by averaging forecasts from all models. The error rates for the combined forecast are as follows: MAE of 251.3997, RMSE of 346.3504, MSE of 119958.57, and MAPE of 9.469518%. The combined forecast seems to outperform individual models, indicating that combining forecasts from multiple models can often lead to improved forecasting accuracy.

In summary, based on these error rates, the nnetar model appears to perform the best, followed by the combined forecast, TBATS, ARIMA, and ETS in descending order.