````
---
title: "Sales Forecasting Dashboard"
author: "Andressa de Andrade Freitas"
output:
  flexdashboard::flex_dashboard:
    orientation: columns
    vertical_layout: fill
    runtime: shiny
---


```{r setup, include=FALSE}
# Set CRAN mirror
options(repos = c(CRAN = "https://cloud.r-project.org"))

# Install packages if not already installed
packages <- c("flexdashboard", "ggplot2", "shiny", "forecast", "tidyverse", "plotly",
"DT")
new_packages <- packages[!(packages %in% installed.packages()[,"Package"])]
if(length(new_packages)) install.packages(new_packages)
update.packages()


# Load libraries
library(flexdashboard)
library(ggplot2)
library(shiny)
library(forecast)
library(tidyverse)
library(plotly)
library(DT)


```

```{r load_data, include=FALSE}
# Load data
fsales <- "https://raw.githubusercontent.com/multidis/hult-inter-bus-reports-
r/main/forecasting/sales_weekly.csv"
sales <- read_csv(fsales)

# Ensure proper data formats
sales$Week <- as.integer(sales$Week)
sales$Store <- as.factor(sales$Store)

# Latest (current) week
nweek_now <- max(sales$Week)
```

```{r ui_elements, include=FALSE}
# Create store selection input
store_choices <- unique(sales$Store)
```

Column {data-width=300}
-------------------------------------

### UI Elements

```{r ui_controls}
# Create UI elements
selectInput("store", "Select Store:",
            choices = store_choices, selected = store_choices[1])
````
````

```
# UI for forecasting options
selectInput("forecast_option", "Select Forecast Type:",
            choices = c("Quarterly Forecast" = "quarterly", "Weekly Forecast" = "weekly"))

# UI for quarter selection
sliderInput("quarter", "Select Future Quarter:",
            min = 1, max = 8, value = 1)

# UI for week selection
sliderInput("week", "Select Week:",
            min = nweek_now + 1, max = nweek_now + 13*3, value = nweek_now + 1)

# UI for confidence intervals
sliderInput("confidence_interval", "Select Confidence Interval Level:",
            min = 80, max = 95, value = 95, step = 5)

# UI for comparing past and future sales
checkboxInput("compare_sales", "Compare Past and Future Sales", value = FALSE)

# UI for comparison of past year and upcoming year
sliderInput("comparison_years", "Select Number of Years for Comparison:",
            min = 1, max = 3, value = 1)
```

Column {data-width=700}

Tabs

```{r tabs, include=TRUE}
# Define tabs for the dashboard
tabsetPanel(
  tabPanel("Current Week Performance",
           value = "current_week",
           tableOutput("current_week_performance_table"),
           plotOutput("current_week_performance_chart")),

  tabPanel("Historical Sales Data",
           value = "historical_data",
           DTOutput("historical_data_table"),
           plotOutput("historical_sales_chart"),
           plotOutput("historical_sales_histogram")),

  tabPanel("Sales Forecasting",
           value = "sales_forecasting",
           plotlyOutput("forecast_chart")),

  tabPanel("Forecast with Confidence Intervals",
           value = "confidence_intervals",
           plotOutput("confidence_chart")),


  tabPanel("Forecast for Next 2 Months",
           value = "forecast_2_months",
           plotOutput("forecast_2_months_chart")),

  tabPanel("Comparison Past vs. Upcoming Year",
           value = "comparison_years",
           plotOutput("year_comparison_chart"))
)
```

```{r current_week_performance, include=TRUE}
```

```
# Current week performance indicator
filtered_sales <- reactive({
  req(input$store)
  subset(sales, Store == input$store)
})

current_week_sales <- reactive({
  latest_week_data <- subset(filtered_sales(), Week == nweek_now)
  forecast_model <- auto.arima(filtered_sales()$Weekly_Sales)
  current_week_forecast <- forecast(forecast_model, h = 1)$mean
  c(actual = latest_week_data$Weekly_Sales, forecast = current_week_forecast)
})

output$current_week_performance_chart <- renderPlot({
  data <- current_week_sales()
  df <- data.frame(
    Metric = c("Actual Sales", "Forecast Sales"),
    Value = c(data["actual"], data["forecast"])
  )

  ggplot(df, aes(x = Metric, y = Value, fill = Metric)) +
    geom_bar(stat = "identity", width = 0.4) +
    geom_text(aes(label = sprintf("%.2f", Value)), vjust = -0.3, color = "black", size =
5) +
    scale_fill_manual(values = c("Actual Sales" = "skyblue", "Forecast Sales" = "blue")) +
    labs(title = "Current Week Sales vs Forecast", x = "Metric", y = "Sales") +
    theme_minimal() +
    theme(
      plot.title = element_text(size = 14, margin = margin(t = 20, b = 20), hjust = 0.5),
      axis.title = element_text(size = 12, margin = margin(t = 10)),
      axis.text = element_text(size = 11),
      panel.grid = element_blank(),
      plot.margin = margin(15, 10, 15, 10)
    ) +
    coord_cartesian(clip = 'off')
}, height = 300)

```

```{r historical_data, include=TRUE}
output$historical_sales_chart <- renderPlot({
  data <- filtered_sales()
  ggplot(data, aes(x = Week, y = Weekly_Sales)) +
    geom_line() +
    geom_point() +
    labs(title = paste("Weekly Sales for Store", input$store),
         x = "Week", y = "Weekly Sales") +
    theme_minimal() +
    theme(
      plot.title = element_text(size = 14, margin = margin(b = 10), hjust = 0.5),
      axis.title = element_text(size = 12, margin = margin(t = 10)),
      axis.text = element_text(size = 10),
      panel.grid = element_blank(),
      plot.margin = margin(10, 10, 10, 10)
    )
}, height = 300, width = 500)  # Defina a largura e a altura explicitamente

output$historical_sales_histogram <- renderPlot({
  data <- filtered_sales()
  ggplot(data, aes(x = Weekly_Sales)) +
    geom_histogram(binwidth = 1000, fill = "skyblue", color = "black") +
    labs(title = paste("Histogram of Weekly Sales for Store", input$store),
         x = "Weekly Sales", y = "Frequency") +
    theme_minimal() +
```

```
    theme(
      plot.title = element_text(size = 14, margin = margin(b = 10), hjust = 0.5),
      axis.title = element_text(size = 12, margin = margin(t = 10)),
      axis.text = element_text(size = 10),
      panel.grid = element_blank(),
      plot.margin = margin(10, 10, 10, 10)
    )
}, height = 300, width = 500)  # Defina a largura e a altura explicitamente
```

```{r sales_forecasting, include=TRUE}
# Sales Forecasting

output$forecast_chart <- renderPlotly({
  req(input$forecast_option)

  # Get inputs
  forecast_option <- input$forecast_option
  store_num <- input$store

  # Prepare the time series data
  store_data <- sales %>% filter(Store == store_num)
  sales_ts <- ts(store_data$Weekly_Sales, frequency = 52)

  if (forecast_option == "quarterly") {
    # Quarterly Forecast
    quarter_weeks <- 13 * input$quarter
    forecast_model <- auto.arima(sales_ts)
    forecast_result <- forecast(forecast_model, h = quarter_weeks)

    # Prepare the forecast data
    forecast_data <- data.frame(
      Week = (nweek_now + 1):(nweek_now + quarter_weeks),
      Sales_Predicted = as.numeric(forecast_result$mean)
    )

    # Create the plot with ggplot2
    p <- ggplot(forecast_data, aes(x = Week, y = Sales_Predicted)) +
      geom_line() +
      labs(x = "Week", y = "Sales Forecast", title = "Quarterly Sales Forecast") +
      theme_minimal()

  } else if (forecast_option == "weekly") {
    # Weekly Forecast
    forecast_weeks <- input$week - nweek_now
    forecast_model <- auto.arima(sales_ts)
    forecast_result <- forecast(forecast_model, h = forecast_weeks)

    # Prepare the forecast data
    forecast_data <- data.frame(
      Week = (nweek_now + 1):(input$week),
      Sales_Predicted = as.numeric(forecast_result$mean)
    )

    # Create the plot with ggplot2
    p <- ggplot(forecast_data, aes(x = Week, y = Sales_Predicted)) +
      geom_line() +
      labs(x = "Week", y = "Weekly Sales Forecast", title = "Weekly Sales Forecast") +
      theme_minimal()
  }

  # Get client dimensions and render the plot
  cd <- session$clientData
  ggplotly(p, height = cd$output_plotly_height, width = cd$output_plotly_width)
```

```
})
```


```{r forecast_confidence_intervals, include=TRUE}
# Forecasting with Confidence Intervals

output$confidence_chart <- renderPlot({
  req(input$store)
  req(input$confidence_interval)

  # Filter data for selected store
  store_data <- sales %>% filter(Store == input$store)

  # Create time series object
  sales_ts <- ts(store_data$Weekly_Sales, frequency = 52)

  # Fit ARIMA model
  arima_model <- auto.arima(sales_ts, seasonal = TRUE)

  # Forecast with user-defined confidence interval
  arima_pred <- forecast(arima_model, h = 13 * 4, level = c(input$confidence_interval))

  # Create plot with confidence intervals
  autoplot(arima_pred) +
    labs(title = paste("Sales Forecast with", input$confidence_interval, "% Confidence
Interval for Store", input$store),
         x = "Week", y = "Forecasted Sales") +
    theme_minimal() +
    theme(
      plot.title = element_text(size = 14, margin = margin(b = 10), hjust = 0.5),
      axis.title = element_text(size = 12, margin = margin(t = 10)),
      axis.text = element_text(size = 10),
      panel.grid = element_blank(),
      plot.margin = margin(10, 10, 10, 10)
    )
}, height = 300)
```


```{r comparison_histogram, include=TRUE}
library(ggplot2)
library(forecast)
library(dplyr)
library(plotly)

# Generate the comparison plot with Plotly
output$comparison_histogram <- renderPlotly({
  req(input$store)

  store_data <- sales %>% filter(Store == input$store)

  # Define the number of weeks for past and future data
  past_weeks <- input$comparison_past_weeks  # Number of weeks to show for past data
  future_weeks <- input$comparison_future_weeks  # Number of weeks to forecast for future
data

  # Ensure that the data spans enough weeks
  if (nrow(store_data) < past_weeks) {
    return(NULL)  # Exit if not enough past data
  }

  # Fit ARIMA model
  forecast_model <- auto.arima(store_data$Weekly_Sales)
```

```
  # Forecast for future weeks
  forecast_result <- forecast(forecast_model, h = future_weeks)

  # Prepare past data
  past_data <- store_data %>%
    arrange(Week) %>%
    tail(past_weeks) %>%
    mutate(Type = "Past Sales")

  # Prepare future data
  future_data <- data.frame(
    Week = seq(from = max(store_data$Week) + 1, by = 1, length.out = future_weeks),
    Weekly_Sales = as.numeric(forecast_result$mean),
    Type = "Forecast Sales"
  )

  # Combine past and future data
  combined_data <- rbind(past_data, future_data)

  # Create interactive plot
  p <- ggplot(combined_data, aes(x = Week, y = Weekly_Sales, color = Type)) +
    geom_line(size = 1) +
    labs(title = paste("Comparison of Past and Future Sales for Store", input$store),
         x = "Week", y = "Sales") +
    theme_minimal() +
    theme(
      plot.title = element_text(size = 14, margin = margin(b = 10), hjust = 0.5),
      axis.title = element_text(size = 12, margin = margin(t = 10)),
      axis.text = element_text(size = 10),
      panel.grid = element_blank(),
      plot.margin = margin(10, 10, 10, 10)
    )

  # Convert ggplot to plotly
  ggplotly(p, height = 300)  # Adjust height as needed
})
#plotlyOutput("comparison_histogram")


```



```{r forecast_2_months, include=TRUE}
# Forecast for the next 2 months


output$forecast_2_months_chart <- renderPlot({
  store_data <- sales %>% filter(Store == input$store)
  sales_ts <- ts(store_data$Weekly_Sales, frequency = 52)

  # Fit ARIMA model
  arima_model <- auto.arima(sales_ts)
  arima_pred <- forecast(arima_model, h = 8) # Forecast for 2 months

  # Create plot with specified height
  autoplot(arima_pred) +
    labs(title = paste("Sales Forecast for the Next 2 Months for Store", input$store),
         x = "Week", y = "Forecasted Sales") +
    theme_minimal() +
    theme(
      plot.title = element_text(size = 14, margin = margin(b = 10), hjust = 0.5),
```

```
      axis.title = element_text(size = 12, margin = margin(t = 10)),
      axis.text = element_text(size = 10),
      panel.grid = element_blank(),
      plot.margin = margin(10, 10, 10, 10)
    )
}, height = 400) # Adjust height as needed

```



```{r year_comparison, include=TRUE}
# Comparison of Past Year vs. Upcoming Year

output$year_comparison_chart <- renderPlot({
  req(input$store)
  req(input$comparison_years)

  store_data <- sales %>% filter(Store == input$store)

  # Ensure data spans a full year for past data
  past_year_data <- store_data %>%
    filter(Week >= (nweek_now - 52) & Week <= nweek_now) %>%
    mutate(Year = "Past Year")

  # Fit ARIMA model for future year forecast
  forecast_model <- auto.arima(store_data$Weekly_Sales)
  future_year_forecast <- forecast(forecast_model, h = 52)

  future_year_data <- data.frame(
    Week = (nweek_now + 1):(nweek_now + 52),
    Weekly_Sales = as.numeric(future_year_forecast$mean),
    Year = "Upcoming Year"
  )

  # Combine past and future year data
  combined_year_data <- rbind(past_year_data %>% select(Week, Weekly_Sales, Year),
                              future_year_data %>% select(Week, Weekly_Sales, Year))

  # Plot the comparison
  ggplot(combined_year_data, aes(x = Week, y = Weekly_Sales, color = Year)) +
    geom_line() +
    labs(title = paste("Comparison of Past Year and Upcoming Year Sales for Store",
input$store),
         x = "Week", y = "Sales") +
    theme_minimal() +
    theme(
      plot.title = element_text(size = 14, margin = margin(b = 10), hjust = 0.5),
      axis.title = element_text(size = 12, margin = margin(t = 10)),
      axis.text = element_text(size = 10),
      panel.grid = element_blank(),
      plot.margin = margin(10, 10, 10, 10)
    )
}, height = 300)
```