DTS303TC ASSESSMENT 2

Jiayuan Zhu

10/24/2022

Contents

PART A: Data Exploration

Set up

```
library(readr)
library(zoo, warn.conflicts = F)
library(tidymodels, warn.conflicts = F)
```

```
## -- Attaching packages -------------------------------------- tidymodels 1.0.0 --

## v broom        1.0.1      v recipes      1.0.1
## v dials        1.0.0      v rsample      1.1.0
## v dplyr        1.0.10     v tibble       3.1.8
## v ggplot2      3.3.6      v tidyr        1.2.1
## v infer        1.0.3      v tune         1.0.1
## v modeldata    1.0.1      v workflows    1.1.0
## v parsnip      1.0.2      v workflowsets 1.0.0
## v purrr        0.3.4      v yardstick    1.1.0

## -- Conflicts ----------------------------------------- tidymodels_conflicts() --
## x purrr::discard()  masks scales::discard()
## x dplyr::filter()   masks stats::filter()
## x dplyr::lag()      masks stats::lag()
## x yardstick::spec() masks readr::spec()
## x recipes::step()   masks stats::step()
## * Use tidymodels_prefer() to resolve common conflicts.
```

```
library(stringr, warn.conflicts = F)
library(modeltime)
library(timetk)
library(forecast, warn.conflicts = F)
```

```
## Registered S3 method overwritten by 'quantmod':
##   method            from
##   as.zoo.data.frame zoo
```

```
library(lubridate, warn.conflicts = F)
```

```
data_make <- read_csv("norway_new_car_sales_by_make.csv")
```

```
## Rows: 4377 Columns: 5
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (1): Make
## dbl (4): Year, Month, Quantity, Pct
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
data_model <- read_csv("norway_new_car_sales_by_model.csv")
```

```
## Rows: 2694 Columns: 6
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (2): Make, Model
## dbl (4): Year, Month, Quantity, Pct
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
data_month <- read_csv("norway_new_car_sales_by_month.csv")
```

```
## Rows: 121 Columns: 17
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## dbl (17): Year, Month, Quantity, Quantity_YoY, Import, Import_YoY, Used, Use...
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

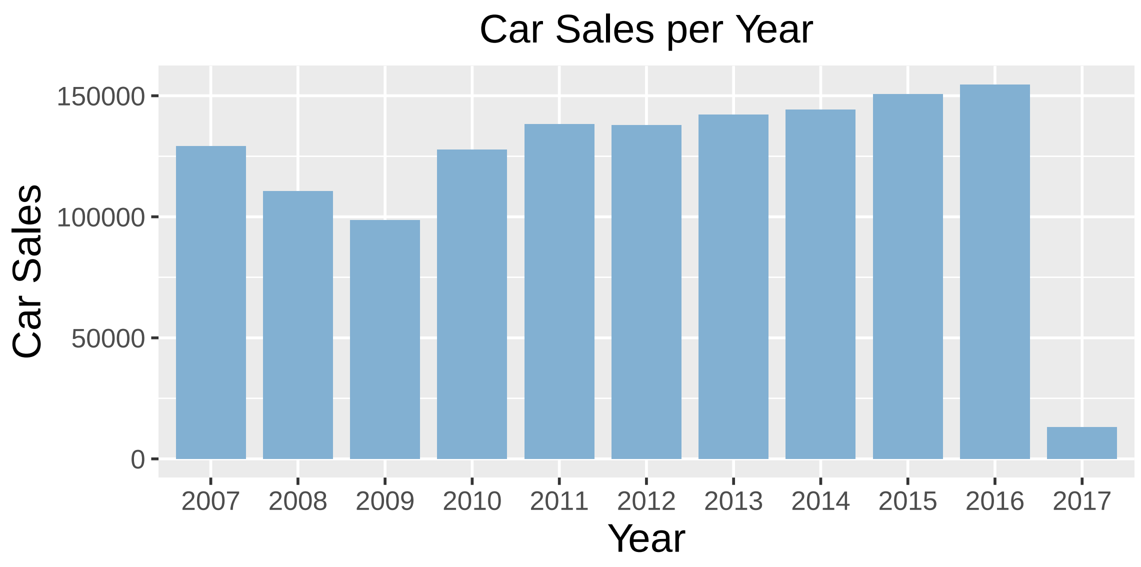
T1

Print year-wise total car sales and visualize the output (Hint: use bar chart for Year vs. total car sales).

```
# Year-wise total car sales
year_sales <- aggregate(Quantity ~ Year, sum, data = data_make)
year_sales
```

```
##    Year Quantity
## 1  2007   129195
## 2  2008   110617
## 3  2009    98675
## 4  2010   127754
## 5  2011   138345
## 6  2012   137967
## 7  2013   142151
## 8  2014   144202
## 9  2015   150686
## 10 2016   154603
## 11 2017    13055
```

```
ggplot(year_sales) +
  geom_bar(aes(x = as.character(Year), y = `Quantity`),
           fill = '#82B0D2',
           stat = 'identity',
           width = 0.8) +
  labs(x = 'Year',
       y = 'Car Sales',
       title = "Car Sales per Year") +
  theme(plot.title = element_text(hjust = 0.5, size = 15),
        axis.text.x = element_text(size = 10),
        axis.text.y = element_text(size = 10),
        axis.title = element_text(size = 15))
```



The output shows a gradual decline in sales from 2007 to 2009, with sales reaching a low point in 2009 and then rising until 2016.

T2

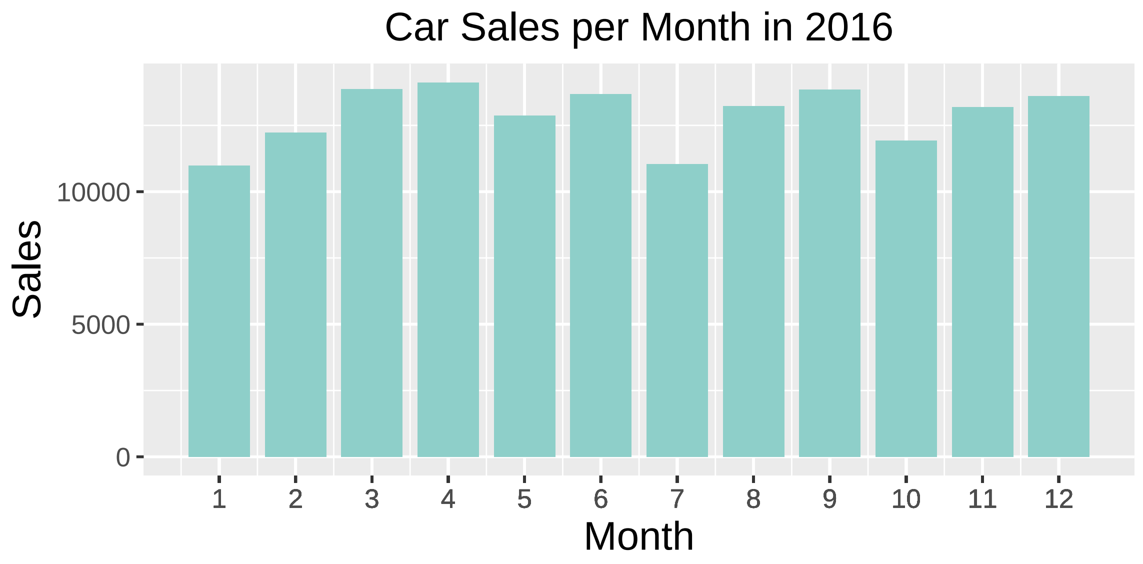
Print monthly total car sales and visualize for a specific year.

```
# Month-wise total car sales of 2016
M2016_sales <- aggregate(Quantity ~ Month, sum, data = data_month %>% filter(Year == 2016))
M2016_sales
```

```
##    Month Quantity
## 1      1    10991
## 2      2    12222
## 3      3    13875
## 4      4    14116
## 5      5    12864
## 6      6    13681
## 7      7    11040
## 8      8    13232
## 9      9    13854
## 10    10    11932
## 11    11    13194
## 12    12    13602
```

```
p2 <- ggplot(M2016_sales) +
  geom_bar(aes(x = Month, y = `Quantity`),
           fill = '#8ECFC9',
           stat = 'identity',
           width = 0.8) +
  labs(x = 'Month',
       y = 'Sales',
       title = "Car Sales per Month in 2016") +
  theme(plot.title = element_text(hjust = 0.5, size = 15),
        axis.text.x = element_text(size = 10),
        axis.text.y = element_text(size = 10),
        axis.title = element_text(size = 15))
```

```r
p2 + scale_x_continuous(labels = as.character(data_month$Month),
                        breaks = data_month$Month)
```



The output shows that the presence of car sales in 2016 is influenced by the month, with an upward trend in sales in each quarter.

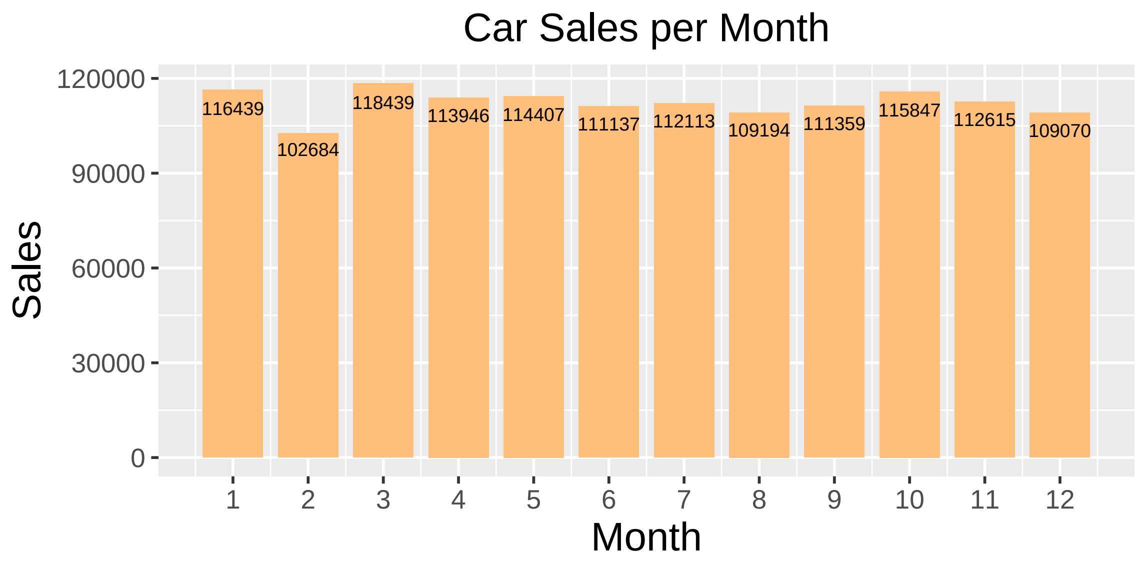
T3

Print monthly total car sales from 2007 to 2017 and visualize them to represent the month numbers (1 for Jan 2 for Feb) and total car sales value. (Hint: Use bar chart). Also, find the month for number of highest and lowest car sales.

```r
# Monthly total car sales from 2007 to 2017
month_sales <- aggregate(Quantity ~ Month, sum, data = data_month)
month_sales
```

```
##    Month Quantity
## 1      1   116439
## 2      2   102684
## 3      3   118439
## 4      4   113946
## 5      5   114407
## 6      6   111137
## 7      7   112113
## 8      8   109194
## 9      9   111359
## 10    10   115847
## 11    11   112615
## 12    12   109070
```

```r
p2 <- ggplot(month_sales) +
  geom_bar(aes(x = Month, y = Quantity),
           fill = '#FFBE7A',
           stat = "identity",
           width = 0.8) +
  geom_text(aes(y = Quantity, label = factor(Quantity), x = Month), color="black", size=2.5, position =
  labs(x = 'Month',
       y = 'Sales',
       title = "Car Sales per Month") +
  theme(plot.title = element_text(hjust = 0.5, size = 15),
        axis.text.x = element_text(size = 10),
        axis.text.y = element_text(size = 10),
        axis.title = element_text(size = 15))

p2 + scale_x_continuous(labels = as.character(month_sales$Month),
                        breaks = month_sales$Month)
```



```
# Highest
highest <- month_sales %>% filter(Quantity == max(Quantity))
highest$Month
```

```
## [1] 3
```

```
# Lowest
lowest <- month_sales %>% filter(Quantity == min(Quantity))
lowest$Month
```

```
## [1] 2
```

The result shows March has the highest sales while February has the lowest. Overall, however, sales did not vary significantly with the month.

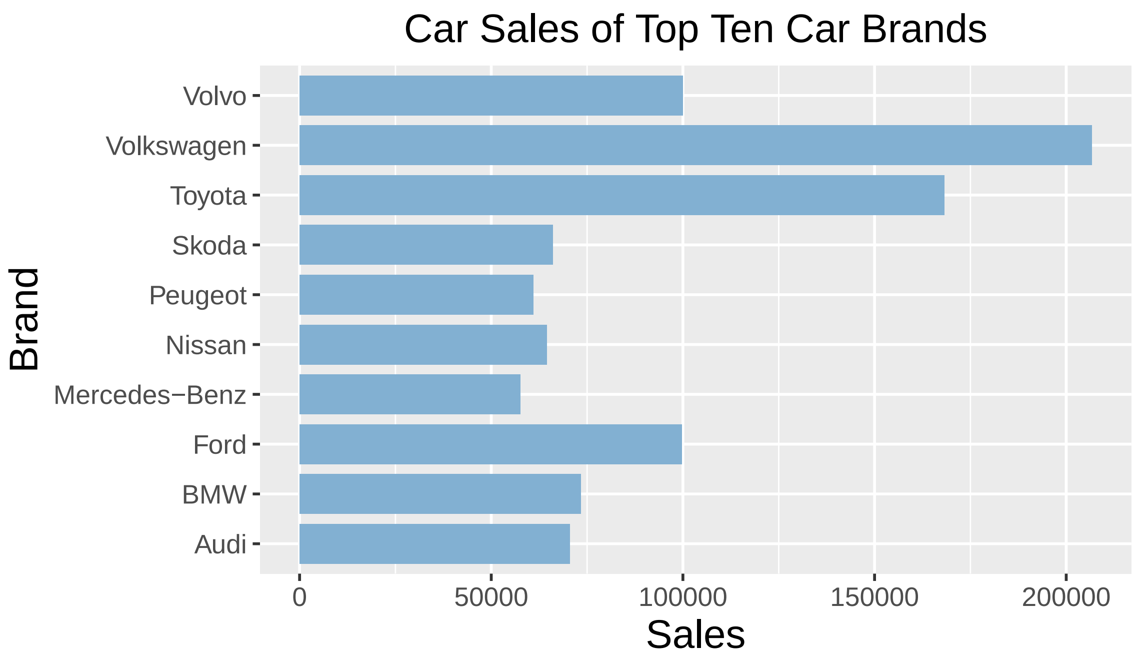
T4

Calculate the total amount of the sales for each manufacturer from 2007 to 2017. Find the top 10 manufacturers based on the total sale and visualize the output. (Hint: Sort make-wise total car sales and visualize them using bar chart).

```
# The total amount of the sales for each manufacturer from 2007 to 2017
make_sales <- aggregate(Quantity ~ Make, sum, data = data_make)

# Select the top ten manufacturers in terms of sales
best_makes <- make_sales[order(make_sales$Quantity, decreasing = T),][1:10,]
best_makes$Make
```

```
##  [1] "Volkswagen"    "Toyota"        "Volvo"         "Ford"
##  [5] "BMW"           "Audi"          "Skoda"         "Nissan"
##  [9] "Peugeot"       "Mercedes-Benz"
```

```
make_sales <- make_sales[make_sales$Make%in% best_makes$Make,]

# Plot the sales of the top manufacturers
ggplot(make_sales) +
  geom_bar(aes(x = Make, y = Quantity),
           fill = '#82B0D2',
           stat = "identity",
           width = 0.8) +
  coord_flip() +
  labs(x = 'Brand',
       y = 'Sales',
       title = "Car Sales of Top Ten Car Brands") +
  theme(plot.title = element_text(hjust = 0.5, size = 15),
        axis.text.x = element_text(size = 10),
        axis.text.y = element_text(size = 10),
        axis.title = element_text(size = 15))
```



The outputs show the top 10 manufacturers and plot their sales. The figure shows that Volkswagen has the highest sales, more than twice the average sales of others. And Toyota is a close second.

T5

Rank top 10 car brands. Visualize the year-wise result using line graph.

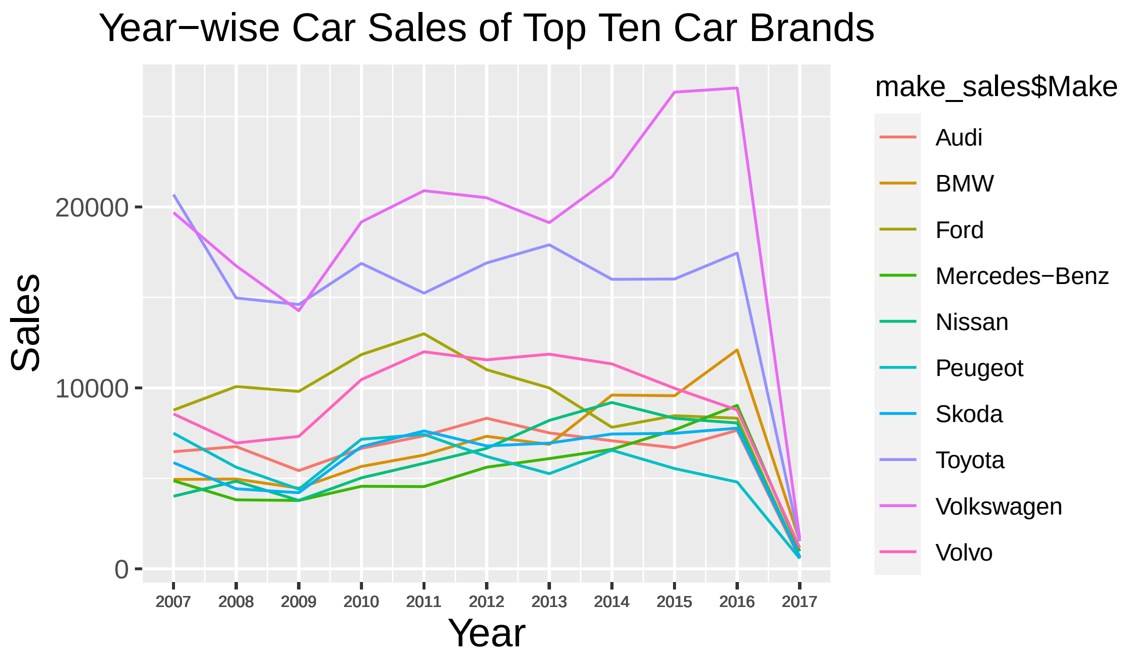
```
# Top 10 car brands
best_makes$Make
```

```
##  [1] "Volkswagen"    "Toyota"        "Volvo"         "Ford"
##  [5] "BMW"           "Audi"          "Skoda"         "Nissan"
##  [9] "Peugeot"       "Mercedes-Benz"
```

```
make_sales <- data_make %>%
  filter(Make %in% best_makes$Make) %>%
  group_by(Make, Year) %>%
  summarise(Sum = sum(Quantity))
```

```
## `summarise()` has grouped output by 'Make'. You can override using the
## `.groups` argument.
```

```
p5 <- ggplot(make_sales) +
  geom_line(aes(x = make_sales$Year, y = make_sales$Sum, color = make_sales$Make)) +
  labs(x = 'Year',
       y = 'Sales',
       title = "Year-wise Car Sales of Top Ten Car Brands") +
  theme(plot.title = element_text(hjust = 0.5, size = 15),
        axis.text.x = element_text(size = 6),
        axis.text.y = element_text(size = 10),
        axis.title = element_text(size = 15))

p5 + scale_x_continuous(labels = as.character(make_sales$Year),
                          breaks = make_sales$Year)
```

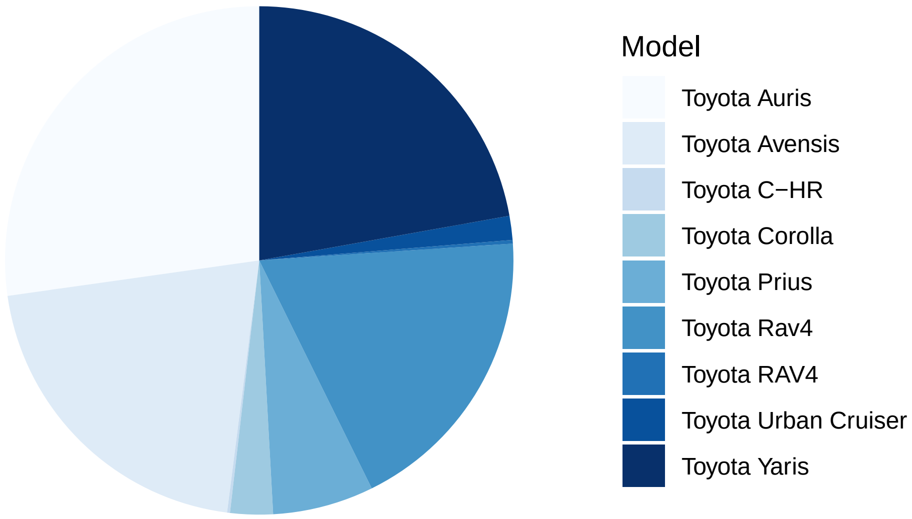


The output shows the sales trends for these ten brands are similar, with all reaching their lowest point in 2009 and their highest year in 2016.

T6

Draw pie chart for the sales of all the models of "Toyota" in year 2012.

```
# Select the Toyata data
toyota <- data_model %>% filter(grepl("Toyota", Model))
# Add up the quantity of each model
toyota_sales <- aggregate(Quantity ~ Model, sum, data = toyota)

ggplot(toyota_sales, aes(x = " ", y = Quantity, fill = Model)) +
  geom_bar(stat="identity", width=1) +
  coord_polar("y", start=0) +
  labs(x = NULL, y = NULL) +
  theme_classic() +
  theme(axis.line = element_blank(),
        axis.text = element_blank(),
        axis.ticks = element_blank()) +
  scale_fill_brewer(palette="Blues")
```



The Toyota Auris had the largest share of Toyota's mod sales, followed by the Toyota Avensis, Toyota, C-HR, and Toyota Corolla, with not much difference between the three. The remaining three mods account for a very small share.

T7

Find which model of each manufacturer has the highest sales in year 2015.

```
best_models <- data_model %>%
  filter(Year == 2015) %>%                            # select data of 2015
  group_by(Model) %>%                                 # group by models
  summarize(
    SumQuantity=sum(Quantity), Make=unique(Make)      # add up quantity of each model
    ) %>%
  group_by(Make) %>%                                  # group by makes
  mutate(
    MaxQuantityByMark = max(SumQuantity, na.rm = T)   # choose the max sum of quantity
  ) %>%
  filter(SumQuantity == MaxQuantityByMark)            # select the model with max quantity

# Output the best models and makes they belong to
best_models %>% select('Model', 'Make')
```

```
## # A tibble: 18 x 2
## # Groups:   Make [18]
##    Model                 Make
##    <chr>                 <chr>
##  1 Audi A3               Audi
##  2 BMW i3                BMW
##  3 Ford Mondeo           Ford
##  4 Hyundai i20           Hyundai
##  5 Kia Sportage           Kia
##  6 Mazda CX-5             Mazda
##  7 Mercedes-Benz B-klasse Mercedes-Benz
##  8 Mitsubishi Outlander   Mitsubishi
##  9 Nissan Leaf            Nissan
## 10 Opel Astra             Opel
## 11 Peugeot 308            Peugeot
## 12 Renault Zoe            Renault
## 13 Skoda Octavia          Skoda
## 14 Suzuki SX4             Suzuki
## 15 Tesla Model S          Tesla
## 16 Toyota Auris           Toyota
## 17 Volkswagen Golf        Volkswagen
## 18 Volvo V70              Volvo
```

T8

Find which model of each manufacturer has the highest sales during 2007 to 2017.

```
best_models <- data_model %>%
  group_by(Model) %>%                                 # group by models
  summarize(
    SumQuantity=sum(Quantity), Make=unique(Make)      # add up quantity of each model
    ) %>%
  group_by(Make) %>%                                  # group by makes
  mutate(
    MaxQuantityByMark = max(SumQuantity, na.rm = T)  # choose the max sum of quantity
  ) %>%
  filter(SumQuantity == MaxQuantityByMark)            # select the model with max quantity

# Output the models of each manufacturer has the highest sale sand makes they belong to
best_models %>%
  select('Model', 'Make')  %>%
  as_tibble() %>%
  print(n=Inf)
```

```
## # A tibble: 23 x 2
##    Model                          Make
##    <chr>                          <chr>
##  1 "Audi A4"                      "Audi"
##  2 "BMW i3"                       "BMW"
##  3 "Citroen C4 Aircross"          "Citroen"
##  4 "Ford Focus"                   "Ford"
##  5 "Honda CR-V"                   "Honda"
##  6 "Hyundai i30"                  "Hyundai"
##  7 "Kia Sportage"                 "Kia"
##  8 "Mazda CX-5"                   "Mazda"
##  9 "Mercedes-Benz B-klasse"       "Mercedes-Benz"
## 10 "Mitsubishi Outlander"         "Mitsubishi"
## 11 "Nissan Qashqai"               "Nissan"
## 12 "Opel Astra"                   "Opel"
## 13 "Peugeot 308"                  "Peugeot"
## 14 "Renault Zoe"                  "Renault"
## 15 "Saab 9-3"                     "Saab"
## 16 "Skoda Octavia"                "Skoda"
## 17 "Subaru XV"                  "Subaru"
## 18 "Suzuki SX4"                 "Suzuki"
## 19 "Tesla Model S"              "Tesla"
## 20 "Toyota Auris"               "Toyota"
## 21 "Volkswagen Golf"            "Volkswagen"
## 22 "Volvo V70"                  "Volvo"
## 23 "\xa0Mercedes-Benz A-klasse" "\xa0Mercedes-Benz"
```

T9

Find which model of each manufacturer has the lowest sales during 2007 to 2017.

```
best_models <- data_model %>%
  group_by(Model) %>%                                 # group by models
  summarize(
    SumQuantity=sum(Quantity), Make=unique(Make)      # add up quantity of each model
    ) %>%
  group_by(Make) %>%                                  # group by makes
  mutate(
    MinQuantityByMark = min(SumQuantity, na.rm = T)  # choose the min sum of quantity
  ) %>%
  filter(SumQuantity == MinQuantityByMark)            # select the model with min quantity

# Output the models of each manufacturer has the lowest sale sand makes they belong to
best_models %>% select('Model', 'Make')
```

```
## # A tibble: 23 x 2
## # Groups:   Make [23]
##    Model              Make
##    <chr>              <chr>
##  1 Audi A6            Audi
##  2 BMW 1-serie        BMW
##  3 Citroen C4 Aircross Citroen
##  4 Ford Kuga          Ford
##  5 Honda CR-V         Honda
##  6 Hyundai Tucson     Hyundai
##  7 Kia cee'd          Kia
##  8 Mazda CX-3         Mazda
##  9 Mercedes-Benz CLA  Mercedes-Benz
## 10 Mitsubishi ASX     Mitsubishi
## # ... with 13 more rows
```

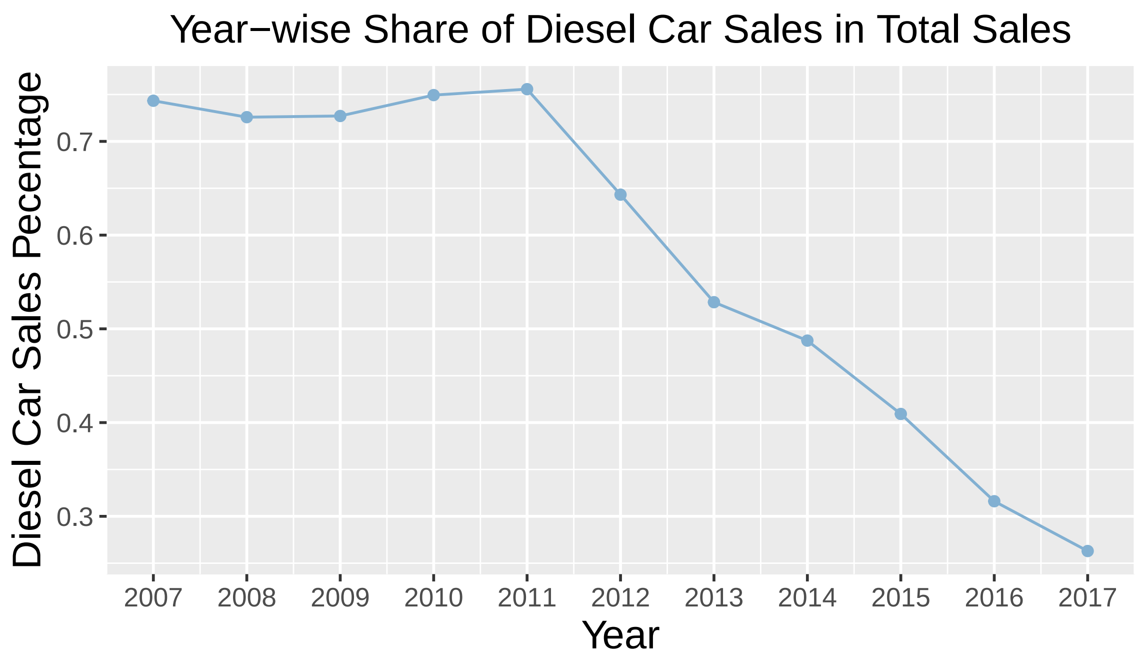
T12

Calculate year-wise share of diesel car sales in total sales.

```
# Compute the year-wise share of diesel car sales
diesel_percent <- data_month %>%
  group_by(Year) %>%
  summarize(DieselPercent=sum(Quantity_Diesel) / sum(Quantity))

diesel_percent
```

```
## # A tibble: 11 x 2
##     Year DieselPercent
##    <dbl>         <dbl>
## 1  2007          0.743
## 2  2008          0.726
## 3  2009          0.727
## 4  2010          0.749
## 5  2011          0.756
## 6  2012          0.643
## 7  2013          0.528
## 8  2014          0.487
## 9  2015          0.409
## 10 2016          0.316
## 11 2017          0.263
```

```
# Plot in line graph
p12 <- ggplot(diesel_percent) +
  geom_line(aes(x = Year, y = DieselPercent, group = 1), color='#82B0D2') +
  geom_point(x = diesel_percent$Year, y = diesel_percent$DieselPercent, color='#82B0D2') +
  labs(x = 'Year',
       y = 'Diesel Car Sales Pecentage',
       title = "Year-wise Share of Diesel Car Sales in Total Sales") +
  theme(plot.title = element_text(hjust = 0.5, size = 15),
        axis.text.x = element_text(size = 10),
        axis.text.y = element_text(size = 10),
        axis.title = element_text(size = 15))

p12 + scale_x_continuous(labels = as.character(diesel_percent$Year),
                         breaks = diesel_percent$Year)
```



The line graph shows that the market share of diesel vehicles did not change significantly from 2007 to 2011, remaining high at over 0.7. From 2012 onwards, the share of diesel vehicles plummeted to almost 0.1 per cent in 2017.

PART B - Machine Learning (Task 1)

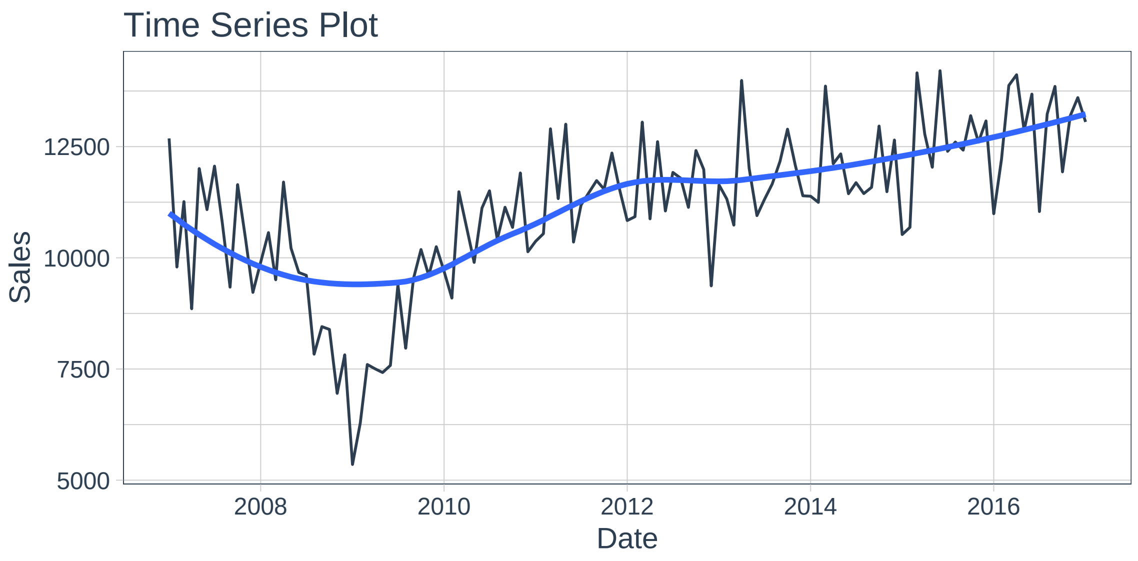
Predict (forecast) and visualize the car sales for all the months of 2020 using the month-wise car sales quantity from the Jan 2007 to Jan 2017.

1. Load data

```
# Merge column "Year" and "Month" to form a new column "Date"
data_month$Date <- as.Date(paste(data_month$Year, data_month$Month, "01", sep="-"), "%Y-%m-%d")

# Select column "Date" and "Quantity"
sales_data <- data_month %>%
  select("Date", "Quantity") %>%     # select
  set_names(c("date", "value"))      # rename

# Check the data
glimpse(sales_data)
```

```
## Rows: 121
## Columns: 2
## $ date  <date> 2007-01-01, 2007-02-01, 2007-03-01, 2007-04-01, 2007-05-01, 200~
## $ value <dbl> 12685, 9793, 11264, 8854, 12007, 11083, 12062, 10786, 9340, 1164~
```

```
# Plot the sales over time
sales_data %>%
  plot_time_series(date, value,
                   .title = "Time Series Plot",
                   .x_lab = "Date",
                   .y_lab = "Sales",
                   .interactive = F)
```

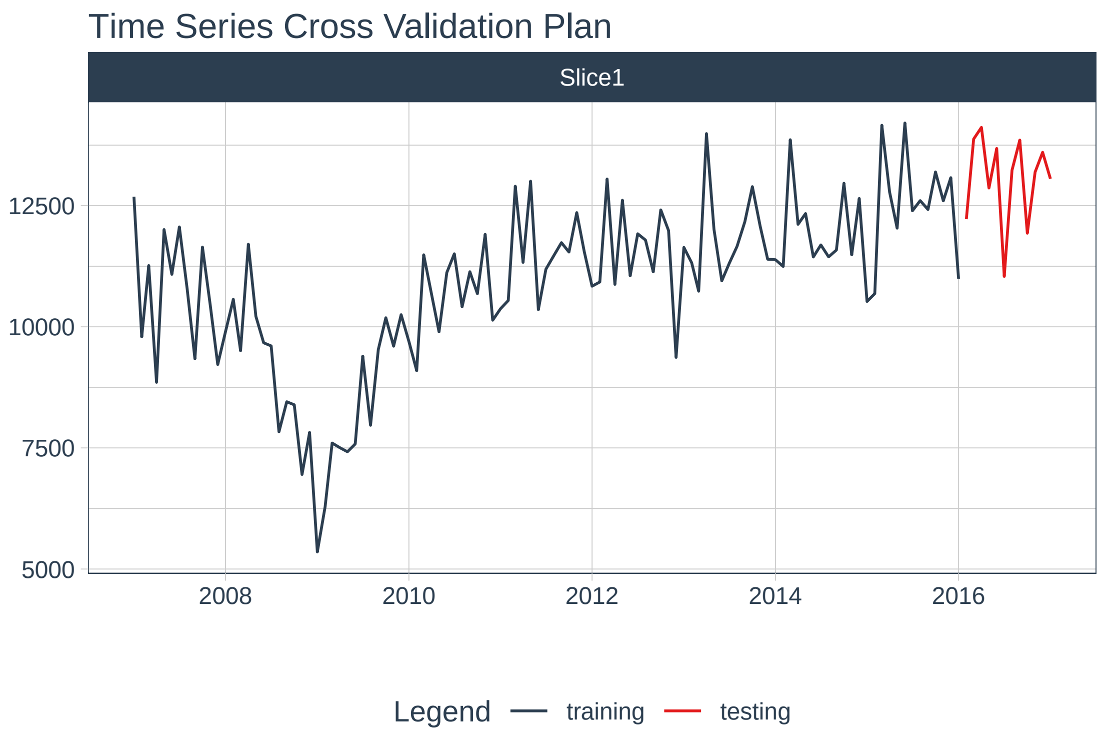


2. Split data

```
# Split the data set into training set and test set. Use the last year as the test set.
splits <- sales_data %>% time_series_split(assess = "1 year", cumulative = T)
```

```
## Using date_var: date
```

```r
# The test set is marked in read.
splits %>%
  tk_time_series_cv_plan() %>%
  plot_time_series_cv_plan(date, value, .interactive = F)
```



3. Models

1) Auto ARIMA

Auto-regressive Integrated Moving Average model predicts the time series by calculating the difference and moving.

```r
# Auto ARIMA
model_arima = arima_reg() %>%
  set_engine("auto_arima") %>%
  fit(value ~ date, training(splits))
```

```
## frequency = 12 observations per 1 year
```

```r
model_arima
```

```
## parsnip model object
##
## Series: outcome
## ARIMA(0,1,2)(1,0,0)[12]
##
## Coefficients:
##           ma1     ma2    sar1
##       -0.8007  0.2183  0.2446
## s.e.   0.1004  0.0996  0.1143
##
## sigma^2 = 1183668:  log likelihood = -907.57
## AIC=1823.14   AICc=1823.53   BIC=1833.87
```

2) Prophet

The prophet algorithm uses the additive model, which consists of four terms as below:

$$y(t) = g(t) + s(t) + h(t) + \epsilon(t)$$

where $g(t)$ describes a piecewise-linear trend (or "growth term"), $s(t)$ describes the various seasonal patterns, $h(t)$ captures the holiday effects, and $\epsilon(t)$ is a white noise error term.

```
# Prophet
model_prophet = prophet_reg(seasonality_yearly = TRUE,
                            seasonality_weekly = FALSE,
                            seasonality_daily = FALSE) %>%
  set_engine("prophet") %>%
  fit(value ~ date, training(splits))
model_prophet
```

```
## parsnip model object
##
## PROPHET Model
## - growth: 'linear'
## - n.changepoints: 25
## - changepoint.range: 0.8
## - yearly.seasonality: 'TRUE'
## - weekly.seasonality: 'FALSE'
## - daily.seasonality: 'FALSE'
## - seasonality.mode: 'additive'
## - changepoint.prior.scale: 0.05
## - seasonality.prior.scale: 10
## - holidays.prior.scale: 10
## - logistic_cap: NULL
## - logistic_floor: NULL
## - extra_regressors: 0
```

3) Elastic Net

Elastic Net is a linear regression model that uses the *L1* and *L2* priors as regularization matrices. This combination is used for sparse models with few non-zero weights, such as *Lasso*, while maintaining the regularization properties of *Ridge*.

```
# Prepocess
recipe_spec = recipe(value ~ date, training(splits)) %>%
  step_timeseries_signature(date) %>%
  step_fourier(date, period = 12, K =5) %>%
  step_dummy(all_nominal())

recipe_spec %>% prep() %>% juice()
```

```
## # A tibble: 109 x 54
##    date       value date_index~1 date_~2 date_~3 date_~4 date_~5 date_~6 date_~7
##    <date>     <dbl>        <dbl>   <int>   <int>   <int>   <int>   <int>   <int>
##  1 2007-01-01 12685   1167609600    2007    2007       1       1       1       0
##  2 2007-02-01  9793   1170288000    2007    2007       1       1       2       1
##  3 2007-03-01 11264   1172707200    2007    2007       1       1       3       2
##  4 2007-04-01  8854   1175385600    2007    2007       1       2       4       3
##  5 2007-05-01 12007   1177977600    2007    2007       1       2       5       4
##  6 2007-06-01 11083   1180656000    2007    2007       1       2       6       5
##  7 2007-07-01 12062   1183248000    2007    2007       2       3       7       6
##  8 2007-08-01 10786   1185926400    2007    2007       2       3       8       7
##  9 2007-09-01  9340   1188604800    2007    2007       2       3       9       8
## 10 2007-10-01 11646   1191196800    2007    2007       2       4      10       9
## # ... with 99 more rows, 45 more variables: date_day <int>, date_hour <int>,
## #   date_minute <int>, date_second <int>, date_hour12 <int>, date_am.pm <int>,
## #   date_wday <int>, date_wday.xts <int>, date_mday <int>, date_qday <int>,
## #   date_yday <int>, date_mweek <int>, date_week <int>, date_week.iso <int>,
## #   date_week2 <int>, date_week3 <int>, date_week4 <int>, date_mday7 <int>,
## #   date_sin12_K1 <dbl>, date_cos12_K1 <dbl>, date_sin12_K2 <dbl>,
## #   date_cos12_K2 <dbl>, date_sin12_K3 <dbl>, date_cos12_K3 <dbl>, ...
```

```
# Elastic Net
model_spec_glmnet <- linear_reg(penalty = 0.01, mixture = 0.5) %>%
  set_engine("glmnet")

workflow_fit_glmnet <- workflow() %>%
  add_model(model_spec_glmnet) %>%
  add_recipe(recipe_spec %>% step_rm(date)) %>%
  fit(training(splits))
```

4) Prophet Boost

Prophet Boost algorithm starts by modelling the univariate series using Prophet, followed by fitting the resulting features using the regressors provided by the pre-processing formulation and regressing the Prophet Residuals using the XGBoost model.

```
# Prophet Boost
model_spec_prophet_boost <- prophet_boost(seasonality_yearly = TRUE,
                                          seasonality_weekly = FALSE,
                                          seasonality_daily = FALSE) %>%
  set_engine("prophet_xgboost")

workflow_fit_prophet_boost <- workflow() %>%
  add_model(model_spec_prophet_boost) %>%
  add_recipe(recipe_spec) %>%
  fit(training(splits))
```

4. Model evaluation and model selection

Model evaluation and model selection using the *modeltime* package

```
model_table <- modeltime_table(
  model_arima,
  model_prophet,
  workflow_fit_glmnet,
  workflow_fit_prophet_boost
)

model_table
```

```
## # Modeltime Table
## # A tibble: 4 x 3
##   .model_id .model     .model_desc
##       <int> <list>     <chr>
## 1         1 <fit[+]>   ARIMA(0,1,2)(1,0,0)[12]
## 2         2 <fit[+]>   PROPHET
## 3         3 <workflow> GLMNET
## 4         4 <workflow> PROPHET W/ XGBOOST ERRORS
```

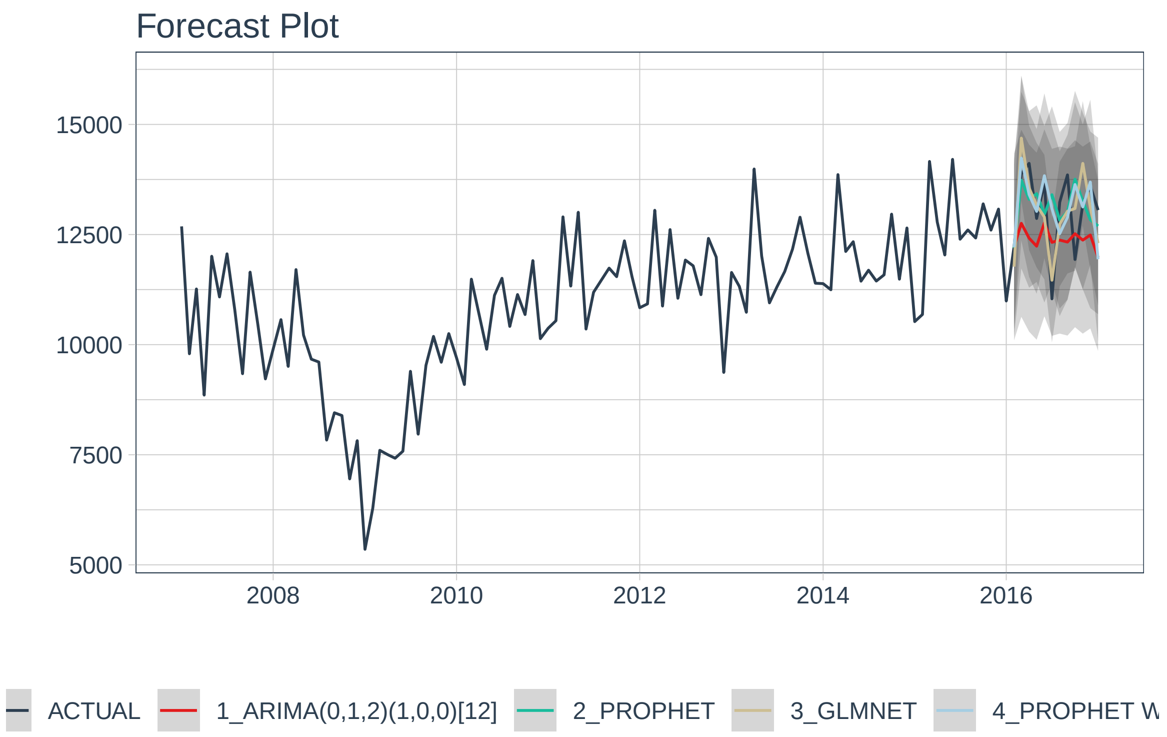
```
# Predict the test set
calibration_table = model_table %>%
  modeltime_calibrate(testing(splits))

# Demonstrate the performance of models on the test set through various criterions
calibration_table %>%
  modeltime_accuracy() %>%
  table_modeltime_accuracy(.interactive = FALSE)
```

Accuracy Table

.model_id	.model_desc	.type	mae	mape	mase	smape	rmse	rsq
1	ARIMA(0,1,2)(1,0,0)[12]	Test	968.97	7.36	0.79	7.57	1060.44	0.11
2	PROPHET	Test	740.91	5.95	0.60	5.74	1001.05	0.00
3	GLMNET	Test	664.30	5.10	0.54	5.08	706.03	0.45
4	PROPHET W/ XGBOOST ERRORS	Test	674.56	5.41	0.55	5.27	933.98	0.08

```
# Plot the prediction results of different models on the test set
calibration_table %>%
  modeltime_forecast(actual_data = sales_data) %>%
  plot_modeltime_forecast(.interactive = FALSE)
```

```
## Using '.calibration_data' to forecast.
```



The result shows that **ARIMA** and **Prophet** have bad performance on the test set. Thus. they are considered for removal in the inference.

5. Predict

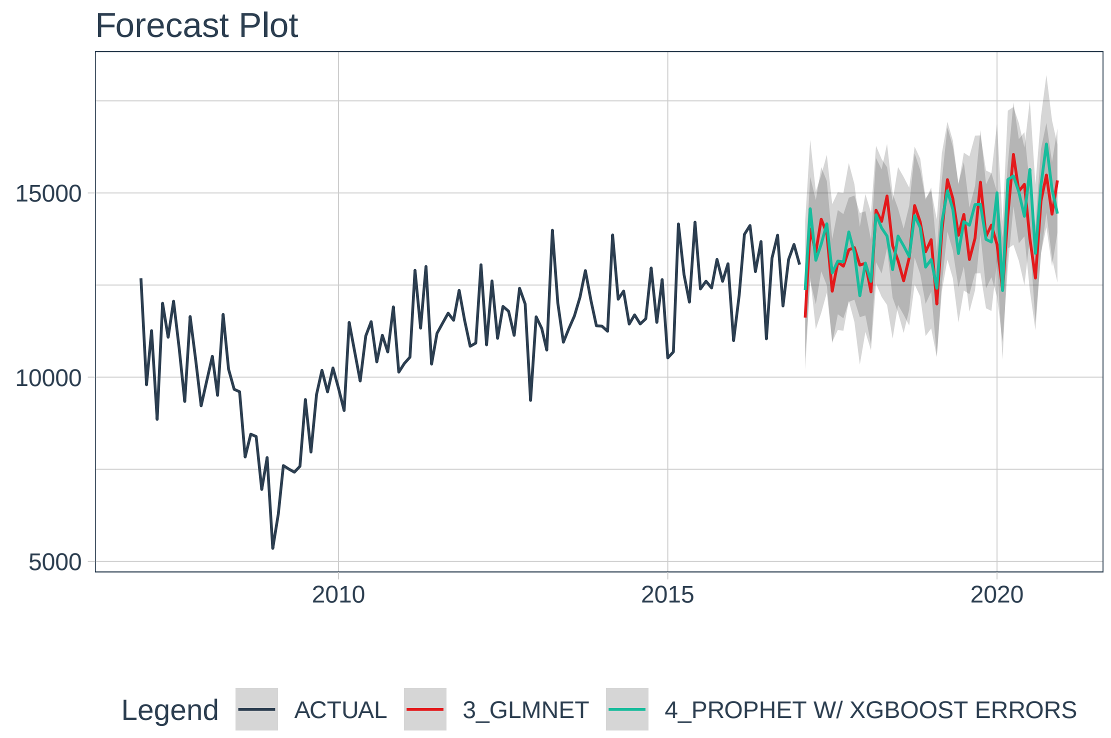
In order to better plot the change in sales, I decided to forecast the maximum date provided to the end of 2020 instead of just the 2020 sales.

```
# Calculate how many months are left until December 2020
diff_time = ceiling(as.numeric(difftime(as.Date("2020-12-01", "%Y-%m-%d"),
                                  max(sales_data$date),units="days")) / (365.25/12))
diff_time
```

```
## [1] 47
```

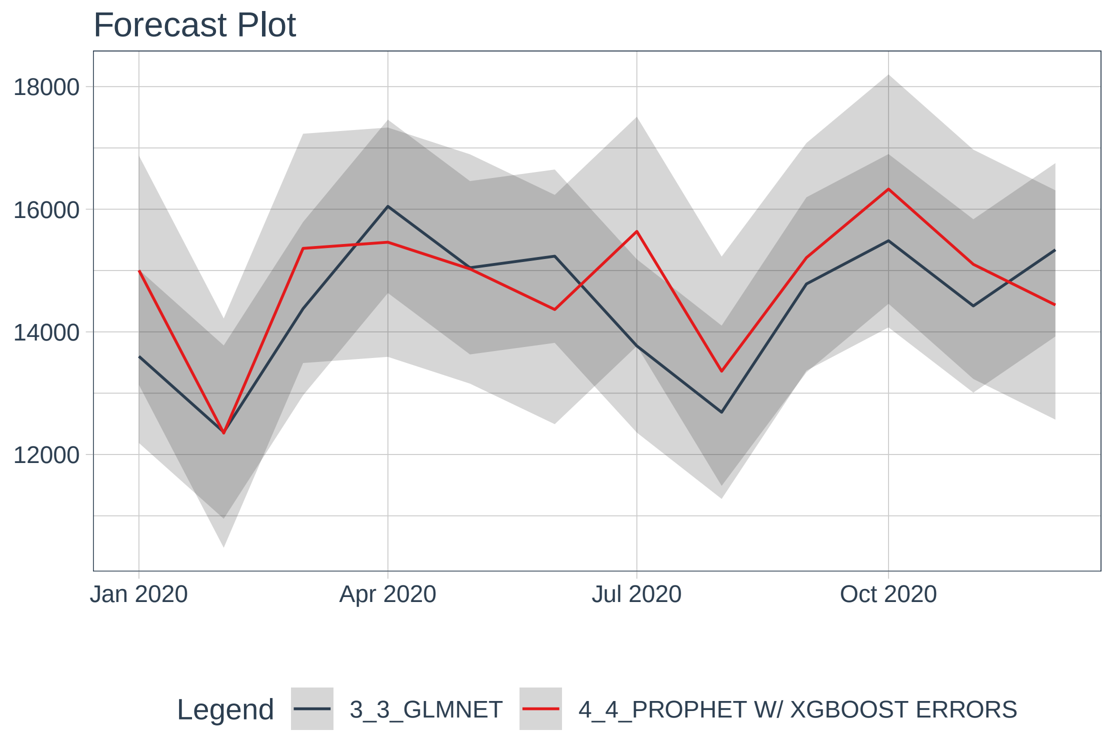
The result shows that it need to predict the sales for the next 47 months.

```
sales_all <- calibration_table %>%
  # Remove the models with low accuracy
  filter(.model_id != 1 & .model_id != 2) %>%
  # Refit and Forecast Forward
  modeltime_refit(sales_data) %>%
  modeltime_forecast(h = "47 months", actual_data = sales_data) %>%
  plot_modeltime_forecast(.interactive = FALSE)

sales_all
```



```
# Split the prediction of 2020
preds <- sales_all$data %>%
  filter(.key != "actual") %>%
  filter(.index >= as.Date("2020-01-01", "%Y-%m-%d"))

# Plot the prediction of 2020 of different models
```

```r
preds %>% plot_modeltime_forecast(.interactive = FALSE)
```



Output predictions of different model.

```r
# Elastic Net
GLMNet_preds <- subset(preds, .model_id == 3) %>%
  select('.index', '.value') %>%
  set_names(c("date", "value"))

GLMNet_preds
```

```
## # A tibble: 12 x 2
##    date        value
##    <date>      <dbl>
##  1 2020-01-01 13601.
##  2 2020-02-01 12365.
##  3 2020-03-01 14379.
##  4 2020-04-01 16047.
##  5 2020-05-01 15046.
##  6 2020-06-01 15234.
##  7 2020-07-01 13771.
##  8 2020-08-01 12689.
##  9 2020-09-01 14782.
## 10 2020-10-01 15486.
## 11 2020-11-01 14423.
## 12 2020-12-01 15339.
```

```r
# Prophet Boost
XGBoost_preds <- subset(preds, .model_id == 5) %>%
  select('.index', '.value') %>%
  set_names(c("date", "value"))
XGBoost_preds
```

```
## # A tibble: 0 x 2
## # ... with 2 variables: date <date>, value <dbl>
```

```r
# Ensemble
ensemble_preds <- aggregate(preds[c(".value")], preds[c(".index")], mean) %>%
  set_names(c("date", "value"))
ensemble_preds
```

```
##          date    value
## 1  2020-01-01 14302.59
## 2  2020-02-01 12357.22
## 3  2020-03-01 14870.18
## 4  2020-04-01 15754.73
## 5  2020-05-01 15036.07
## 6  2020-06-01 14799.28
## 7  2020-07-01 14704.77
## 8  2020-08-01 13023.58
## 9  2020-09-01 14996.32
## 10 2020-10-01 15907.51
## 11 2020-11-01 14762.36
## 12 2020-12-01 14888.62
```

6. Conclusion

Throughout the task, four models were selected, two Statistical approaches: **ARIMA** and **Prophet**, and two machine learning approaches: **Elastic Net** and **Prophet Boost**. Among those, **ARIMA** uses differences and fits time series, while **Prophet** takes into account seasonal factors and holiday effects in addition to growth term. Machine learning approaches, however, first pre-process the data, in particular the Fourier transform, to extract the features of the time series, and then fit these multi-dimensional features with a model. The **Elastic Net** used here is a linear regression model with special regularization, while **Prophet Boost** combines **Prophet** and **XGBoost**.

Then, the data set was divided into training set and test set, where the last year of the data set is treated as a test set. The results show that the two statistical approaches are relatively poor, especially **ARIMA**. this may be due to the fact that **Prophet** considers more elements, with seasonal factors and holiday factors, which **ARIMA** does not consider. The machine learning approaches, on the other hand, all performed better. It is possible that the Fourier transform extracts more hidden factors from the time series than just those considered by **Prophet**, so they learn more information from it and perform better on the test set. Of the two machine learning approaches, the results are similar, but **Elastic Net** is simpler and less time-consuming, so **Elastic Net** performs the best of the four on this data set.

PART C: Data Security

Data Compliance, Data Security, and Data Privacy

Data compliance, short for data privacy compliance, is the process of adhering to various regulations and standards to maintain the integrity and availability of regulated data (e.g. personal data, medical data) and/or sensitive data (e.g. customer lists), all of which are intended to help protect regulated and/or sensitive data from unauthorised use. Data compliance is the formal governance structure used to ensure that an organisation complies with the laws, regulations and standards governing its data. This process governs the ownership, organisation, storage and management of digital assets or data to prevent their loss, theft, misuse or destruction. Furthermore, there are some data protection compliance standards, such as SOC 2, CMMC, and ISO 27001 are formulated to protect the data compliance.

Data security refers to the protective measures taken to protect data from unauthorised access and to maintain the confidentiality, integrity and availability of data. Data security management refers to policies and procedures to protect the confidentiality, integrity and availability (CIA) of information and security systems. Data security management focuses on protecting data, such as personally identifiable information and business-critical intellectual property. For companies, for example, data security is important because it ensures business continuity. Without good security measures, an attacker could steal valuable information or cripple your system with a ransomware attack, bringing everything to a halt.

Privacy generally involves a person's ability to decide for themselves when, how and to what extent personal information about them is shared or communicated with others. This personal information may be a person's name, location, contact details or behaviour on the internet or in the real world. In the laws of many countries, privacy is considered a basic human right, and violating privacy can lead to legal sanctions. For people to be willing to participate online, they need to trust that their personal information will be handled with care. Organisations use privacy practices to show their customers and users that their personal data can be trusted. Data breaches can lead to the theft of valuable customer information, which can have a negative impact on the data owner. Hackers can use all this sensitive information to commit various crimes such as identity theft and credit card fraud.

Big Data Ecosystem

The Figure 1 shows the general framework of the Big Data Ecosystem - Hadoop Ecosystem. HDFS, MapReduce and HBase form the core of Hadoop. HDFS provides the distributed file system for MapReduce and HBase, Hadoop's proprietary NoSQL database, while MapReduce and YARN handle the pre-processing of the data. This allows the processed data to be used by Mahout for machine learning and Pig, Hive and Spark for distributed programming. In addition, Zookeeper provides service programming for the entire system, while Ambari provides system provisioning and the Hadoop database. Ambari provides system provisioning and Sqoop provides data entry for Hadoop Core.

Compliance, privacy and security are important for big data ecosystem. Specifically, Security tools need to work in three levels of data:

Level 1: Data Source. Big data sources come from a variety of sources and data types. In addition, the file system has a whole world of machine-generated data, including logs and sensors. The system must ensure that this data is secure during transport from the source to the platform.

Level 2: Stored Data. Protecting the stored data requires a sophisticated security toolset, including static encryption, strong user authentication, and intrusion and planning protection. You also need to run your security toolset on a distributed cluster platform with many servers and nodes.

Stage 3: Output data. To prevent theft of the output data, both the output data and the incoming data must be encrypted. It must also be ensured that the output to the end user does not contain controlled data.

Various measures for protecting data in Hadoop are described below.

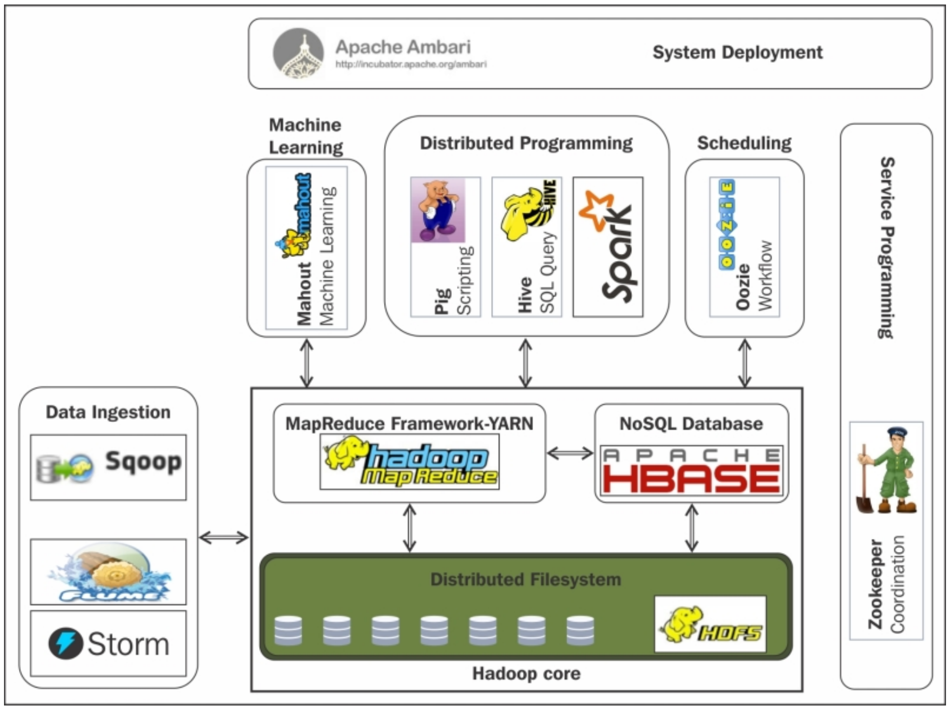


Figure 1: The Hadoop ecosystem

SSL

SSL is essential for protecting data and provides privacy, critical security and data integrity for information in the ecosystem.

SSL encrypts sensitive information The main reason for using SSL is to encrypt sensitive information sent over the Internet so that only the intended recipient can access it. This is important because information sent over the internet is passed from computer to computer to reach the destination server. If not encrypted with an SSL certificate, any computer between the user and the server can see your credit card number, username and password, and other confidential information. If an SSL certificate is used, this information can only be read by the server to which the user sends the data. This protects the data from hackers.

SSL provides authentication In addition to encryption, a proper SSL certificate also provides authentication. This means that the user can be sure that they are sending a message to the correct server and not to an imposter trying to steal it. This can only be avoided by purchasing an SSL certificate from a trusted SSL provider to send messages.

Ranger

Apache Ranger is a framework for enabling, monitoring and managing comprehensive data security across the Hadoop platform. It provides a comprehensive approach to securing Hadoop clusters. As shown in the Figure 2, Ranger consists of three components.

Ranger admin Portal: This is the central interface for security management. Here users can create and update policies in the database. In addition, this portal includes an audit server. This audit server sends the audit data collected from the plugins to the HDFS/relational database.

Ranger Plugins : The plugins pull policies from the central server and store them locally. When an end user requests access to the database, these plugins evaluate the user and grant access based on the validation results. In addition, this collects the data requested by the user and follows a separate thread to send it back to the audit server.

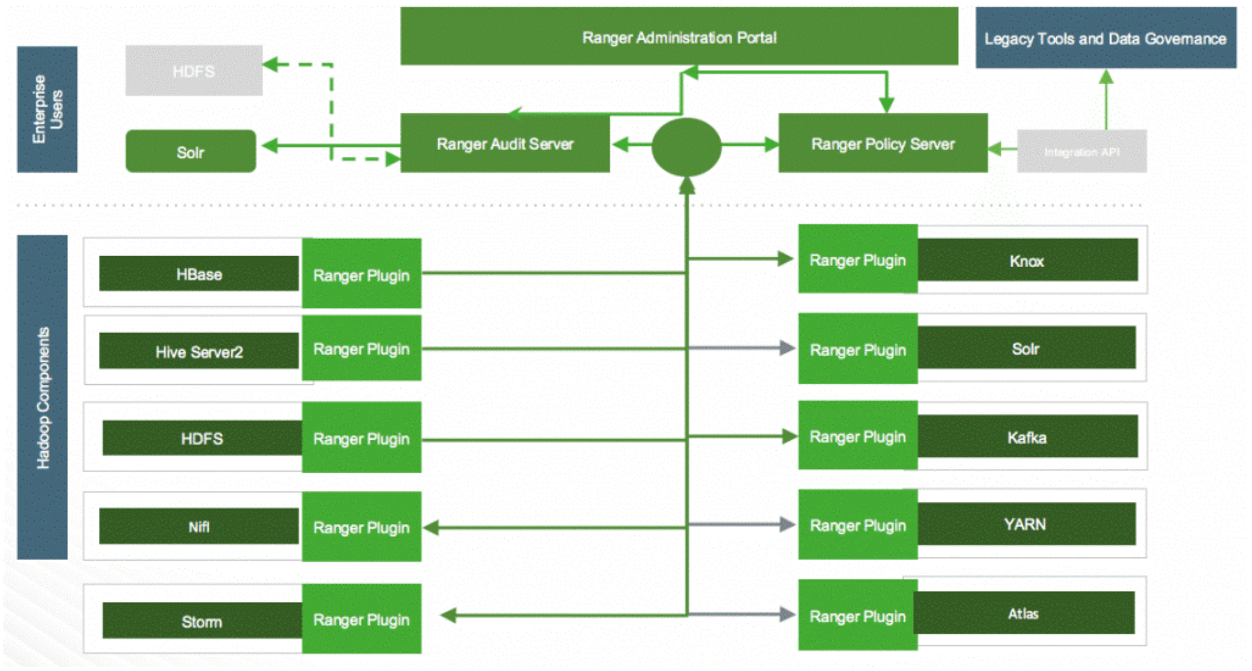


Figure 2: Apache Ranger in Hadoop platform

User group sync: provides user synchronisation.

Kerberos

Kerberos is a computer network authentication protocol that works on top of tickets, allowing nodes to communicate on non-secure networks to prove their identity in a secure way. Where users and servers rely on a third party, the Kerberos server, to authenticate each other.

As shown in the Figure 3, Kerberos servers are known as Key Distribution Centers (KDCs). A KDC consists of two main components, An Authentication Server (AS) and A Ticket Granting Server (TGS). Once per user logon session, a command is sent to the AS requesting a ticket-granting ticket. It performs initial authentication and uses a ticket-granting ticket encrypted with the Kerberos password of the user's delegate. Once per type of service, a request for a service-granting ticket is sent to the TGS, which generates a service-granting ticket based on the ticket-granting ticket. Based on the service-granting ticket, the client is able to access the server, which is also required to provide server authentication for authentication.

Encryption

HDFS implements transparent end-to-end encryption. Once configured, data read from and written to special HDFS directories is transparently encrypted and decrypted without requiring any changes to the user's application code. This encryption is also end-to-end, i.e. the data can only be encrypted and decrypted by the client, which increases the security of the data transmission. To achieve transparent encryption, we have encryption zones for HDFS, i.e. special directories whose contents are transparently encrypted when written and decrypted when read. Each encryption zone is associated with an encryption zone key, which is specified when the encryption zone is created. Each file in the encryption zone has its own unique data encryption key (DEK); HDFS never processes the DEK directly, only an encrypted data encryption key (EDEK). The client decrypts the EDEK and then uses the subsequent DEK to read and write the data; the HDFS data node sees only an encrypted stream of bytes.

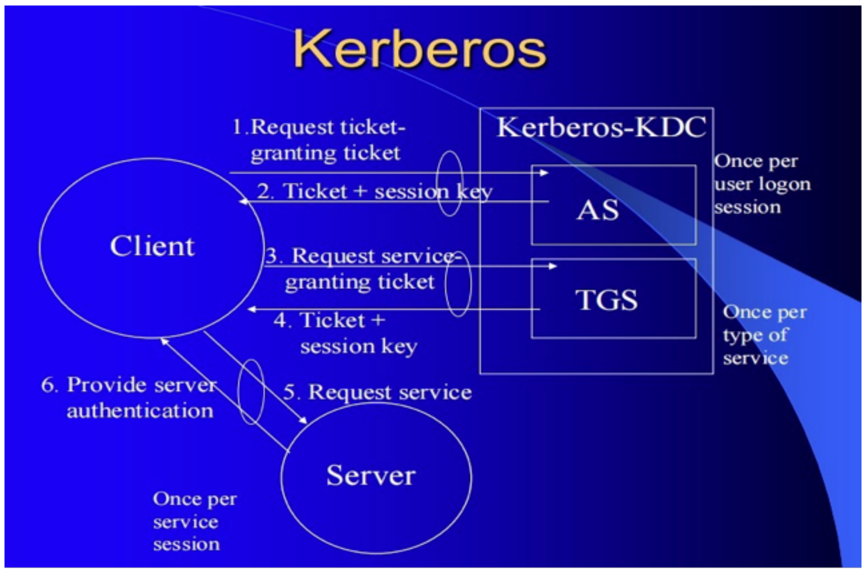


Figure 3: Kerberos