Sales Data Forecast

George Garcia, and Zachariah Freitas

Department of Engineering, University of San Diego

ADS 506: Applied Time Series Analysis

November 26, 2022

Project Introduction & Problem Statement

With inflation on the rise and the economy facing a recession, companies have started to look to reduce business costs; in a survey conducted by CBNC, they found that around 50% of companies will be reducing their headcount in the following year (Iacurci, 2022). To prevent any reduction in staff due to an economic impact, the executive leadership has asked us in our organization to supply a better forecast model than what is currently implemented by the finance team. Presently the finance team uses a simple heuristic model to generate the forecasts. Their method is to take a two-week average for the same days in the last two weeks of the previous month to predict the following month. The business objective for this project is to improve upon the RMSE of the naïve model. A better sales forecast will help our organization perform better business resource planning and help our leadership manage Wall Street expectations. Each 10% increase in RMSE will reduce costs by 1-2% and prevent unnecessary layoffs. To test the performance of our forecast, we will forecast the month of October on September 30 and attempt to improve upon the current naïve RMSE forecast benchmark of 28.16.

Literature Review

Upon doing some literature review on evaluation metrics for our forecasting, we found that most metrics are flawed and tend to have biases. According to Kolassa and Martin (2011) found that MAPE tends systematically under forecast. We also found that Root Mean Square Error (RMSE) is not reliable when we judge the data on reliability, construct validity, sensitivity to small changes, protection against outliers, and their relationship to decision making. Armstrong, JS and Collopy (1992). Since our data was stable and didn't have any outliers, we still chose to evaluate our models using RMSE instead of MAPE. This is because our stated goal

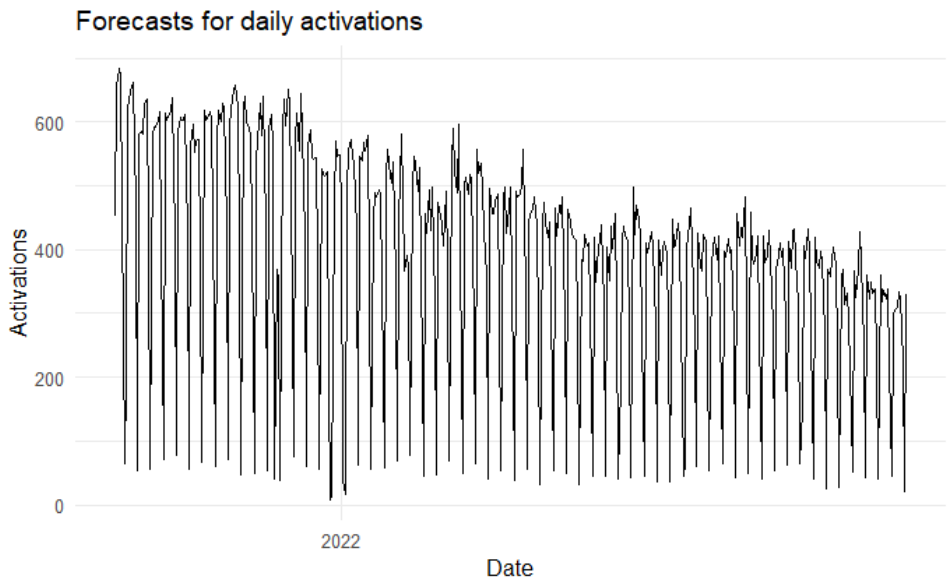
is to have prediction accuracy and we feel that RMSE is the most unbiased measure in this circumstance.

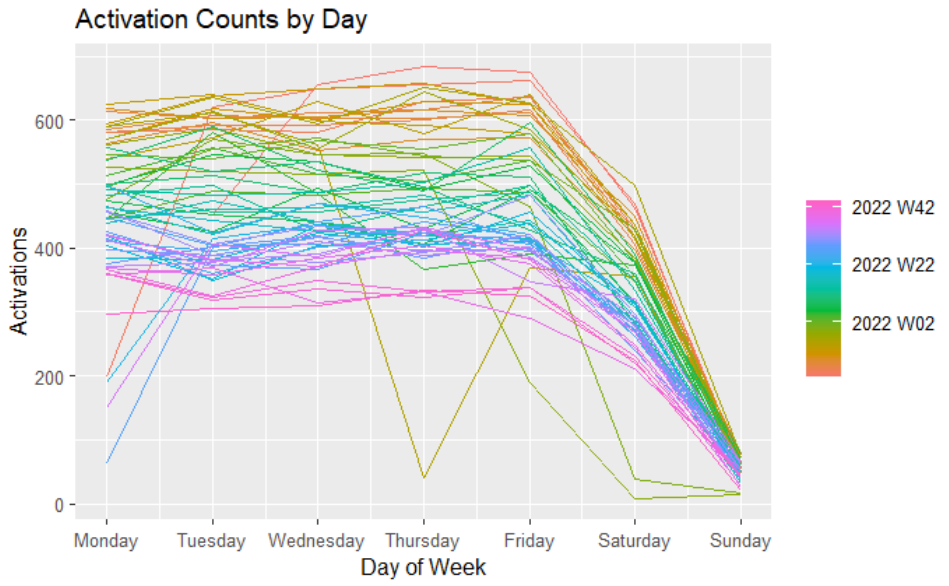
Exploratory Data Analysis (EDA)

The dataset utilized for this study came from the finance team in our organization and was an excel file that contained daily sales for our organization. The data included forty-six features. The following features: svc_agreement_activation_date , weekday, dayofweek, dayofmonth, weekend_flag, holiday_flag, dayofweek_1, dayofweek_2, dayofweek_3, dayofweek_4, dayofweek_5, dayofweek_6, dayofweek_7, holiday_flag_lag7, and holiday_flag_lag14 were nominal. The rest of the features were lags, from one to thirty days, of the main feature activation_count. The data had a total of 423 observations. The target feature was activation_count. The data set spans from 2021-08-31 to 2022-10-31. The programming language, R, was used to conduct this study.

Data Preparation

All data was provided to us by the Finance team. Since all the data points were present and there were no outliers, all the data was used for the prediction modeling. All smoothing, differencing, and any further transformation were done depending on the model. We reviewed our data and found that we had weekly seasonality with a downward trend for weekdays but not weekends as you can see in the two charts below.





Model - Linear Regression

The first model we applied on our dataset was simple linear regression using the tslm function. tslm is used to fit linear models to time series including trend and seasonality components. Our target feature was activation_counts, which is a continuous feature, the problem is a

supervised regression problem. One of the simplest models for solving a regression problem is linear regression; in this case, where multiple predictors are used to predict for one output, the model is called a multiple linear regression model. Thus, a simple multiple linear regression model was utilized as the baseline model; a model that would be used to compare the performance of other models against.

Model - Exponential Regression

The second model we tried is basically the same as the regression model mentioned above. This time we transformed the regression model into an exponential regression. In the tslm function there is a lambda value that can be used to do a box cox transformation on the dependent variable. We set the lambda value to 0, which performs a log transformation on the dependent value, turning our linear regression model into an exponential regression model.

Model - Neural Network

The third model we applied to our problem was a Feed-forward neural networks with a single hidden layer and lagged inputs for forecasting univariate time series. We allowed the nnetar algorithm to autofit the parameters.

Autoregressive Integrated Moving Average Model (ARIMA)

Our fourth model was an autoregressive integrated moving average model. ARIMA models directly model the autocorrelation of the series values as well as the autocorrelations of the forecast errors. The auto.arima() function fits best ARIMA model to univariate time series.

The function returns the best fitting ARIMA model according to either AIC, AICc or BIC value. The function conducts a search over possible model within the order constraints provided.

Ensemble Modeling

Our fifth and final model was a simple ensemble of the four models mentioned above. We simply take the predictions from each of the four models above and average them out.

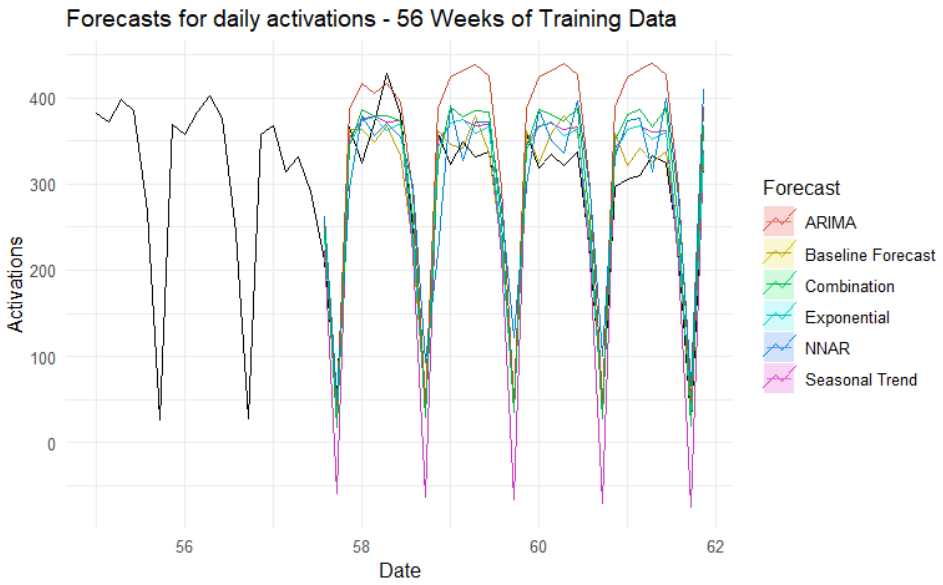
$$\frac{Neural\ Net\ +\ Linear\ Regression\ +\ Exponential\ Regression\ +\ ARIMA}{4}$$

Evaluation Metric

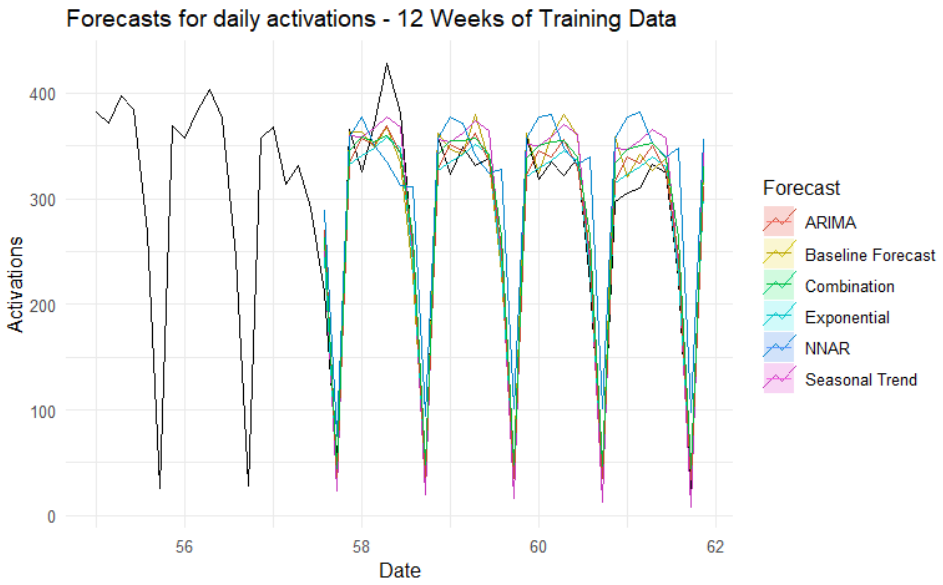
We decided to compare our model performance using root mean square error (RMSE). We chose RMSE because during our literature review, we discovered none of the data issues that would make us choose another evaluation metric.

Results

Upon running all of our models we found that some did a better job of predicting certain days of the week. For example, Seasonal Trend had a difficult time predicting Sunday, While some, like the ARIMA, Overestimated the weekdays. You can see the performance differences in the following chart.



When comparing the performance of our models to the baseline model, none of them outperformed the baseline model. The main difference between our models and the baseline model was the lookback period for the predection. Taking note of this, we tried reducing the lookback period of our training dataset from 56 weeks down to 12 weeks. The performance difference was dramatic. Nearly all of our models saw greater than a 25% increase in RMSE except for our Neural Net model. This makes sense because Neural Net models tend to work better in high frequency data scenarios.



Note in the table below that none of our model choices outperformed our baseline model when we had a larger training dataset. Also note that the best performance came from our Exponential Model.

Model RMSE Performance Results Table

Data Length (Weeks)	Baseline Performance (Two Weeks Only)	Linear Regression	Exponential Regression	Neural Net	ARIMA	Ensemble Model
56	28.16	53.05	30.88	45.72	74.87	38.20
12	28.16	28.83	22.00	53.29	26.72	28.83

Discussion

Reducing the lookback for our training model, we saw that we could improve our model RMSE by approximately 22%. This is dramatically better than expected. The results above suggest that we may want to investigate testing different lookback periods. We may find we would benefit with an even shorter lookback period. Also having the best model be a log model

may suggest that the other models may benefit from a log transformation as well. We may also want to try different configurations of the ensemble model to see if we can find a combination that outperforms our best model so that we can benefit from the best parts of all the models and mitigate our risk from the worst parts of all our models. In addition to the partition, we used to test our model choice, we may want to test a rolling partition to make sure our model performs consistently better historically.

Conclusion

In conclusion we've been tasked with finding a model that works better than our Finance team's Naïve baseline model. We've found a model that outperforms that model by 22%. We believe there is evidence that we may be able to improve that by more by trying some of the suggestions stated above in the discussion.

References

Armstrong, JS and Collopy, F (1992). Error measures for generalizing about forecasting methods: empirical comparisons. Int. J. Forecasting, 8, 69-80.

C. Tofallis. A better measure of relative prediction accuracy for model selection and model estimation. *Journal of the Operational Research Society*, 66:1352–1362, 2015.

Hyndman, RJ and Khandakar, Y (2008) "*Automatic time series forecasting: The forecast package for R*", Journal of Statistical Software, 26(3).

Iacurci, G. (2022, August 22). *50% of employers expect job cuts, survey finds. here's how to prepare for a potential layoff*. CNBC. Retrieved December 5, 2022, from https://www.cnbc.com/2022/08/22/50percent-of-employers-expect-layoffs-a-survey-found-heres-how-to-prepare.html

Kolassa, S. and Martin.R. (2011). Percentage errors can ruin your day (and rolling the dice show how. *Foresight*, issue 23, Fall 2011.

Shmueli, G., & Jr, K. L. C. (2016). *Practical Time Series Forecasting with R: A Hands-On Guide [2nd Edition] (Practical Analytics)* (2nd ed.). Axelrod Schnall Publishers.

Venables, W. N. and Ripley, B. D. (2002) *Modern Applied Statistics with S*. Fourth edition. Springer.

N. K. Ahmed, A. F. Atiya, N. El Gayar, and H. El-Shishiny. An empirical comparison of machine learning models for time series forecasting. *Econometric Reviews*, 29:594–621, 2010.

Appendix

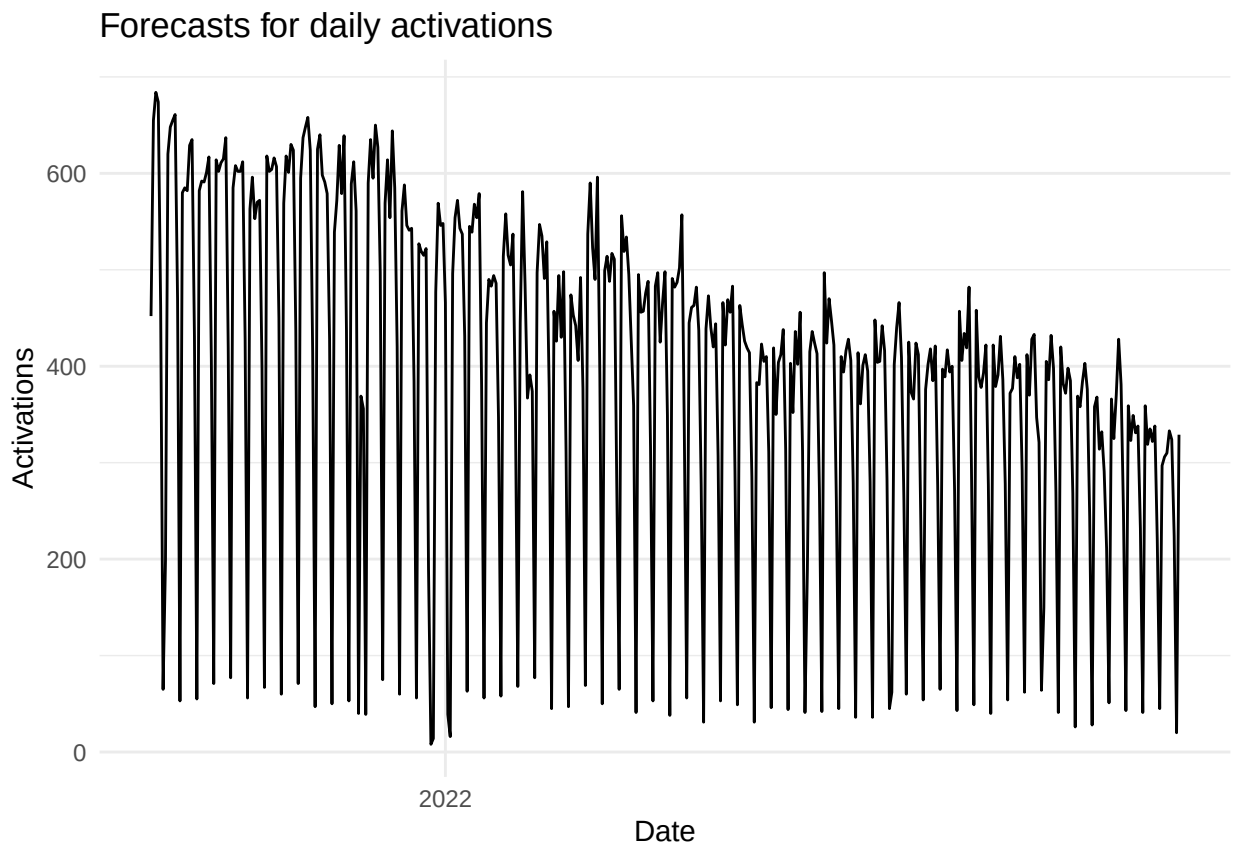
EDA

Zachariah Freitas

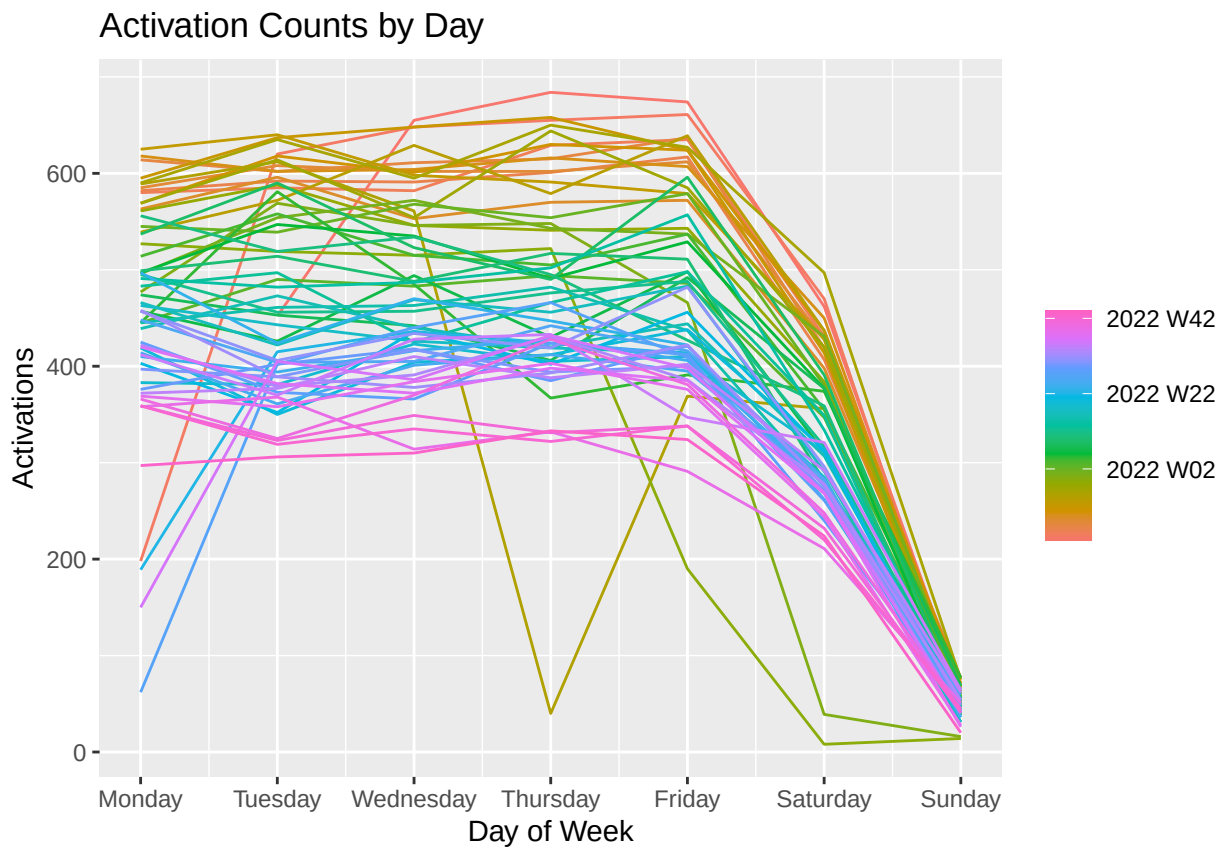
2022-12-05

EDA

```
# Create Time Series Object
myts <- ts(df$activation_count,
           start = c(2021, as.numeric(format(df$date[1], "%j"))),
           frequency = 364)


autoplot(myts) +
  xlab("Date") +
  ylab("Activations") +
  ggtitle("Forecasts for daily activations") +
  theme_minimal() +
  guides(colour=guide_legend(title="Forecast"))
```



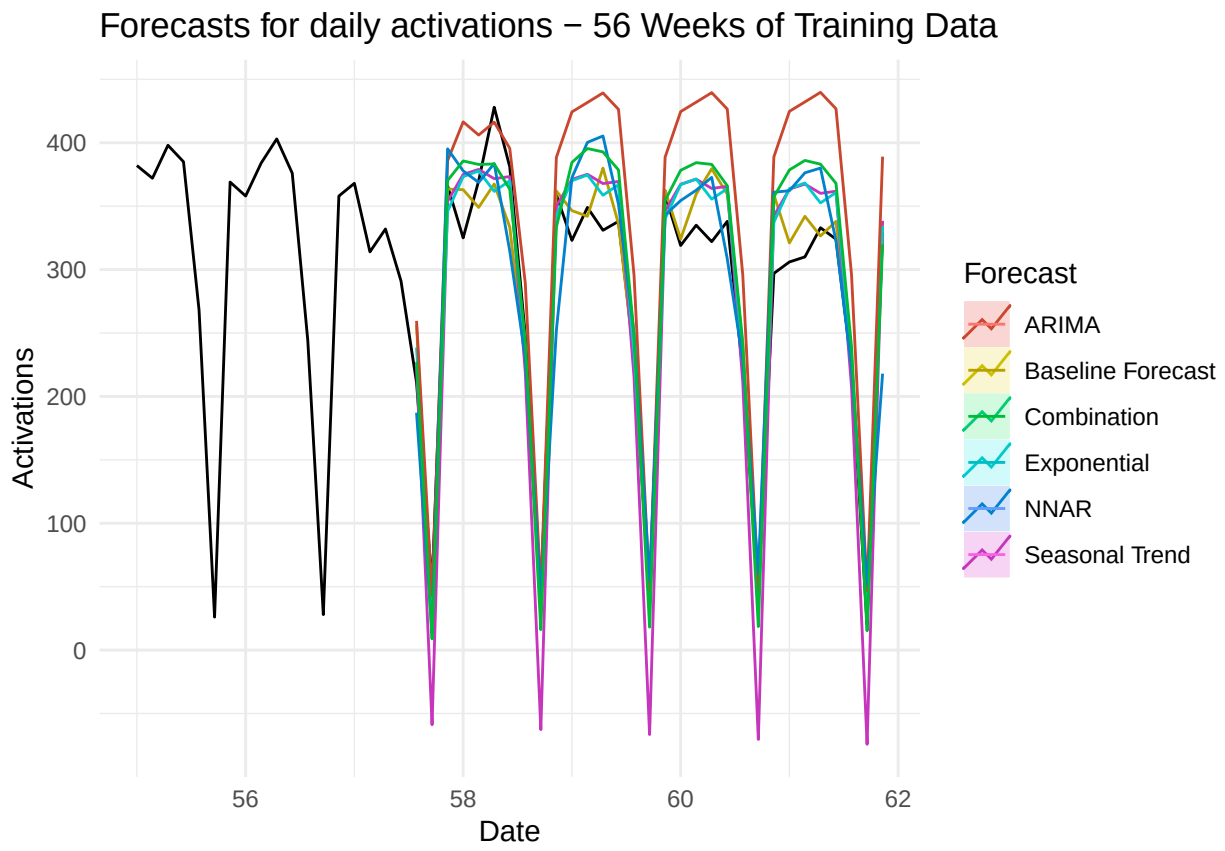

```
temp <- df %>%
  mutate(svc_agreement_activation_date = as.Date(svc_agreement_activation_date)) %>%
  as_tsibble(., index = svc_agreement_activation_date)

temp %>%
  gg_season(activation_count, period = "week") +
  theme(legend.position = "right") +
  labs(y="Activations", x="Day of Week", title="Activation Counts by Day")
```




```
library(forecast)

myts <- ts(df$activation_count,
           start = c(1, 1),
           frequency = 7)

# Split Data
train <- window(myts, end = c(57, 4)) #273 orig 300
valid <- window(myts, start = c(57, 5)) # October of 2022

# Set Forecast periods
h <- length(valid)


##############
# Train Models
# Trend + Seasonal Model
ST <- forecast::forecast(forecast::tslm(train ~ trend + season),h=h)

# Exponential Model
EXP <- forecast::forecast(forecast::tslm(train ~ trend + season, lambda = 0),h=h)

# NNAR - Autofit neural network
NNAR <- forecast::forecast(nnetar(train), h=h)#, lambda=0)
```

```
# Auto ARIMA
ARIMA <- forecast::forecast(auto.arima(train, lambda=0, biasadj=TRUE),h=h)


# Ensemble Approach
Combination <- (EXP[["mean"]] +
                  ARIMA[["mean"]] +
                  ST[["mean"]] +
                  NNAR[["mean"]])/4
```

```
############
# Baseline Naïve Model

naive_df <- df %>%
  mutate(naive_forecast = (lag7 + lag14)/2) %>%
  select(svc_agreement_activation_date, naive_forecast, activation_count)


# Create Time Series Object
mynaivets <- ts(naive_df$naive_forecast,
                start = c(1,1),
                frequency = 7)

# Plot Results
autoplot(window(myts, start=c(55, 1))) +
  autolayer(window(mynaivets, start=c(57, 5)), series="Baseline Forecast") +
  autolayer(ST, series="Seasonal Trend", PI=FALSE) +
  autolayer(EXP, series="Exponential", PI=FALSE) +
  autolayer(ARIMA, series="ARIMA", PI=FALSE) +
  autolayer(NNAR, series="NNAR", PI=FALSE) +
  autolayer(Combination, series="Combination") +
  xlab("Date") +
  ylab("Activations") +
  ggtitle("Forecasts for daily activations - 56 Weeks of Training Data") +
  theme_minimal() +
  guides(colour=guide_legend(title="Forecast"))
```



Including Plots

```
# Baseline model

resids <- naive_df %>%
  filter(svc_agreement_activation_date >= as.Date('2022-10-01')) %>%
  mutate(residuals = naive_forecast - activation_count) %>%
  select(residuals)

# RMSE
print("Baseline Forecast")
```

```
## [1] "Baseline Forecast"
```

```
print("RMSE - Test Set")
```

```
## [1] "RMSE - Test Set"
```

```
round(sqrt(mean(resids$residuals^2)),2)
```

```
## [1] 28.16
```

```
print("")
```

```
## [1] ""
```

```
print(strrep("#", 80))
```

```
## [1] "################################################################################"
```

```
print("Seasonal Trend")
```

```
## [1] "Seasonal Trend"
```

```
round(forecast::accuracy(ST, valid),2)
```

```
##                 ME  RMSE   MAE    MPE  MAPE MASE ACF1 Theil's U
## Training set 0.00 67.21 41.80 -19.71 42.10 0.95  0.3        NA
## Test set     3.22 53.06 42.06  42.51 54.68 0.96  0.2      0.08
```

```
print("")
```

```
## [1] ""
```

```
print(strrep("#", 80))
```

```
## [1] "################################################################################"
```

```
print("Exponential")
```

```
## [1] "Exponential"
```

```
round(forecast::accuracy(EXP, valid),2)
```

```
##                  ME  RMSE   MAE    MPE  MAPE MASE ACF1 Theil's U
## Training set  11.00 64.59 39.37 -16.10 28.59 0.89 0.31        NA
## Test set     -14.18 30.89 25.35  -5.65 12.20 0.58 0.29      0.06
```

```
print("")
```

```
## [1] ""
```

```
print(strrep("#", 80))
```

```
## [1] "################################################################################"
```

```
print("Neural Network")
```

```
## [1] "Neural Network"
```

```
round(forecast::accuracy(NNAR, valid),2)
```

```
##                  ME  RMSE   MAE   MPE  MAPE MASE  ACF1 Theil's U
## Training set -0.08 12.43  8.24 -1.50  4.48 0.19 -0.04        NA
## Test set     -7.97 46.92 37.51 -7.38 20.25 0.85  0.19      0.31
```

```
print("")
```

```
## [1] ""
```

```
print(strrep("#", 80))
```

```
## [1] "################################################################################"
```

```
print("ARIMA")
```

```
## [1] "ARIMA"
```

```
round(forecast::accuracy(ARIMA, valid),2)
```

```
##                   ME  RMSE   MAE    MPE  MAPE MASE  ACF1 Theil's U
## Training set -31.11 80.60 53.02 -23.73 29.49 1.20 -0.14        NA
## Test set     -62.31 74.87 63.58 -24.80 25.99 1.44  0.48      0.19
```

```
print("")
```

```
## [1] ""
```

```
print(strrep("#", 80))
```

```
## [1] "################################################################################"
```

```
print("Combination")
```

```
## [1] "Combination"
```

```
round(forecast::accuracy(Combination, valid),2)
```

```
##               ME  RMSE   MAE  MPE  MAPE ACF1 Theil's U
## Test set -20.31 41.32 35.07 1.17 18.94 0.44      0.08
```

```
print("")
```

```
## [1] ""
```

You can also embed plots, for example:

```
c(
  Baseline = round(sqrt(mean(resids$residuals^2)),2),
  ST = forecast::accuracy(ST, valid)["Test set","RMSE"],
  EXP = forecast::accuracy(EXP, valid)["Test set","RMSE"],
  NNAR = forecast::accuracy(NNAR, valid)["Test set","RMSE"],
  ARIMA = forecast::accuracy(ARIMA, valid)["Test set","RMSE"],
  Combination = forecast::accuracy(Combination, valid)["Test set","RMSE"])
```

```
##    Baseline          ST         EXP        NNAR       ARIMA Combination
##    28.16000    53.05689    30.88719    46.91997    74.87482    41.31594
```

Let's try adding the Baseline Forecast to the ensemble to see if we get an improvement in our prediction.

```
baseline.predict <- df %>%
  select(svc_agreement_activation_date, activation_count, lag7, lag14) %>%
  mutate(naive_forecast = (lag7 + lag14)/2) %>%
  filter(svc_agreement_activation_date >= as.Date('2022-10-01'))


NewCombination <- (EXP[["mean"]] +
                   ARIMA[["mean"]] +
                   ST[["mean"]] +
                   NNAR[["mean"]] +
                   baseline.predict[["naive_forecast"]])/5

round(forecast::accuracy(NewCombination, valid),2)
```

```
##                ME  RMSE   MAE   MPE MAPE ACF1 Theil's U
## Test set -17.82 37.16 31.28 -0.11   16 0.42      0.08
```

Try Shorter Lookback Windows

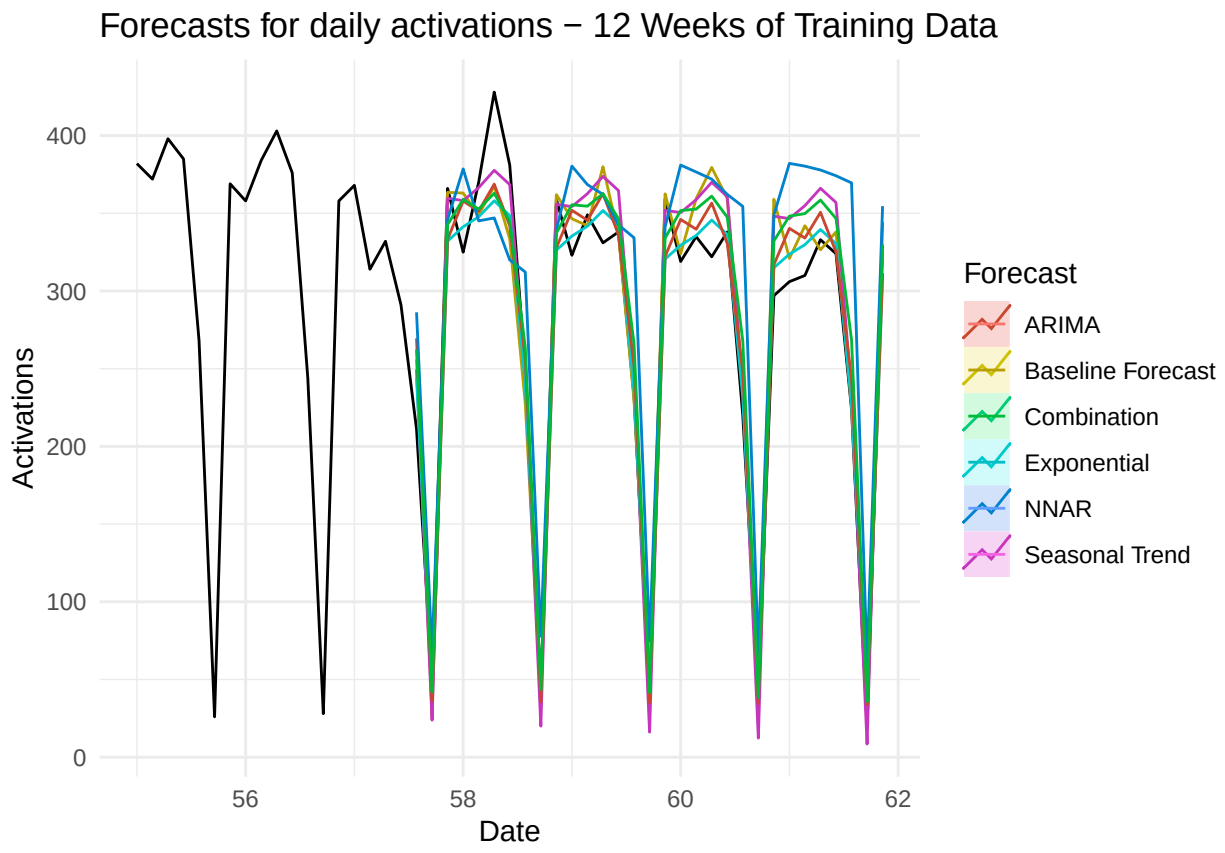
Seeing that the better performing model uses a smaller lookback period I want to try the same models with a smaller look back period.

```
library(forecast)

myts <- ts(df$activation_count,
           start = c(1, 1),
           frequency = 7)

# Split Data
train <- window(myts, start = c(45, 5), end = c(57, 4)) #273 orig 300
```

```
valid <- window(myts, start = c(57, 5)) # October of 2022

# Set Forecast periods
h <- length(valid)


##############
# Train Models
# Trend + Seasonal Model
ST <- forecast::forecast(forecast::tslm(train ~ trend + season),h=h)

# Exponential Model
EXP <- forecast::forecast(forecast::tslm(train ~ trend + season, lambda = 0),h=h)

# NNAR - Autofit neural network
NNAR <- forecast::forecast(nnetar(train), h=h)#, lambda=0)


# Auto ARIMA
ARIMA <- forecast::forecast(auto.arima(train, lambda=0, biasadj=TRUE),h=h)


# Ensemble Approach
Combination <- (EXP[["mean"]] +
                  ARIMA[["mean"]] +
                  ST[["mean"]] +
                  NNAR[["mean"]])/4
```

```
#############
# Baseline Naïve Model

naive_df <- df %>%
  mutate(naive_forecast = (lag7 + lag14)/2) %>%
  select(svc_agreement_activation_date, naive_forecast, activation_count)


# Create Time Series Object
mynaivets <- ts(naive_df$naive_forecast,
                start = c(1,1),
                frequency = 7)

# Plot Results
autoplot(window(myts, start=c(55, 1))) +
  autolayer(window(mynaivets, start=c(57, 5)), series="Baseline Forecast") +
  autolayer(ST, series="Seasonal Trend", PI=FALSE) +
  autolayer(EXP, series="Exponential", PI=FALSE) +
  autolayer(ARIMA, series="ARIMA", PI=FALSE) +
  autolayer(NNAR, series="NNAR", PI=FALSE) +
  autolayer(Combination, series="Combination") +
  xlab("Date") +
  ylab("Activations") +
  ggtitle("Forecasts for daily activations - 12 Weeks of Training Data") +
  theme_minimal() +
```

```
  guides(colour=guide_legend(title="Forecast"))
```



Including Plots

```
# Baseline model

resids <- naive_df %>%
  filter(svc_agreement_activation_date >= as.Date('2022-10-01')) %>%
  mutate(residuals = naive_forecast - activation_count) %>%
  select(residuals)

# RMSE
print("Baseline Forecast")
```

```
## [1] "Baseline Forecast"
```

```
print("RMSE - Test Set")
```

```
## [1] "RMSE - Test Set"
```

```
round(sqrt(mean(resids$residuals^2)),2)
```

```
## [1] 28.16
```

```
print("")
```

```
## [1] ""
```

```
print(strrep("#", 80))
```

```
## [1] "################################################################################"
```

```
print("Seasonal Trend")
```

```
## [1] "Seasonal Trend"
```

```
round(forecast::accuracy(ST, valid),2)
```

```
##                    ME  RMSE   MAE   MPE  MAPE MASE  ACF1 Theil's U
## Training set   0.00 36.40 21.89 -1.99  9.45 0.68 -0.03        NA
## Test set     -11.71 28.84 24.92  4.24 16.38 0.78  0.26      0.07
```

```
print("")
```

```
## [1] ""
```

```
print(strrep("#", 80))
```

```
## [1] "################################################################################"
```

```
print("Exponential")
```

```
## [1] "Exponential"
```

```
round(forecast::accuracy(EXP, valid),2)
```

```
##                 ME  RMSE   MAE   MPE  MAPE MASE ACF1 Theil's U
## Training set 2.26 37.25 24.90 -1.45 10.57 0.78 0.02        NA
## Test set     1.81 22.01 16.48 -1.95  8.90 0.51 0.17      0.09
```

```
print("")
```

```
## [1] ""
```

```
print(strrep("#", 80))
```

```
## [1] "################################################################################"
```

```
print("Neural Network")
```

```
## [1] "Neural Network"
```

```
round(forecast::accuracy(NNAR, valid),2)
```

```
##                   ME  RMSE   MAE    MPE  MAPE MASE ACF1 Theil's U
## Training set  -0.01 35.55 23.24  -3.31 11.19 0.73 0.08        NA
## Test set     -35.75 59.87 50.21 -26.82 30.53 1.57 0.15       0.1
```

```
print("")
```

```
## [1] ""
```

```
print(strrep("#", 80))
```

```
## [1] "################################################################################"
```

```
print("ARIMA")
```

```
## [1] "ARIMA"
```

```
round(forecast::accuracy(ARIMA, valid),2)
```

```
##                  ME  RMSE   MAE   MPE  MAPE MASE ACF1 Theil's U
## Training set  3.50 38.67 23.54 -2.31 10.73 0.73 0.20        NA
## Test set     -3.81 26.72 22.28 -1.97 11.64 0.70 0.01      0.08
```

```
print("")
```

```
## [1] ""
```

```
print(strrep("#", 80))
```

```
## [1] "################################################################################"
```

```
print("Combination")
```

```
## [1] "Combination"
```

```
round(forecast::accuracy(Combination, valid),2)
```

```
##               ME  RMSE   MAE   MPE  MAPE ACF1 Theil's U
## Test set -12.37 30.18 25.53 -6.63 11.85 0.19      0.06
```

```
print("")
```

```
## [1] ""
```

You can also embed plots, for example:

```
c(
  Baseline = round(sqrt(mean(resids$residuals^2)),2),
  ST = forecast::accuracy(ST, valid)["Test set","RMSE"],
  EXP = forecast::accuracy(EXP, valid)["Test set","RMSE"],
  NNAR = forecast::accuracy(NNAR, valid)["Test set","RMSE"],
  ARIMA = forecast::accuracy(ARIMA, valid)["Test set","RMSE"],
  Combination = forecast::accuracy(Combination, valid)["Test set","RMSE"])
```

```
##    Baseline          ST         EXP        NNAR       ARIMA Combination
##    28.16000    28.83790    22.00725    59.87182    26.72047    30.18254
```

Let's try adding the Baseline Forecast to the ensemble to see if we get an improvement in our prediction.

```
baseline.predict <- df %>%
  select(svc_agreement_activation_date, activation_count, lag7, lag14) %>%
  mutate(naive_forecast = (lag7 + lag14)/2) %>%
  filter(svc_agreement_activation_date >= as.Date('2022-10-01'))


NewCombination <- (EXP[["mean"]] +
                   ARIMA[["mean"]] +
                   ST[["mean"]] +
                   NNAR[["mean"]] +
                   baseline.predict[["naive_forecast"]])/5

round(forecast::accuracy(NewCombination, valid),2)
```

```
##               ME  RMSE   MAE   MPE  MAPE ACF1 Theil's U
## Test set -11.46 28.71 23.97 -6.34 11.64 0.23      0.06
```