ISA 444: Business Forecasting

28: A 20-minute Introduction to ML for TS Data

Fadel M. Megahed, PhD

Endres Associate Professor
Farmer School of Business
Miami University

@FadelMegahed
fmegahed
fmegahed@miamioh.edu
Automated Scheduler for Office Hours

Spring 2023

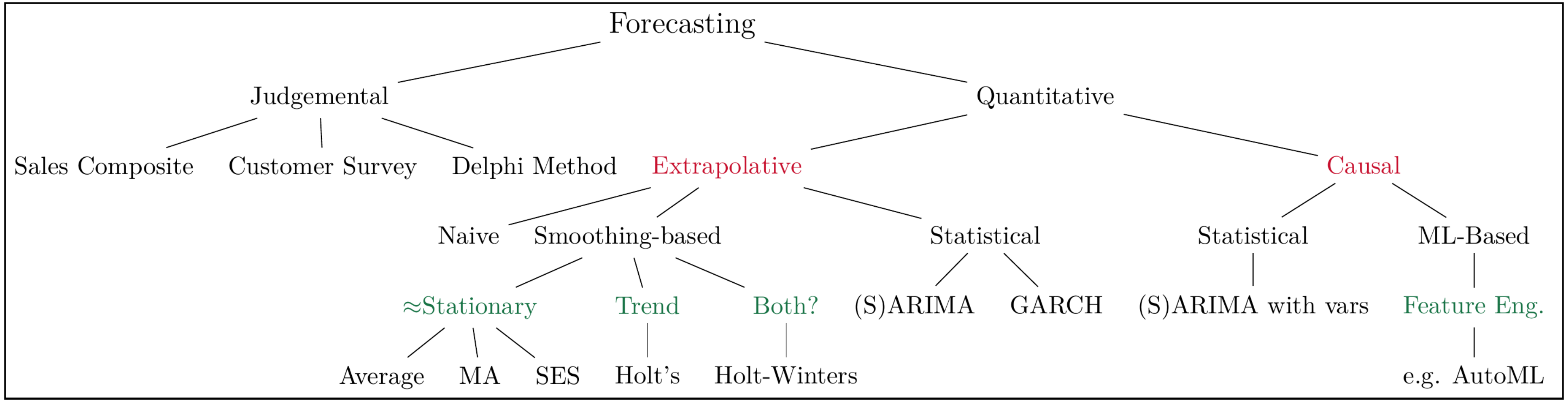
Quick Refresher from Last Class

- ✅ Combine regression with ARIMA models to model a time series with autocorrelated errors.
- ✅ Use the `xreg` argument to combine ARIMA models with regression predictors.

Class Activity Solution

```r
uschange = fpp2::uschange

# Solutions for the Questions
# (1a) Extrapolative forecasting using auto.arima (i.e., only the time-series for Consumption)
model2 = forecast::auto.arima(uschange[, 'Consumption'])
summary(model2)
resplot(res = model2$residuals, fit = model2$fitted, freq = 4)

# (1b) Reg with Income and Savings and ARIMA structure imposed on the error term
model3 = forecast::auto.arima(uschange[,'Consumption'], xreg = uschange[, c('Income', 'Savings') ])
summary(model3)
resplot(res = model3$residuals, fit = model3$fitted, freq = 4)

# (1c) Lm using both income and savings
model4 = lm(uschange[,'Consumption'] ~ uschange[,'Income'] + uschange[,'Savings'])
summary(model4)

# (1d) Income only
model5 = lm(uschange[,'Consumption'] ~ uschange[,'Income'] )
summary(model5)
resplot(model5$residuals, model5$fitted.values, freq = 4)


# How to make predictions about future values (made up values)
predict(model1, newxreg = c(0.1, -0.2)) # for model1 (one predictor)
# Alternatively
forecast::forecast(model1, xreg = c(0.1, -0.2)) # for model1 ( (one predictor)

# For forecast models with multiple predictors, you will have to add the information as a data.matrix
forecast::forecast(model3,
         xreg = data.frame( Income = c(0.1, -0.2), Savings = c(0.1, 0.2) ) |> data.matrix()  )
```

Overview of Univariate Forecasting Methods



A 10,000 foot view of forecasting techniques

Notes: My (incomplete) classification of **univariate** forecasting techniques, i.e., they exclude popular approaches used in multivariate time series forecasting.

Learning Objectives for Today's Class

- Explain how ML, and other advanced models, can be applied to TS data (given that we will be introducing this for **20 minutes** prior to answering questions pertaining to your final exam, this will be a very quick demo).

ML for TS Data

(An Example from Fadel's Research)

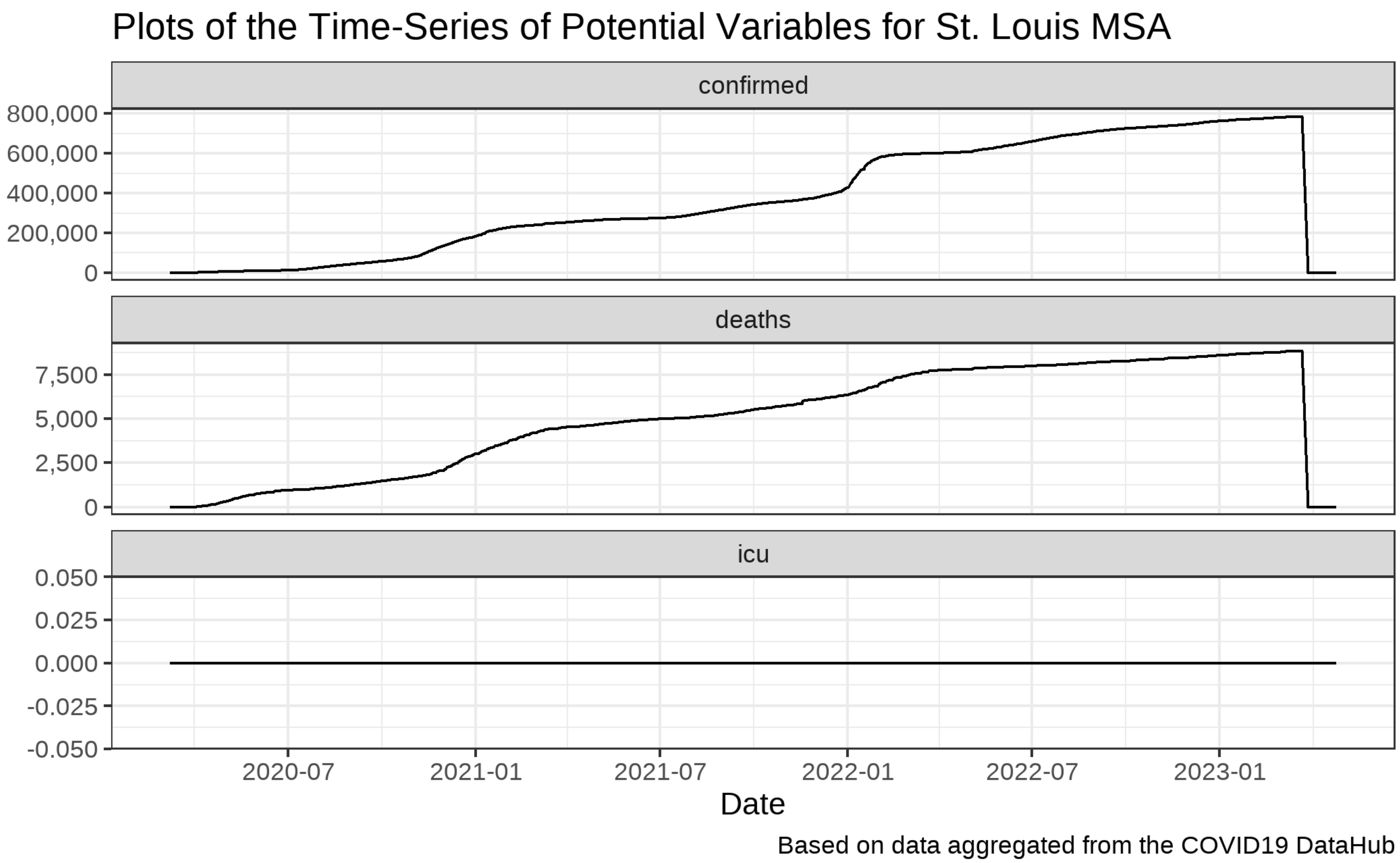
COVID Deaths in Saint Louis City, MO

```r
# creating a temp file for downloading the data
temp = tempfile()

# download the file to temporary location
download.file("https://storage.covid19datahub.io/country/USA.csv.zip", temp)

# unzip and read the file
covid_tbl = unz(temp, "USA.csv") |>
  # reading the data from the CSV
  readr::read_csv() |>
  # filtering to Missouri and Illinois
  dplyr::filter(administrative_area_level_2 %in% c('Illinois', 'Missouri') )


st_louis_tbl = covid_tbl |>
  dplyr::filter(
    (administrative_area_level_2 == 'Illinois' &
       administrative_area_level_3 %in% c('Bond', 'Calhoun', 'Clinton', 'Jersey', 'Macoupin', 'Madison', 'Monroe') ) |
      (administrative_area_level_2 == 'Missouri' &
         administrative_area_level_3 %in% c('Crawford', 'Franklin', 'Jefferson', 'Lincoln', 'St. Charles', 'St. Clair', 'St. Louis', 'St. Louis City', 'Wa
  )

# Aggregating the counts by day (so that we have an approximation of total numbers for st. louis)
st_louis_agg_tbl =
  st_louis_tbl |>
  # grouping by date so we can created an aggregated summation across all counties
  dplyr::group_by(date) |>
  dplyr::select(date,
         # variables to be summed across all counties in St. Louis
         confirmed, deaths, recovered, hosp, icu, vent,
         population,
         # other variables that can be included in the analysis later
         stringency_index) |>
  dplyr::summarise_at(
    dplyr::vars(confirmed, deaths, recovered, hosp, icu, vent, population),
    .funs = sum, na.rm = T
    ) |> dplyr::ungroup()

unlink(temp) # remove temp file
```

Visualizing the TS Data

```
st_louis_agg_tbl |>
  tidyr::pivot_longer(cols = c(confirmed, deaths, icu),
                      names_to = 'statistic') |>
  ggplot2::ggplot(
    ggplot2::aes(x = date, y = value)
  ) +
  ggplot2::geom_line() +
  ggplot2::facet_wrap(~ statistic, scales = 'free_y', ncol = 1) +
  ggplot2::theme_bw() +
  ggplot2::scale_x_date(breaks = scales::pretty_breaks(n = 6)) +
  ggplot2::scale_y_continuous(labels = scales::comma) +
  ggplot2::labs(title = 'Plots of the Time-Series of Potential Variables for St. Louis MSA',
       caption = 'Based on data aggregated from the COVID19 DataHub',
       x = 'Date',
       y = NULL)
```

Visualizing the TS Data



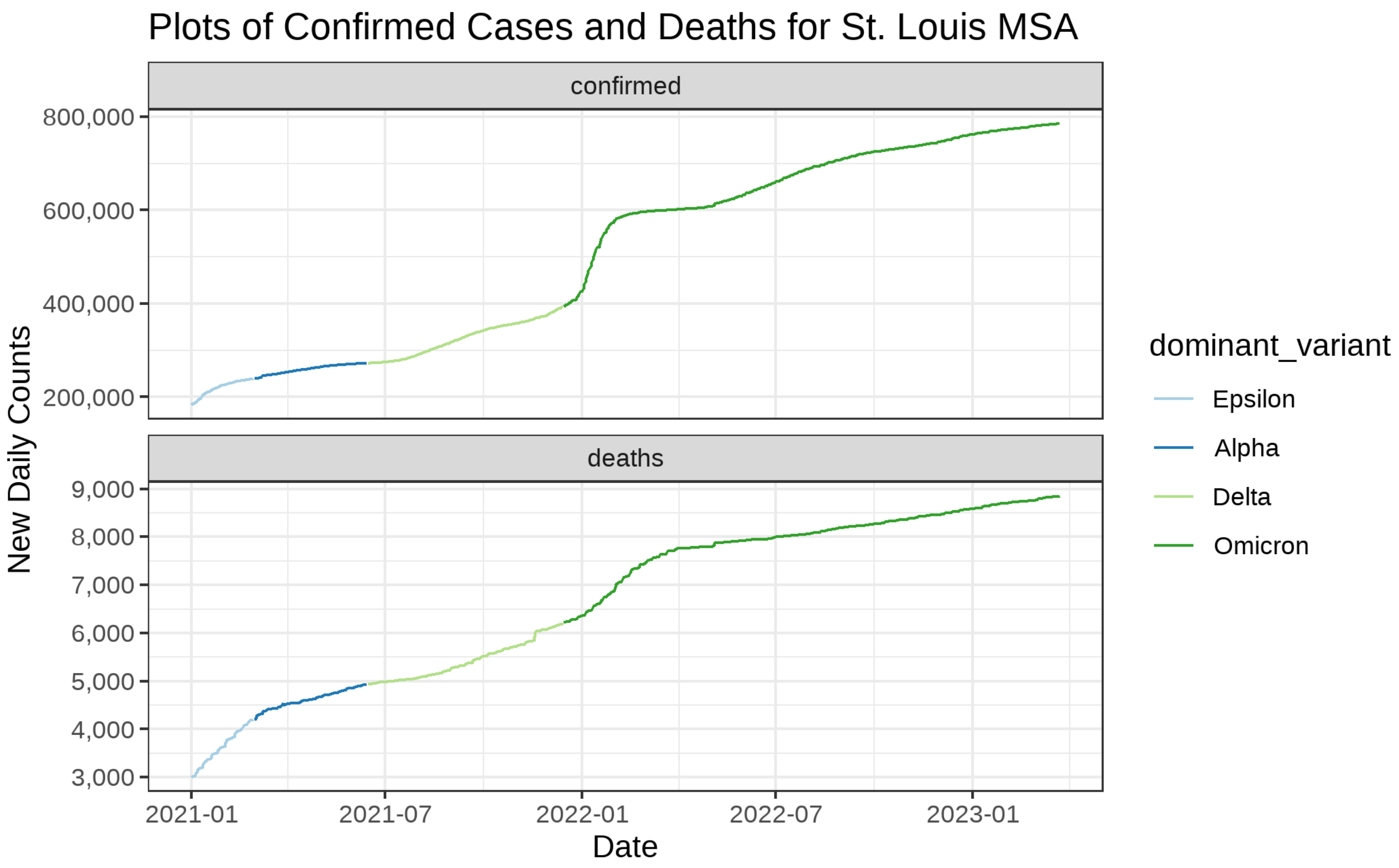
Updated Time Series

```
st_louis_agg_tbl =
  st_louis_agg_tbl |>
  # removing anomalies
  dplyr::filter(
    date >= lubridate::ymd('2021-01-01') &
    date <= lubridate::ymd('2023-03-23')) |>
  # creating potential predictors
  dplyr::mutate(
    # based on https://www.nytimes.com/interactive/2021/health/coronavirus-variant-tracker.html
    dominant_variant =
      dplyr::case_when(
        date < lubridate::ymd('2021-03-01') ~ 'Epsilon',
        date < lubridate::ymd('2021-06-15') ~ 'Alpha',
        date < lubridate::ymd('2021-12-15') ~ 'Delta',
        date >= lubridate::ymd('2021-12-15') ~ 'Omicron'
      ) |> forcats::as_factor(),
    # creating a list of special holidays
         holidays =
      dplyr::if_else(date %in% tidyquant::HOLIDAY_SEQUENCE(start_date = min(date),
                                                           end_date = max(date),
                                                           calendar = 'NYSE'),
              true = 'yes',
              false = 'no') |> forcats::as_factor()
           ) |>
  tidyr::drop_na()
```

Visualizing the Updated TS

```r
st_louis_agg_tbl |>
  tidyr::pivot_longer(cols = c(confirmed, deaths),
                      names_to = 'statistic') |>
  ggplot2::ggplot(
    ggplot2::aes(x = date, y = value, color = dominant_variant)
  ) +
  ggplot2::geom_line() +
  ggplot2::facet_wrap(~ statistic, scales = 'free_y', ncol = 1) +
  ggplot2::theme_bw() +
  ggplot2::scale_x_date(breaks = scales::pretty_breaks(n = 6)) +
  scale_y_continuous(labels = scales::comma) +
  ggplot2::scale_color_brewer(palette = 'Paired') +
  ggplot2::labs(x = 'Date',
       y = 'New Daily Counts',
       title = 'Plots of Confirmed Cases and Deaths for St. Louis MSA')
```

Visualizing the Updated TS



Creating time splits for Training and Validation

```
splits = rsample::initial_time_split(st_louis_agg_tbl, prop = 0.9)

print(splits)

paste('The starting and ending dates for training are',
      splits$data[splits$in_id, 'date'] |> head(n=1) |> dplyr::pull(),
      'and',
      splits$data[splits$in_id, 'date'] |> tail(n=1) |> dplyr::pull(),
      'respectively. For the holdout data, the starting and trainig dates are',
      splits$data[splits$out_id, 'date'] |> head(n=1) |> dplyr::pull(),
      'and',
      splits$data[splits$out_id, 'date'] |> tail(n=1) |> dplyr::pull())
```

```
## <Training/Testing/Total>
## <730/82/812>
## [1] "The starting and ending dates for training are 2021-01-01 and 2022-12-31 respectivel
```

Training Different Time-Series Models

I have quickly trained the following three models:

- A univariate Auto ARIMA model with no xreg
- An Auto ARIMA model with confirmed, holidays (NYSE holidays) and dominant variant as our xreg
- The Prophet Model, originally developed by Facebook. See the Forecasting at Scale Paper for more details.

Source: I have capitalized on the modeltime package vignette to quickly run these models. Note that the `modeltime API` also allows for running other machine learning models for time-series applications.

auto.arima() with no xreg

```r
library(modeltime)
# a univariate ARIMA model using “Auto Arima” using arima_reg()
# using the modeltime pkg this will automatically pick the weekly seasonality
model_fit_arima =
  modeltime::arima_reg() |>
  parsnip::set_engine(engine = "auto_arima") |> # this requires library(modeltime)
  parsnip::fit(deaths ~ date, data =  rsample::training(splits) )
```

`auto.arima()` with xreg

```r
# ARIMA with xreg
model_fit_arima_xreg =
  modeltime::arima_reg() |>
  parsnip::set_engine(engine = "auto_arima") |>
  # confirmed, holidays and dominant variant as our xreg
  parsnip::fit(
    deaths ~ date + confirmed + holidays + dominant_variant,
      data = rsample::training(splits)
    )
```

The Prophet Model

```r
library(prophet)

model_fit_prophet =
  modeltime::prophet_reg() |>
  parsnip::set_engine(engine = "prophet") |>
  parsnip::fit(deaths ~ date + confirmed + holidays + dominant_variant,
      data = rsample::training(splits) )
```

Model Table

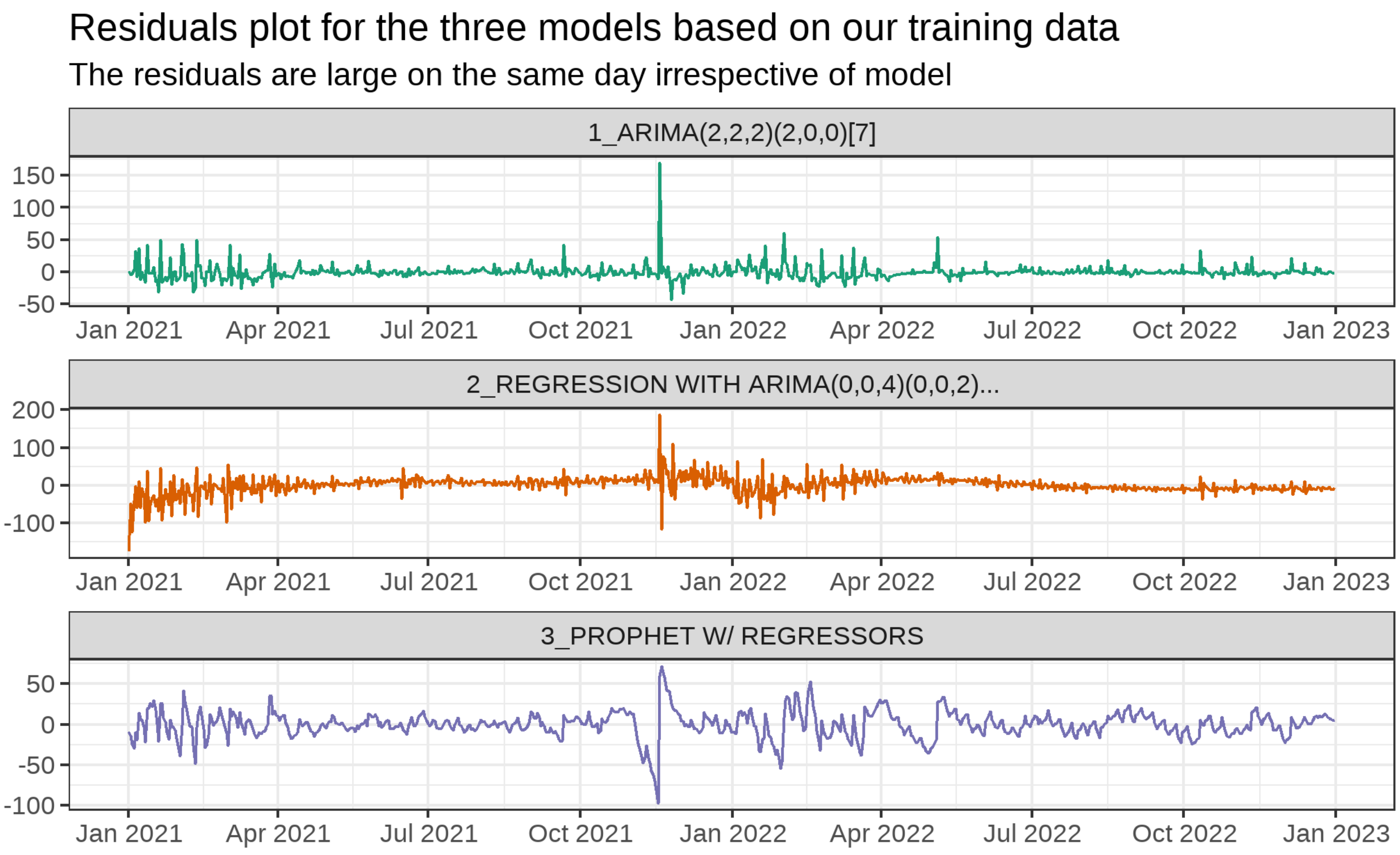
```
models_tbl =
  modeltime::modeltime_table(
    model_fit_arima,
    model_fit_arima_xreg,
    model_fit_prophet
)

models_tbl
```

```
## # Modeltime Table
## # A tibble: 3 × 3
##   .model_id .model   .model_desc
##       <int> <list>   <chr>
## 1         1 <fit[+]> ARIMA(2,2,2)(2,0,0)[7]
## 2         2 <fit[+]> REGRESSION WITH ARIMA(0,0,4)(0,0,2)[7] ERRORS
## 3         3 <fit[+]> PROPHET W/ REGRESSORS
```

Training Performance: Residuals

```r
models_tbl |>
    modeltime::modeltime_calibrate(new_data = rsample::training(splits)) |>
    modeltime::modeltime_residuals() |>
    modeltime::plot_modeltime_residuals(.interactive = FALSE,
                                         .type = 'timeplot') +
  ggplot2::scale_x_date(breaks = scales::pretty_breaks(12)) +
  ggplot2::scale_color_brewer(palette = 'Dark2') +
  ggplot2::facet_wrap(~ .model_desc, ncol = 1, scales = 'free') +
  ggplot2::theme_bw() +
  ggplot2::theme(legend.position = 'none') +
  ggplot2::labs(
    title = 'Residuals plot for the three models based on our training data',
    subtitle = 'The residuals are large on the same day irrespective of model')
```

Training Performance: Residuals



Statistical Tests for Residuals

```
models_tbl |>
  modeltime::modeltime_calibrate(new_data = rsample::training(splits)) |>
  modeltime::modeltime_residuals() |>
  modeltime::modeltime_residuals_test()
```

```
## # A tibble: 3 × 6
##   .model_id .model_desc              shapiro_wilk box_pierce ljung_box durbin_watson
##       <int> <chr>                           <dbl>      <dbl>     <dbl>         <dbl>
## 1         1 ARIMA(2,2,2)(2,0,0)…         3.28e-35      0.846     0.846         2.01
## 2         2 REGRESSION WITH ARI…         6.57e-25      0.956     0.956         1.93
## 3         3 PROPHET W/ REGRESSO…         4.54e-16      0         0             0.456
```

Training Performance

```
models_tbl |>
    modeltime::modeltime_calibrate(new_data = rsample::training(splits)) |>
    modeltime::modeltime_accuracy() |>
    modeltime::table_modeltime_accuracy()
```

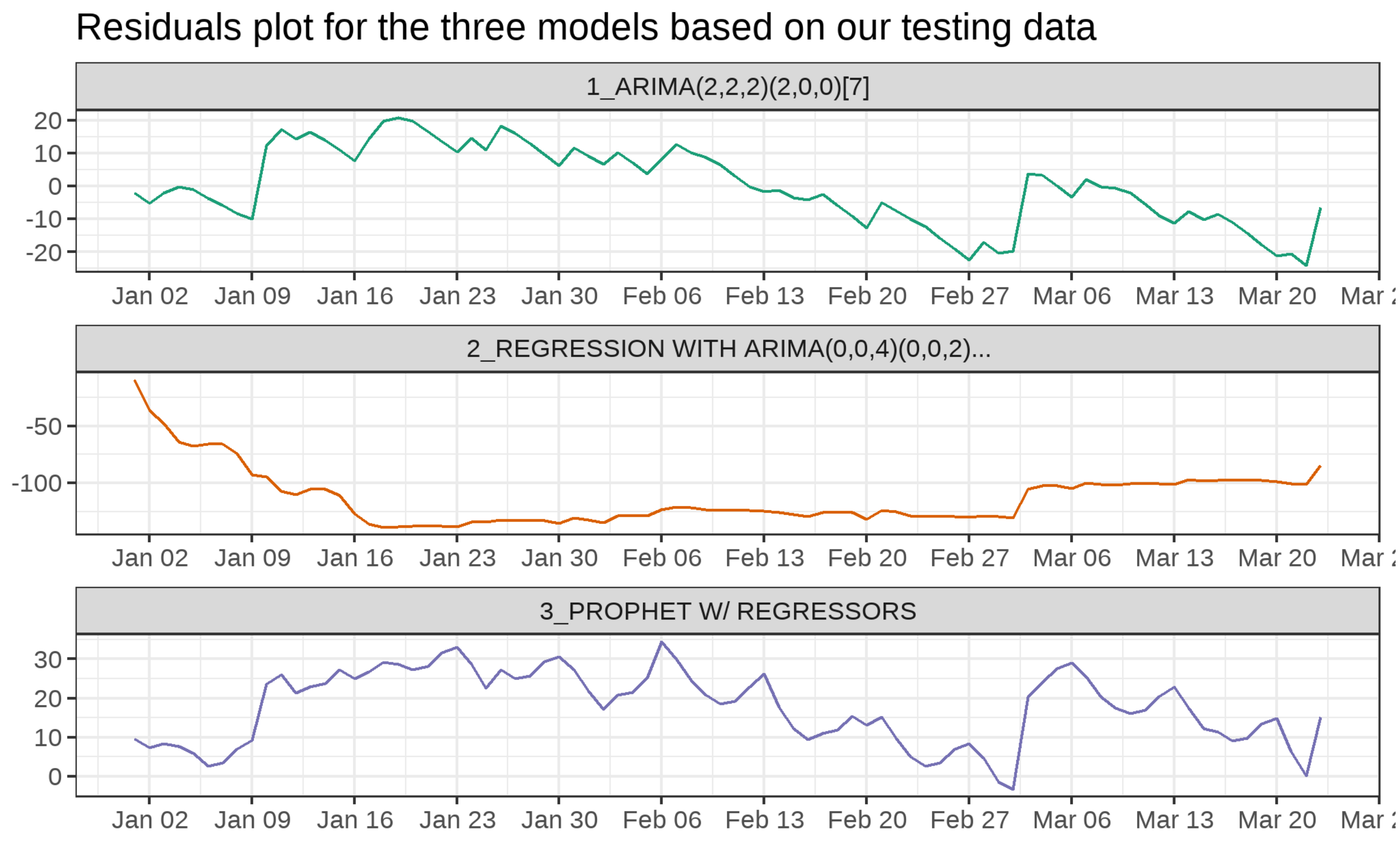
Search

.model_id	.model_desc	.type	mae	mape	mase	smape	rmse	rsq
1	ARIMA(2,2,2)(2,0,0)[7]	Fitted	6.28	0.11	0.81	0.11	11.66	1
2	REGRESSION WITH ARIMA(0,0,4)(0,0,2)[7] ERRORS	Fitted	16.14	0.3	2.08	0.3	24.73	1
3	PROPHET W/ REGRESSORS	Fitted	11.44	0.19	1.48	0.19	16.27	1

Training Performance: Residuals

```
models_tbl |>
    modeltime::modeltime_calibrate(new_data = rsample::testing(splits)) |>
    modeltime::modeltime_residuals() |>
    modeltime::plot_modeltime_residuals(.interactive = FALSE,
                                         .type = 'timeplot') +
  ggplot2::scale_x_date(breaks = scales::pretty_breaks(12)) +
  ggplot2::scale_color_brewer(palette = 'Dark2') +
  ggplot2::facet_wrap(~ .model_desc, ncol = 1, scales = 'free') +
  ggplot2::theme_bw() +
  ggplot2::theme(legend.position = 'none') +
  ggplot2::labs(
    title = 'Residuals plot for the three models based on our testing data')
```

Testing Performance: Residuals



Testing Peformance

```
models_tbl |>
    modeltime::modeltime_calibrate(new_data = rsample::testing(splits)) |>
    modeltime::modeltime_accuracy() |>
    modeltime::table_modeltime_accuracy()
```

Search

.model_id	.model_desc	.type	mae	mape	mase	smape	rmse	rsq
1	ARIMA(2,2,2)(2,0,0)[7]	Test	9.82	0.11	2.89	0.11	11.66	0.98
2	REGRESSION WITH ARIMA(0,0,4)(0,0,2)[7] ERRORS	Test	112.53	1.29	33.15	1.28	115.24	0.91
3	PROPHET W/ REGRESSORS	Test	17.77	0.2	5.23	0.2	19.91	0.99

Recap

Summary of Main Points

By now, you should be able to do the following:

- Explain how ML, and other advanced models, can be applied to TS data (given that we will be introducing this for **20 minutes** prior to answering questions pertaining to your final exam, this will be a very quick demo).

Things to Do to Prepare for the Final Exam

- Go through the slides, examples and make sure you have a good understanding of what we have covered.
- **Exam Setup:**
  - Q1 and Q2 interpretation of regression coefficients
  - Q3 interpretation of a residuals plot (`resPlot()`)
  - Q4 and Q5 interpretation of `tslm()` outputs
  - Q6 Interpretation of a `lm()` or `tslm()` model summary
  - Q7-Q8 interpretation of ARIMA with xreg
  - Q9-Q22 interpretations of which models to fit, autocorrelation, etc based on a plot of a time-series and its ACF
  - Q23-Q32 conceptual multiple choice questions