Project 1
DATA624
6/18/23
Group 1
Members:

1. Bridget Boakeye
2. Keith Colella
3. Daniel Craig
4. Avery Davidowitz

OVERVIEW

The client provided a dataset requiring multiple forecasts. The dataset contains various series across five categories, and the client has identified two series per category for forecasting. Due to confidentiality concerns, all data has been de-identified. As a result, no business knowledge / domain expertise can be applied to the forecasting process.

Moreover, the client has noted that relationships across series should not be considered for modeling. As a result, our modeling has focused on using each series' unique history alone for prediction. In other words, we looked at traditional forecasting approaches that only use the past values of a series to predict future trends such as Auto Regressive (AR) and Moving Average (MA).

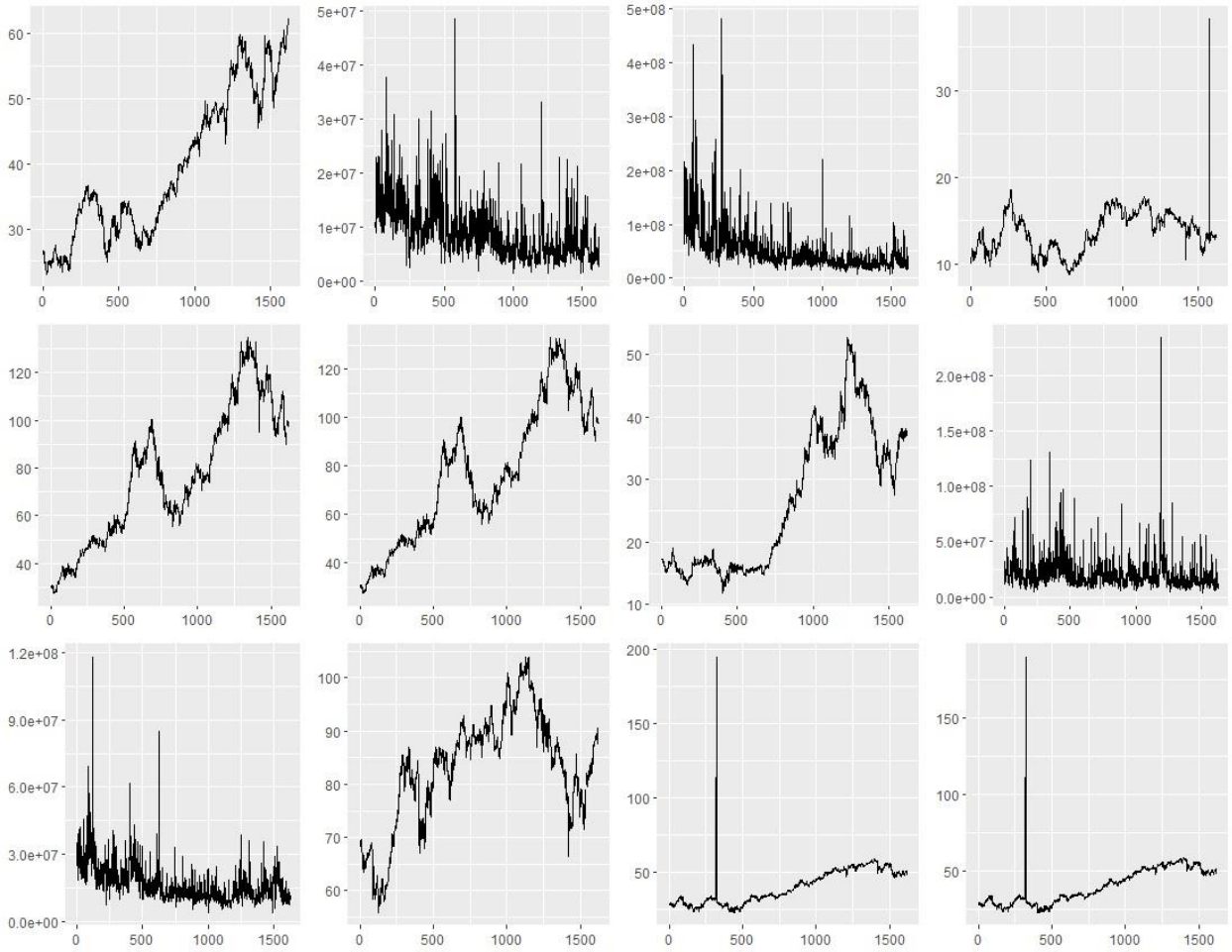
We assigned the series to four lead analysts, all of whom followed a similar protocol to explore the data, assess periodicity, fit models and evaluate the results. The final deliverable will consist of a proposed modeling approach for each of the 12 series, as well as a 140 point forecast for each.

The protocol for determining forecasts was as follows:

- Data was split into training and testing tests.
- Exploratory data analysis was used to review various characteristics of the data both visually and numerically to inform data cleaning and model calibration.
- Models were fit using the AR and MA approaches noted above (known as ARIMA models when used together).
- Finally, we evaluate our results with error metrics.

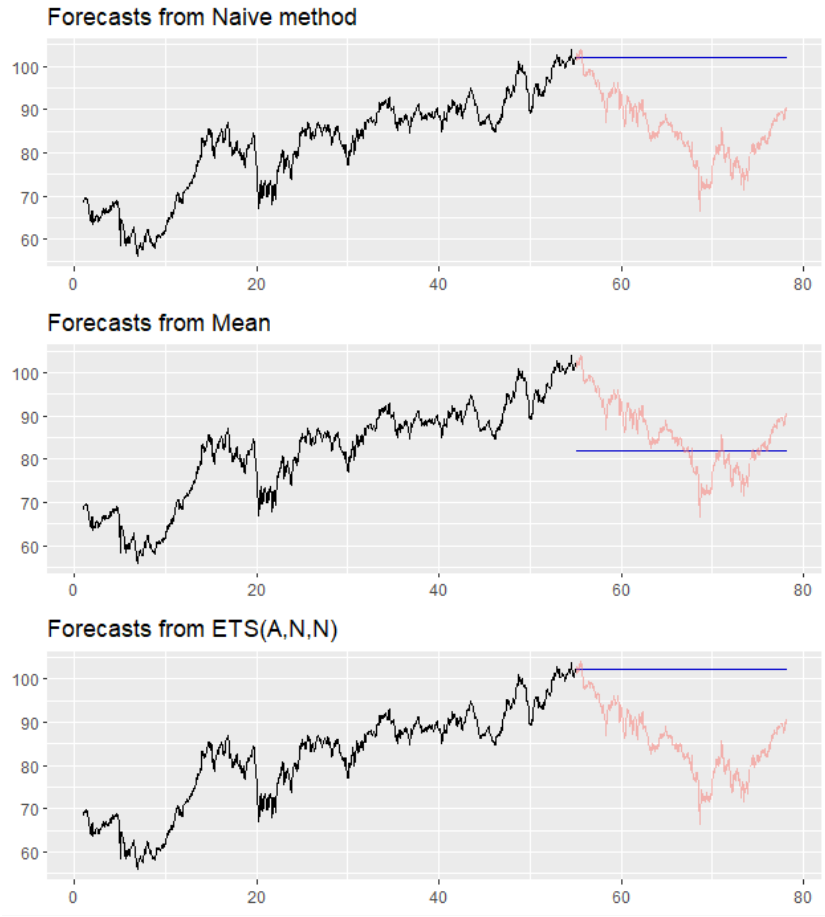
BASELINE FORECASTS

A quick visualization of the various series provided by the clients shows significant variety. First, we observe relatively noisy data with apparent outliers. Moreover, we see varied trends



In dealing with noisy data, we find it best to consider simple forecasting approaches before moving to more sophisticated modeling. These simple approaches serve as baselines to evaluate our modeled forecasts, but they also can provide meaningful forecasts in case advanced modeling does not prove fruitful.

We consider three baseline approaches. Samples of the forecasts produced by each of the above methods for series S01Var01 are shown below. The black line indicates the training data, the blue line indicates the forecast, and the red line indicates test data.



The results of these baseline approaches are detailed in the final section on modeling results.

EXPLORATORY DATA ANALYSIS

We approached our exploratory data analysis from a common framework. The steps conducted were:

1. Visualized the data using line plots.
2. Identified and handled missing values in the dataset using a variety of imputation techniques such as imputing the missing value with the most recent available observation and linear interpolation.
3. Identified outliers, skewness, and variability; where there was skewness, we attempted to correct it through log and Box-Cox transformation techniques.
4. Decomposed the time series into its components to identify underlying patterns and trends.
5. Conducted stationarity analysis and took differences where necessary.
6. Conducted autocorrelation analysis.

There were two primary takeaways from our EDA. Firstly, the datasets provided by the engineering team were generally clean. We identified gaps - missing values, extreme outliers - that needed to be corrected before forecasting. Filling in gaps for future datasets at the point of collection can be useful for providing more robust datasets in the future. Secondly, while we identified that some of our datasets had trend patterns and were non- stationary, we didn't find any significant seasonality. As a result, our models were non-seasonal models. We did attempt seasonal ARIMA models for some datasets for comparison.

Overall, the EDA performed on the dataset provided valuable insights into the datasets. Issues such as missing data and stationarity were addressed to support subsequent modeling and forecasting.

MODELING METHODOLOGY

We wished to create models robust to changing trends in the data, so we used 70% of the data to train models and the remaining 30% to evaluate their predictive capabilities. To set a baseline for prediction, we then created forecasts using three simple approaches: naive, average and exponential smoothing approaches. We used these baseline models to evaluate the performance of more sophisticated approaches.

The baseline models consisted of:

1. Naive Forecasting: The last value of the time series is taken as the forecasted value for the future.
2. Average Forecasting: The mean of the entire series is taken as the forecasted value for the future.
3. Exponential Smoothing (ETS): A weighted average of past values is taken as the forecast for the future. Older values are assigned less weight, and more recent values are assigned greater weight.

The most advanced model used was ARIMA. We first manually explored ARIMA models to select the key parameters in order to consider models with and without seasonality. We then compared these models to algorithmically calibrated ARIMA models, or "auto ARIMA" models.

RESULTS / CONCLUSION

ARIMA models proved to be quite effective in modeling this group of variables. At times, a naive approach or average approach performed quite well. Most of the variables were able to achieve a MAPE score of around 10%, while a handful were much higher and would be unreliable predictors. The table below with our final results is available for reference. At our client's request, we focused on the mean average percentage error (MAPE) to compare each model's predictive accuracy. This metric expresses forecast error in terms of percentages, which allows for comparison across series measured in different units (as noted above, the de-identified series contained no info on unit of measurement). In

some cases, our ARIMA models provided considerable improvements over the simple baseline approaches. However, in other cases the noisiness of the data introduced significant challenges for modeling. Therefore, we opted to leverage the baseline approaches for forecasting when they performed well.

More advanced machine learning methods may perform better for subsequent analysis. However, a more robust analysis predicting over a considerably sized 140 period span may also require additional data and the removal of data sanitizing requirements.

Lead	Category	Variable	Naïve	Average	ETS	ARIMA	Chosen Model
Daniel	S01	Var01	12.22%	22.40%	8.98%	5.81%	ARIMA(2,1,1)
Daniel	S01	Var02	31.96%	52.96%	34.81%	30.50%	ARIMA(2,1,1)
Keith	S02	Var02	28.15%	127.42%	37.66%	28.42%	ARIMA(2,1,2)
Keith	S02	Var03	18.25%	10.64%	18.24%	18.01%	Average
Bridget	S03	V05	13.03%	45.58%	13.14%	10.86%	ARIMA(3,1,2) drift
Bridget	S03	V07	13.44%	43.59%	11.61%	10.91%	ARIMA(1,1,3) drift
Avery	S04	Var01	13.17%	45.07%	13.17%	13.18%	ETS
Avery	S04	Var02	74.43%	47.89%	43.85%	53.54%	ETS
Daniel	S05	Var02	28.07%	55.10%	35.79%	28.49%	Naive
Keith	S05	Var03	20.32%	7.72%	20.32%	20.22%	Average
Bridget	S06	V05	4.74%	35.13%	4.73%	9.99%	ETS
Avery	S06	Var07	4.70%	34.85%	6.32%	9.64%	Naive

APPENDIX

R code for the general workflow is included below for interested parties. The complete R code used to generate this report is available at the team GitHub repository for Project 1

````
```{r setup, message = FALSE}
library(tidyverse)
library(forecast)
library(urca)
```

Choice of variable / model parameters

```{r}
# knitting
show_code <- FALSE

# data
category <- 'S04'
variable <- 'Var02'
frequency <- 21
````
````

```
# skewness (method must be 'BoxCox' or 'log')
transform <- TRUE
trans_method <- 'BoxCox'
transform_diff <- FALSE
trans_method_diff <- 'BoxCox'

# seasonality
seasonal_periods <- c(5,21,252)

# outliers
outlier_removal <- FALSE

# missing values (method must be 'linear' or 'spline')
interpolate <- TRUE
missing_method <- 'linear'

# arima
p <- 2
d <- 1
q <- 3
drift <- FALSE

# auto arima
approximation <- TRUE
stepwise <- TRUE

#seasonal rima
p_s <- 1
d_s <- 0
q_s <- 1
drift_s <- FALSE
```

Read in data. Convert to ts object and plot.

```{r, echo = show_code}
xl <- readxl::read_excel('DATA624_Project1_Data_Schema.xlsx',
                 sheet = category, skip = 2)

series <- xl[1:1622,variable]

series <- ts(deframe(series), frequency = frequency)

head(series)
autoplot(series)
```

Split training and test sets

```{r, echo = show_code}
train <- head(series, round(length(series) * 0.7))
test <- tail(series, round(length(series) * 0.3))

cat('Obs. in test set:',length(train),
    '\nObs. in test set:',length(test),
    '\nTotal observations:',length(series))
```

```
autoplot(train)
autoplot(test)
```

Split training and test sets with differenced series

```{r, echo = show_code}
train_diff <- diff(train)
test_diff <- diff(test)

autoplot(train_diff)
autoplot(test_diff)
```

Baseline forecasts - Naive, Average and ETS

```{r, echo = show_code}
naive_model <- naive(train, h = length(test), level = 0)
autoplot(naive_model) +
  autolayer(test, alpha = 0.5)

mean_model <- meanf(train, h = length(test), level = 0)
autoplot(mean_model) +
  autolayer(test, alpha = 0.5)

ets_model <- ets(train)
ets_forecast <- forecast(ets_model, h = length(test), PI = FALSE)
autoplot(ets_forecast) +
  autolayer(test, alpha = 0.5)
```

Baseline forecasts on differenced series

```{r, echo = show_code}
naive_model_d <- naive(train_diff, h = length(test), level = 0)
naive_model_d_levels <- ts(cumsum(c(tail(train,1),
naive_model_d$mean)),
                           frequency = frequency, start =
time(test)[1])
autoplot(train) +
  autolayer(naive_model_d_levels) +
  autolayer(test, alpha = 0.5) +
  scale_color_manual(values = c('blue','red'),
                     label = c('actual','prediction'))

mean_model_d <- meanf(train_diff, h = length(test), level = 0)
mean_model_d_levels <- ts(cumsum(c(tail(train,1), mean_model_d$mean)),
                          frequency = frequency, start = time(test)[1])
autoplot(train) +
  autolayer(mean_model_d_levels) +
  autolayer(test, alpha = 0.5) +
  scale_color_manual(values = c('blue','red'),
                     label = c('actual','prediction'))

ets_model_d <- ets(train_diff)
ets_forecast_d <- forecast(ets_model_d, h = length(test), PI = FALSE)
```

```
ets_model_d_levels <- ts(cumsum(c(tail(train,1), ets_forecast_d$mean)),
                         frequency = frequency, start = time(test)[1])
autoplot(train) +
  autolayer(ets_model_d_levels) +
  autolayer(test, alpha = 0.5) +
  scale_color_manual(values = c('blue','red'),
                     label = c('actual','prediction'))
```

Check for skewness

```{r, echo = show_code}
ggplot(fortify(train), aes(y)) +
  geom_histogram(bins = 50)

if (transform == TRUE) {
  if (trans_method == 'BoxCox') {
    train <- BoxCox(train,
                    lambda = BoxCox.lambda(transform, method =
'loglik'))
  } else if (method == 'log') {
    train <- log(train)
  }
  cat('Series transformed using',trans_method)
}

ggplot(fortify(train_diff), aes(y)) +
  geom_histogram(bins = 50)

if (transform_diff == TRUE) {
  if (trans_method_diff == 'BoxCox') {
    train_diff <- BoxCox(train_diff,
                         lambda = BoxCox.lambda(transform, method =
'loglik'))
  } else if (trans_method_diff == 'log') {
    train_diff <- log(train_diff)
  }
  cat('Differenced series transformed using',trans_method_diff)
}
```

Check for stationarity

```{r, echo = show_code}
summary(ur.kpss(train))

cat('\n################################',
    '\nSuggested number of differences:',
    ndiffs(train))
```

Check for outliers

```{r, echo = show_code}
# tsoutliers(train)
# tsoutliers(train_diff)
```

```
ggplot(fortify(train), aes(x,y)) +
  geom_point(
    color = if_else(time(train) %in% tsoutliers(train)$index,
                    'red', 'black'))

ggplot(fortify(train_diff), aes(x,y)) +
  geom_point(
    color = if_else(time(train_diff) %in% tsoutliers(train_diff)$index,
                    'red', 'black'))

if (outlier_removal == TRUE) {
  train <- tsclean(train, replace.missing = FALSE)
  train_diff <- tsclean(train_diff, replace.missing = FALSE)
  cat('Outliers removed.')
}
```

Check for missing values

```{r, echo = show_code}
length(train[is.na(train)])

if (interpolate == TRUE) {
  if (missing_method == 'linear') {
    train <- zoo::na.approx(train)
    train_diff <- zoo::na.approx(train_diff)
  } else if (missing_method == 'spline') {
    train <- zoo::na.spline(train)
    train_diff <- zoo::na.spline(train_diff)
  }
  cat('Missing values replaced using',missing_method,'method')
}
```

Check for seasonality

```{r, echo = show_code}
for (period in seasonal_periods) {
  seasonal_ts <- ts(train, frequency = period)
  cat('\nSuggested number of seasonal diffs for freq =',period,'-',
      nsdiffs(seasonal_ts))
  print(stl(seasonal_ts,
            s.window = period,
            t.window = period,
            robust = TRUE) %>%
        autoplot())
}
```

Check ACF/PACF

```{r, echo = show_code}
ggAcf(train)
ggPacf(train)

ggAcf(train_diff)
ggPacf(train_diff)
```

```

Manual ARIMA model fit

```{r, echo = show_code}
arima_model <- Arima(train, order = c(p,d,q), include.drift = drift)
summary(arima_model)
checkresiduals(arima_model)
arima_forecast <- forecast(arima_model, h = length(test))
autoplot(arima_forecast) +
  autolayer(test)
```

Manual seasonal ARIMA fit

```{r, echo = show_code}
arima_model_s <- Arima(train,
                       order = c(p,d,q),
                       seasonal = c(p_s,d_s,q_s),
                       include.drift = drift_s)
summary(arima_model_s)
checkresiduals(arima_model_s)
arima_forecast_s <- forecast(arima_model_s, h = length(test))
autoplot(arima_forecast_s) +
  autolayer(test)
```

Auto ARIMA

```{r, echo = show_code}
autoarima_model <- auto.arima(train,
                              approximation = approximation,
                              stepwise = stepwise)
summary(autoarima_model)
checkresiduals(autoarima_model)
autoarima_forecast <- forecast(autoarima_model, h = length(test))
autoplot(autoarima_forecast) +
  autolayer(test)
```

Evaluation

```{r, echo = show_code}
naive_mape <- mean(abs(naive_model$mean - test) / test, na.rm = TRUE)
naive_mape_d <- mean(abs(naive_model_d_levels - test) / test, na.rm =
TRUE)
mean_mape <- mean(abs(mean_model$mean - test) / test, na.rm = TRUE)
mean_mape_d <- mean(abs(mean_model_d_levels - test) / test, na.rm =
TRUE)
ets_mape <- mean(abs(ets_forecast$mean - test) / test, na.rm = TRUE)
ets_mape_d <- mean(abs(ets_model_d_levels - test) / test, na.rm = TRUE)

arima_mape <- mean(
  abs(arima_forecast$mean - test) / test, na.rm = TRUE)

arima_mape_s <- mean(
  abs(arima_forecast_s$mean - test) / test, na.rm = TRUE)
```
```

```
autoarima_mape <- mean(
  abs(autoarima_forecast$mean - test) / test, na.rm = TRUE)

results <- data.frame(
  model = c('naive','naive_d',
            'mean','mean_d',
            'ets','ets_d',
            'arima','seasonal arima', 'autoarima'),
  mape = c(naive_mape, naive_mape_d,
           mean_mape, mean_mape_d,
           ets_mape, ets_mape_d,
           arima_mape, arima_mape_s, autoarima_mape)
)

results %>%
  arrange(mape)
```