DA5020.A9.Hsiao-Yu.Peng

Hsiao-Yu Peng

2023-11-17

Q1

```r
data <- read.csv("~/Desktop/2023Fall_Syllabus/DA5020/module9/Air_Traffic_Passenger_Statistics(1).csv", 

glimpse(data)
```

```
## Rows: 46,670
## Columns: 12
## $ Activity.Period             <int> 200507, 200507, 200507, 200507, 200507, 20~
## $ Operating.Airline           <chr> "ATA Airlines", "ATA Airlines", "ATA Airli~
## $ Operating.Airline.IATA.Code <chr> "TZ", "TZ", "TZ", "AC", "AC", "CA", "CA", ~
## $ Published.Airline           <chr> "ATA Airlines", "ATA Airlines", "ATA Airli~
## $ Published.Airline.IATA.Code <chr> "TZ", "TZ", "TZ", "AC", "AC", "CA", "CA", ~
## $ GEO.Summary                 <chr> "Domestic", "Domestic", "Domestic", "Inter~
## $ GEO.Region                  <chr> "US", "US", "US", "Canada", "Canada", "Asi~
## $ Activity.Type.Code          <chr> "Deplaned", "Enplaned", "Thru / Transit", ~
## $ Price.Category.Code         <chr> "Low Fare", "Low Fare", "Low Fare", "Other~
## $ Terminal                    <chr> "Terminal 1", "Terminal 1", "Terminal 1", ~
## $ Boarding.Area               <chr> "B", "B", "B", "B", "B", "G", "G", "A", "A~
## $ Passenger.Count             <int> 27271, 29131, 5415, 35156, 34090, 6263, 55~
```

The dataset has 46,670 rows and 12 columns. Activity.Period column is integer type, but it's related to datetime. We'll preprocess it to year and month.

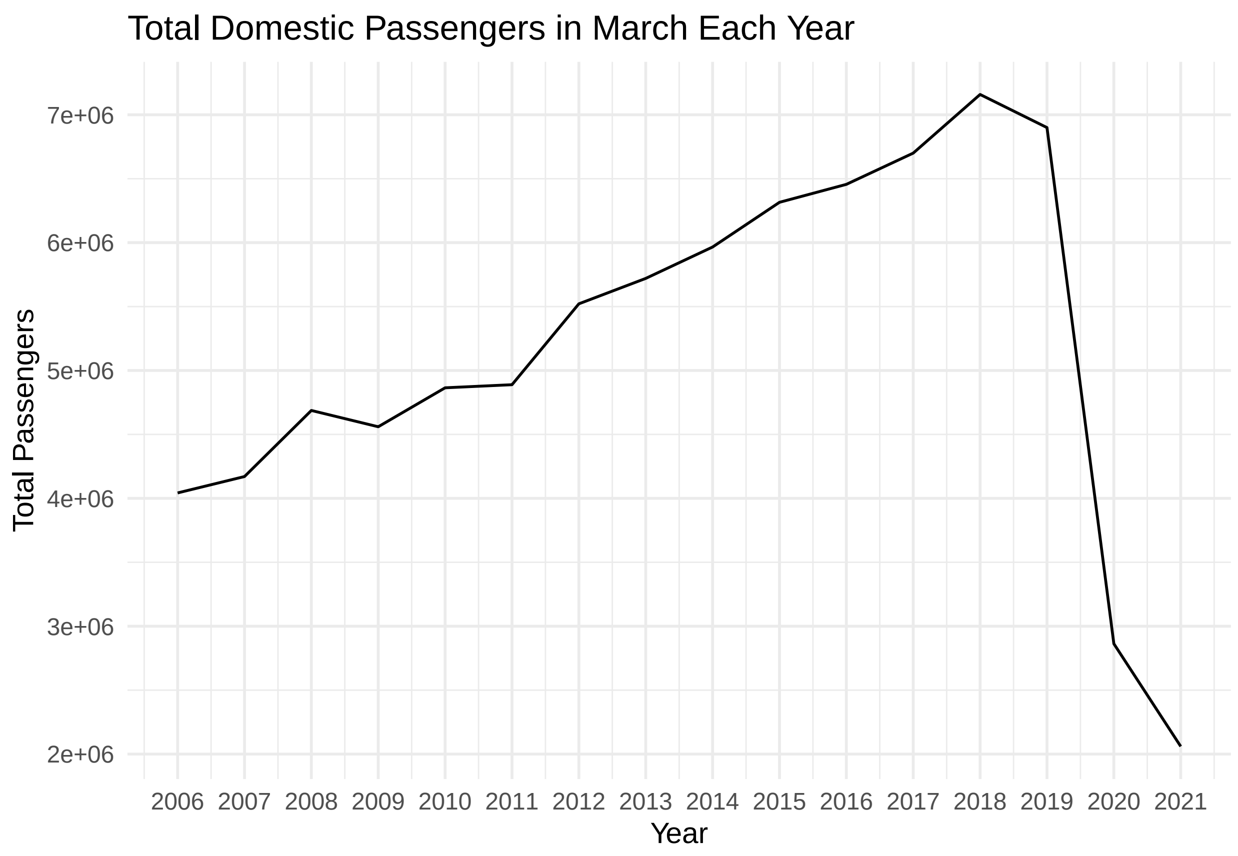
Q2

```r
# extract march each year, Domestic passengers activity
period <- data %>%
  dplyr::filter(GEO.Summary == "Domestic", str_detect(Activity.Period, "....03"))

# Divided year and month in Activity.Period
domesticActivity <- period %>%
  mutate(date = ym(Activity.Period)) %>%
  mutate_at(vars(date), funs(year, month))
```

```
## Warning: `funs()` was deprecated in dplyr 0.8.0.
## i Please use a list of either functions or lambdas:
##
## # Simple named list: list(mean = mean, median = median)
##
## # Auto named with `tibble::lst()`: tibble::lst(mean, median)
##
## # Using lambdas list(~ mean(., trim = .2), ~ median(., na.rm = TRUE))
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```

```
# Count total passengers
total_passenger <- domesticActivity %>%
  group_by(year) %>%
  summarize(Total_Passenger = sum(Passenger.Count))
```

```
# Visualize the trend of total passengers

ggplot(total_passenger, aes(x = year, y= Total_Passenger)) +
  geom_line() +
  labs(title = "Total Domestic Passengers in March Each Year",
       x = "Year",
       y = "Total Passengers") +
  scale_x_continuous(labels = total_passenger$year, breaks = total_passenger$year) +
  theme_minimal()
```



The domestic passenger count in 2006 was approximately 4 million and gradually increased to its peak at 7 million in 2018. In 2019, it experienced a slight decrease, followed by a significant decline in 2020 and 2021 with 2 million domestic passengers.

Q3

```
# Forecasting using a simple moving average

# Calculate the simple moving average (SMA)
SMA <- total_passenger %>%
  filter(year %in% c(2016, 2017, 2018)) %>%
  summarize(SMA = mean(Total_Passenger)) %>%
  pull("SMA")


# Calculate the actual total passengers for March 2019
actual_2019 <- total_passenger %>%
  filter(year == 2019) %>%
  select(Total_Passenger) %>%
  pull()

# Predict March 2019 using SMA
predicted_2019 <- SMA

# Calculate the error
error <- actual_2019 - predicted_2019

# Print the results
cat("Forecasted value for March 2019:", predicted_2019, "\n")
```

```
## Forecasted value for March 2019: 6771607
```

```
cat("Actual value for March 2019:", actual_2019, "\n")
```

```
## Actual value for March 2019: 6899726
```

```
cat("Error:", error, "\n")
```

```
## Error: 128119.3
```

In this case, the forecasted value with smooth moving average is lower than the actual value by approximately 128,119.3. This suggests that the Simple Moving Average (SMA) method used for forecasting in this instance resulted in an underestimation of the actual passenger count for March 2019.

Q4

```r
# Select three-year passengers data
three_year_passenger <- total_passenger %>%
  filter(year %in% c(2016, 2017, 2018))
weight <- c(3, 5, 7)
sw <- weight * three_year_passenger
weight_avg <- sum(sw$Total_Passenger) / sum(weight)

# Calculate the error
error <- actual_2019 - weight_avg

# Print the results
cat("Forecasted value for March 2019:", weight_avg, "\n")
```

```
## Forecasted value for March 2019: 6865349
```

```r
cat("Actual value for March 2019:", actual_2019, "\n")
```

```
## Actual value for March 2019: 6899726
```

```r
cat("Error:", error, "\n")
```

```
## Error: 34377.47
```

The forecasted value from the weighted average calculation is slightly lower than the actual value by approximately 34,377.47. Compared to SMA in Q2, this suggests that the weighted moving average method used for forecasting in this instance resulted in a relatively small underestimation of the actual passenger count for March 2019.

Q5

```r
exp_table <- total_passenger
exp_table$Ft <- 0
exp_table$E <- 0

# Filter data for the period (2008 to 2018)
exp_table <- exp_table %>%
  filter(year %in% c(2008:2019))

alpha <- 0.7

# Initialize the first forecast value (Ft) with the first actual value
exp_table$Ft[1] <- exp_table$Total_Passenger[1]

# Initialize the first error (E) with the difference between the first two observations
exp_table$Ft[2] <- exp_table$Ft[1] + alpha * exp_table$E[1]

# Calculate the forecast value and error
for (i in 2:nrow(exp_table)) {
  exp_table$Ft[i] <- exp_table$Ft[i-1] + alpha * exp_table$E[i-1]
  exp_table$E[i] <- exp_table$Total_Passenger[i] - exp_table$Ft[i]
}

# View the resulting exp_table
print(exp_table)
```

```
## # A tibble: 12 x 4
##     year Total_Passenger       Ft        E
##    <dbl>           <int>    <dbl>    <dbl>
##  1  2008         4687274 4687274        0
##  2  2009         4559998 4687274  -127276
##  3  2010         4864702 4598181.  266521.
##  4  2011         4888920 4784746.  104174.
##  5  2012         5521460 4857668.  663792.
##  6  2013         5720054 5322322.  397732.
##  7  2014         5965800 5600734.  365066.
##  8  2015         6315394 5856280.  459114.
##  9  2016         6455762 6177660.  278102.
## 10  2017         6700232 6372331.  327901.
## 11  2018         7158826 6601862.  556964.
## 12  2019         6899726 6991737.  -92011.
```

The negative sign in the error indicates that the forecasted value is higher than the actual value. In this case, the forecast using exponential smoothing overestimated the actual passenger count for March 2019 by approximately -92010.74.

Q6

```
# Filter data for the training period (2008 to 2018)
training_data <- total_passenger %>%
  filter(year %in% c(2008:2018))

# Build a simple linear regression model
linear_model <- lm(Total_Passenger ~ year, data = training_data)

# Print the coefficients
print(linear_model)
```

```
##
## Call:
## lm(formula = Total_Passenger ~ year, data = training_data)
##
## Coefficients:
## (Intercept)         year
##  -524789117       263538
```

The intercept is -524789117 and the slope is 264548 of the linear model.

```r
# Forecast total passenger activity for 2019 and 2020 using the coefficients
forecast_2019 <- -524789117 + 263538 * 2019
forecast_2020 <- -524789117 + 263538* 2020

# Actual values for 2019 and 2020
actual_2019 <- total_passenger %>%
  filter(year == 2019) %>%
  select(Total_Passenger) %>%
  pull()

actual_2020 <- total_passenger %>%
  filter(year == 2020) %>%
  select(Total_Passenger) %>%
  pull()


# Error calculation for 2019 and 2020
error_2019 <- actual_2019 - forecast_2019
error_2020 <- actual_2020 - forecast_2020

# Print the forecasted values
cat("Forecasted total passenger activity for 2019:", forecast_2019, "\n")
```

```
## Forecasted total passenger activity for 2019: 7294105
```

```r
cat("Error for 2019:", error_2019, "\n")
```

```
## Error for 2019: -394379
```

```r
cat("Forecasted total passenger activity for 2020:", forecast_2020, "\n")
```

```
## Forecasted total passenger activity for 2020: 7557643
```

```r
cat("Error for 2020:", error_2020, "\n")
```

```
## Error for 2020: -4694569
```

The negative errors for both 2019 and 2020 indicate that the model tends to overestimate the total passenger activity. Specifically, the error for 2020 is approximately -4,694,569, indicating a substantial deviation. This means that the actual values for these years are lower than what the linear regression model predicted.

Q7

1. Exponential Smoothing model

```r
exp_table <- total_passenger
exp_table$Ft <- 0
exp_table$E <- 0

```

```r
# Filter data for the period (2008 to 2018)
exp_table <- exp_table %>%
  filter(year %in% c(2008:2018))

alpha <- 0.7

# Initialize the first forecast value (Ft) with the first actual value
exp_table$Ft[1] <- exp_table$Total_Passenger[1]

# Initialize the first error (E) with the difference between the first two observations
exp_table$Ft[2] <- exp_table$Ft[1] + alpha * exp_table$E[1]

# Calculate the forecast value and error
for (i in 2:nrow(exp_table)) {
  exp_table$Ft[i] <- exp_table$Ft[i-1] + alpha * exp_table$E[i-1]
  exp_table$E[i] <- exp_table$Total_Passenger[i] - exp_table$Ft[i]
}

# View the resulting exp_table
print(exp_table)
```

```
## # A tibble: 11 x 4
##     year Total_Passenger       Ft         E
##    <dbl>           <int>    <dbl>     <dbl>
##  1  2008         4687274 4687274        0
##  2  2009         4559998 4687274  -127276
##  3  2010         4864702 4598181.  266521.
##  4  2011         4888920 4784746.  104174.
##  5  2012         5521460 4857668.  663792.
##  6  2013         5720054 5322322.  397732.
##  7  2014         5965800 5600734.  365066.
##  8  2015         6315394 5856280.  459114.
##  9  2016         6455762 6177660.  278102.
## 10  2017         6700232 6372331.  327901.
## 11  2018         7158826 6601862.  556964.
```

```r
exp_table$squared_error <- (exp_table$E)^2
exp_MSE <- sum(exp_table$squared_error)/length(exp_table$year)
# Print the results
cat("Mean Squared Error (Exponential Smoothing Model):",exp_MSE , "\n")
```

```
## Mean Squared Error (Exponential Smoothing Model): 139638420983
```

2. Linear Regression Model

```r
# Filter data for the training period (2008 to 2018)
training_data <- total_passenger %>%
  filter(year %in% c(2008:2018))

# Build a simple linear regression model
linear_model <- lm(Total_Passenger ~ year, data = training_data)

print(linear_model)
```

```
##
## Call:
## lm(formula = Total_Passenger ~ year, data = training_data)
##
## Coefficients:
## (Intercept)         year
##  -524789117       263538
```

Intercept is -524789117, slope is 263538. y = 263538*year - 524789117

```r
# Extract the forecasted values
forecast_values_linear <- -524789117 + 263538 * (2008:2018)

# Extract the actual values for the forecast periods
actual_values_linear <- total_passenger %>%
  filter(year %in% c(2008:2018)) %>%
  select(Total_Passenger) %>%
  pull()

# Calculate the squared errors for each observation
squared_errors_linear <- sum((actual_values_linear - forecast_values_linear)^2)

# Calculate the Mean Squared Error (MSE)
mse_linear <- squared_errors_linear/length((2008:2018))

# Print the results
cat("Mean Squared Error (Linear Regression):", mse_linear, "\n")
```

```
## Mean Squared Error (Linear Regression): 20069151944
```

Linear Regression model has the smaller MSE, 20069151944.