Practice / Forecasting

Nithya Sarabudla

01-26-2024

Task Set I

Question 1

```
# Load the USArrests dataset
data <- USArrests

# Calculate the mean and standard deviation of the murder rate
mean_murders <- mean(USArrests$Murder)
sd_murders <- sd(USArrests$Murder)

# Find rows that contain outliers and get their names
outlier_states <- rownames(USArrests[abs((USArrests$Murder - mean_murders) / sd_murders) > 1.5,])

# Print the outlier state names
print(outlier_states)
```

```
## [1] "Florida"        "Georgia"        "Louisiana"      "Mississippi"
## [5] "North Dakota"   "South Carolina"
```

Question 2

Pearson correlation assumes that the data is normally distributed and measures linear relationships. It is suitable for continuous, numeric data when the relationship is linear.

Spearman correlation is a rank-based correlation that does not assume linearity. It is appropriate when the variables may have a monotonic relationship but not necessarily linear.

Kendall correlation is also a rank-based correlation that measures the strength of dependence between two variables. It is suitable when the relationship is not necessarily linear and may involve ties in the data.

Assess the strength of the correlation: Once you calculate the correlation coefficient, you can assess its strength using the following:

A correlation coefficient close to 1 indicates a strong positive linear relationship. A correlation coefficient close to -1 indicates a strong negative linear relationship. A correlation coefficient close to 0 indicates a weak or no linear relationship.

```r
# Calculate Pearson correlation coefficient
pearson_corr <- cor(USArrests$UrbanPop, USArrests$Murder, method = "pearson")

# Calculate Spearman correlation coefficient
spearman_corr <- cor(USArrests$UrbanPop, USArrests$Murder, method = "spearman")

# Calculate Kendall correlation coefficient
kendall_corr <- cor(USArrests$UrbanPop, USArrests$Murder, method = "kendall")

# Print the correlation coefficients
print(paste("Pearson correlation:", pearson_corr))
```

```
## [1] "Pearson correlation: 0.0695726217359934"
```

```r
print(paste("Spearman correlation:", spearman_corr))
```

```
## [1] "Spearman correlation: 0.106716341834532"
```

```r
print(paste("Kendall correlation:", kendall_corr))
```

```
## [1] "Kendall correlation: 0.0742891418849837"
```

The Pearson correlation coefficient is close to 0 (0.0695726), indicating a very weak positive linear relationship. This suggests that there is little to no linear association between urban population and murder rate.

The Spearman correlation coefficient (0.1067) is also quite low, suggesting a weak positive rank-based relationship. While it's slightly higher than the Pearson correlation, it still suggests a weak association.

The Kendall correlation coefficient (0.0743) is similar to the Pearson correlation and indicates a weak positive rank-based relationship.

In summary, all three correlation coefficients suggest a weak positive relationship between urban population and murder rate in the dataset. However, the correlations are weak, which means that changes in urban population do not strongly predict changes in the murder rate, and vice versa.

Task Set II

Question 1

```r
library(gsheet)

url <- "https://docs.google.com/spreadsheets/d/1tOnM9XceK4Ak8tzWQ2vDelWlJexzJiS3LbT6MN6_rW0/edit#gid=0"
data <- read.csv(text=gsheet2text(url, format='csv'),
                 stringsAsFactors=FALSE,
                 encoding = "UTF-8")
```

```r
# Removing the last line of the code
data <- head(data, n = -1)
```

```r
# Weighted Moving Average (WMA)
wmaForecast <- function(data, weights) {
  n <- nrow(data)
  wma <- (weights[1] * data$Subscribers[n] + weights[2] * data$Subscribers[n - 1]) / sum(weights)
  return(wma)
}

# Test WMA with weights (5 for the most recent year, and 2 for others)
wma_weights <- c(5, 2)
next_year_wma <- wmaForecast(data, wma_weights)
cat("Weighted Moving Average Forecast for Year 12:", next_year_wma, "\n")
```

```
## Weighted Moving Average Forecast for Year 12: 194662700
```

```r
# Exponential Smoothing (ES)
esForecast <- function(data, alpha) {
  n <- nrow(data)
  forecast_values <- numeric(n)
  forecast_values[1] <- data$Subscribers[1]

  for (i in 2:n) {
    forecast_values[i] <- alpha * data$Subscribers[i - 1] + (1 - alpha) * forecast_values[i - 1]
  }

  next_year_forecast <- alpha * data$Subscribers[n] + (1 - alpha) * forecast_values[n]
  return(next_year_forecast)
}

# Test ES with alpha = 0.4
alpha <- 0.4
next_year_es <- esForecast(data, alpha)
cat("Exponential Smoothing Forecast for Year 12:", next_year_es, "\n")
```

```
## Exponential Smoothing Forecast for Year 12: 165168214
```

```r
# Linear Regression Trend Line (TL)
tlForecast <- function(data, year) {
  model <- lm(Subscribers ~ Year, data = data)
  next_year_forecast <- predict(model, newdata = data.frame(Year = year))
  return(next_year_forecast)
}

# Test TL for Year 12
next_year_tl <- tlForecast(data, 12)
cat("Linear Regression Forecast for Year 12:", next_year_tl, "\n")
```

```
## Linear Regression Forecast for Year 12: 203610220
```

Question 2 and 3

```r
# Initialize new columns for MSE
data$MSE_ExponentialSmoothing <- 0
data$MSE_LinearRegression <- 0
data$MSE_WeightedMovingAverage <- 0

# Calculate MSE for Exponential Smoothing (ES)
for (i in 3:nrow(data)) {
  # Forecast the ith value using Exponential Smoothing (ES)
  forecast_ES <- esForecast(data[1:(i-1), ], alpha = 0.4)

  # Calculate the squared error and store it in the data frame
  data$MSE_ExponentialSmoothing[i] <- (data$Subscribers[i] - forecast_ES)^2
}

# Calculate the mean MSE for Exponential Smoothing (ES)
mean_MSE_ES <- mean(data$MSE_ExponentialSmoothing)

# Format and print the MSE in standard numeric notation
cat("Mean Squared Error (MSE) for Exponential Smoothing (ES): ", format(mean_MSE_ES, scientific = FALSE)
```

```
## Mean Squared Error (MSE) for Exponential Smoothing (ES):  1471030393938056
```

```r
# Fit a linear regression model using the entire dataset
model <- lm(Subscribers ~ Year, data = data)

# Calculate MSE for Linear Regression
for (i in 1:nrow(data)) {
  # Forecast the ith value using Linear Regression
  forecast_LR <- predict(model, newdata = data.frame(Year = data$Year[i]))

  # Calculate the squared error and store it in the data frame
  data$MSE_LinearRegression[i] <- (data$Subscribers[i] - forecast_LR)^2
}

# Calculate the mean MSE for Linear Regression
mean_MSE_LR <- mean(data$MSE_LinearRegression)
cat("Mean Squared Error (MSE) for Linear Regression : ", format(mean_MSE_LR, scientific = FALSE), "\n")
```

```
## Mean Squared Error (MSE) for Linear Regression :  126534746000244
```

```r
# Calculate MSE for Weighted Moving Average (MA)
for (i in 3:nrow(data)) {
  # Forecast the ith value using Weighted Moving Average (MA)
  forecast_MA <- wmaForecast(data[1:(i-1), ], weights = c(5, 2))

  # Calculate the squared error and store it in the data frame
  data$MSE_WeightedMovingAverage[i] <- (data$Subscribers[i] - forecast_MA)^2
}
```

```r
# Calculate the mean MSE for Weighted Moving Average (MA)
mean_MSE_MA <- mean(data$MSE_WeightedMovingAverage)
cat("Mean Squared Error (MSE) for Weighted Moving Average (MA): ", format(mean_MSE_MA, scientific = FALS
```

```
## Mean Squared Error (MSE) for Weighted Moving Average (MA):  544143882735677
```

Question 4

```r
# Determine which model has the smallest mean squared error
smallest_MSE_model <- which.min(c(mean_MSE_ES, mean_MSE_LR, mean_MSE_MA))
models <- c("Exponential Smoothing (ES)", "Linear Regression (LR)", "Weighted Moving Average (MA)")
cat("The model with the smallest Mean Squared Error (MSE) is:", models[smallest_MSE_model], "\n")
```

```
## The model with the smallest Mean Squared Error (MSE) is: Linear Regression (LR)
```

Question 5

```r
## Define the ensembleForecast function
ensembleForecast <- function(linear_regression_forecast, exponential_smoothing_forecast, weighted_moving
  weight_trend_line <- 4
  weight_exponential_smoothing <- 2
  weight_weighted_moving_average <- 1

  total_weight <- weight_trend_line + weight_exponential_smoothing + weight_weighted_moving_average
  weighted_average_forecast <- (weight_trend_line * linear_regression_forecast +
                                weight_exponential_smoothing * exponential_smoothing_forecast +
                                weight_weighted_moving_average * weighted_moving_average_forecast) / to

  return(weighted_average_forecast)
}

# Calculate the ensemble forecast by passing the forecasted values as arguments
ensemble_forecast <- ensembleForecast(next_year_tl, next_year_es, next_year_wma)

# Print the ensemble forecast
cat("Ensemble Forecast for Year 12:", ensemble_forecast, "\n")
```

```
## Ensemble Forecast for Year 12: 191348573
```