Interactive Data Visualization

Este notebook proporciona una plataforma interactiva para la exploración de datos de series de tiempo de distintos productos pretroleros en Guatemala, permitiendo a los usuarios visualizar resultados a través de gráficos adecuados según el tipo de datos. Facilita la visualización de predicciones o clasificaciones a partir de tres modelos sencillos y presenta información relevante mediante gráficos enlazados. Los usuarios pueden ajustar el nivel de detalle en las visualizaciones y comparar los modelos predictivos.

Authors:

- Adrian Flores
- Andrea Ramirez

Import Libraries ⬇️

In [ ]:
```
# Data manipulation and visualization
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.ticker as mticker
import ipywidgets as widgets
import plotly.graph_objs as go
from IPython.display import display
from plotly.subplots import make_subplots

# Standard libraries
import warnings
warnings.filterwarnings('ignore')

# ===== ===== Reproducibility Seed ===== =====
# Set a fixed seed for the random number generator for reproducibility
random_state = 42

# Set matplotlib inline
%matplotlib inline

# Set default figure size
plt.rcParams['figure.figsize'] = (16, 8)

# Define custom color palette
palette = ['#648FFF', '#775EF0', '#DD2680', '#FE6100', '#FFB001']

# Set the style of seaborn
sns.set_theme(style="whitegrid")
```

Data Upload

In [ ]:
```
def read_and_process_excel(file_names):
    dfs = []  # Initialize an empty list to store DataFrames

    for file_name in file_names:
        # Read the Excel file while skipping the first six rows of headers
        df = pd.read_excel(file_name, skiprows=6)

        # Drop the last three rows from the DataFrame to remove any unwanted
        df = df.iloc[:-3]

        # Convert the 'Fecha' column to datetime format
        df['Fecha'] = pd.to_datetime(df['Fecha'])

        # Set the 'Fecha' column as the index of the DataFrame
        df.set_index('Fecha', inplace=True)

        # Select only the specified columns and create a new column 'Diesel'
        df['Diesel'] = df['Diesel alto azufre'].fillna(0) + df['Diesel bajo

        # Select only the relevant columns: Gasolina regular, Gasolina super
        df = df[['Gasolina regular', 'Gasolina superior', 'Gas licuado de pe

        # Append the processed DataFrame to the list
        dfs.append(df)

    return dfs  # Return the list of DataFrames
```

In [ ]:
```
# List of Excel file names to be processed
file_names = ["consumo.xlsx", "importacion.xlsx"]
dataset_names = ["Consumo", "Importacion", "Precios"]

# Call the function to read and process the Excel files, storing the result
dataframes = read_and_process_excel(file_names)
```

In [ ]:
```
def read_price_dfs(sheetname, skip):
  # Read the Excel file while skipping the first six rows of headers
  df = pd.read_excel("precios.xlsx", skiprows=skip, sheet_name=sheetname)
  # Drop the last three rows from the DataFrame to remove any unwanted data
  df = df.iloc[1:-3]
  # Convert the 'Fecha' column to datetime format
  df['Fecha'] = pd.to_datetime(df['FECHA'])
  # Set the 'Fecha' column as the index of the DataFrame
  df.set_index('Fecha', inplace=True)
  # Remove last column
  df = df.iloc[:, :-1]
  # Rename the columns correctly
  df.rename(columns={
      'FECHA': 'Fecha',
      'Tipo de Cambio': 'Tipo de Cambio',
      'Superior': 'Gasolina superior',
      'Regular': 'Gasolina regular',
```

```
        'Diesel': 'Diesel',
        'Bunker': 'Bunker',
        'Glp Cilindro 25Lbs.': 'Gas licuado de petróleo'
    }, inplace=True)
    # Select only the relevant columns: Gasolina regular, Gasolina superior, G
    df = df[['Gasolina regular', 'Gasolina superior', 'Gas licuado de petróleo
    # Drop NaN values from the final DataFrame
    df.dropna(inplace=True)
    return df
```

In [ ]:
```
list_price_params = [("2021", 6), ("2022", 6), ("2023", 7), ("2024", 7)]
```

In [ ]:
```
# Initialize an empty list to hold DataFrames
df_list = []

# Loop through each parameter to read and append DataFrames to the list
for year, skip in list_price_params:
    df = read_price_dfs(year, skip)
    df_list.append(df)

# Concatenate all DataFrames in the list into a single DataFrame
df = pd.concat(df_list)
# Optionally, sort the index if necessary
df.sort_index(inplace=True)
# Display the final DataFrame
dataframes.append(df)
```

Exploratory Analysis 🔎

(1) Descripción General de los Datos

In [ ]:
```
for i, df in enumerate(dataframes):
    # Get the number of rows in the merged DataFrame
    rows_num = df.shape[0]
    # Print the number of records in the DataFrame
    print(f"The given dataset {dataset_names[i]} has", rows_num, "registers.")
```

```
The given dataset Consumo has 293 registers.
The given dataset Importacion has 281 registers.
The given dataset Precios has 1302 registers.
```

Observaciones 💡 -->

> - El primer conjunto de datos se centra en la recopilación de información histórica sobre el consumo nacional de petróleo y productos petroleros en Guatemala. Este conjunto abarca entradas desde enero de 2000 hasta la fecha actual. La información fue obtenida de la página oficial del Ministerio de Energía y Minas. Cuenta con alrededor de 293 registros y 4 columnas.

> - El segundo conjunto de datos se centra en la recopilación de información histórica sobre la importación de productos derivados del petróleo en Guatemala. Este conjunto abarca entradas desde enero de 2001 hasta la fecha actual. La información fue obtenida de la página oficial del Ministerio de Energía y Minas. Cuenta con alrededor de 281 registros y 4 columnas, de manera similar al conjunto anterior.

> - El tercer conjunto de datos se centra en la recopilación de información histórica sobre los precios de productos derivados del petróleo en Guatemala (GTQ/Galón). Este conjunto abarca entradas desde enero de 2021 hasta la fecha actual. La información fue obtenida de la página oficial del Ministerio de Energía y Minas. Cuenta con 1302 registros y 4 columnas.

In [ ]:
```
for df in dataframes:
  # Basic information about the dataset
  print(df.info(), "\n")
```

```
<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 293 entries, 2000-01-01 to 2024-05-01
Data columns (total 4 columns):
 #   Column                   Non-Null Count  Dtype  
---  ------                   --------------  -----  
 0   Gasolina regular         293 non-null    float64
 1   Gasolina superior        293 non-null    float64
 2   Gas licuado de petróleo  293 non-null    float64
 3   Diesel                   293 non-null    float64
dtypes: float64(4)
memory usage: 11.4 KB
None

<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 281 entries, 2001-01-01 to 2024-05-01
Data columns (total 4 columns):
 #   Column                   Non-Null Count  Dtype  
---  ------                   --------------  -----  
 0   Gasolina regular         281 non-null    float64
 1   Gasolina superior        281 non-null    float64
 2   Gas licuado de petróleo  281 non-null    float64
 3   Diesel                   281 non-null    float64
dtypes: float64(4)
memory usage: 11.0 KB
None

<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 1302 entries, 2021-01-01 to 2024-07-28
Data columns (total 4 columns):
 #   Column                   Non-Null Count  Dtype 
---  ------                   --------------  ----- 
 0   Gasolina regular         1302 non-null   object
 1   Gasolina superior        1302 non-null   object
 2   Gas licuado de petróleo  1302 non-null   object
 3   Diesel                   1302 non-null   object
dtypes: object(4)
memory usage: 50.9+ KB
None
```

Observaciones 💡 -->

> - En este análisis observamos que no es necesario realizar alteraciones en los tipos de las variables en ninguno de los conjuntos de datos, ya que todas están definidas de manera adecuada. Es relevante destacar que, aunque no se observan valores nulos, existe la posibilidad de que algunos de ellos estén codificados de manera diferente. Por lo tanto, es importante tener esto en cuenta durante el proceso de limpieza. Sin embargo, a medida que se avance en el análisis exploratorio, se podrá obtener información más detallada al respecto.

(2) Clasificación de las Variables

Conjunto de Datos de Consumo -->

Nombre	Descripción	Clasificación
Fecha	Fecha de registro de los datos de consumo de combustibles.	Categórica
Gasolina regular	Consumo nacional de gasolina regular en barriles de 42 galones.	Numérica
Gasolina superior	Consumo nacional de gasolina superior en barriles de 42 galones.	Numérica
Gas licuado de petróleo	Consumo nacional de gas licuado de petróleo en barriles de 42 galones.	Numérica
Diesel	Consumo nacional de diesel en barriles de 42 galones.	Numérica

Conjunto de Datos de Importación -->

Nombre	Descripción	Clasificación
Fecha	Fecha de registro de la importación de productos derivados del petróleo.	Categórica
Gasolina regular	Importación nacional de gasolina regular en barriles de 42 galones.	Numérica
Gasolina superior	Importación nacional de gasolina superior en barriles de 42 galones.	Numérica
Gas licuado de petróleo	Imporatación nacional de gas licuado de petróleo en barriles de 42 galones.	Numérica
Diesel	Importación nacional de diesel en barriles de 42 galones.	Numérica

Conjunto de Datos de Precio -->

Nombre	Descripción	Clasificación
Fecha	Fecha de registro del precio en quetzales.	Categórica
Gasolina regular	Precio de gasolina regular en quetzales por galón.	Numérica
Gasolina superior	Precio de gasolina superior en quetzales por galón.	Numérica
Gas licuado de petróleo	Precio del gas licuado de petróleo en quetzales por galón.	Numérica
Diesel	Precio del diesel en quetzales por galón.	Numérica

(3) Exploración y Limpieza Inicial de los Datos

🖊️ Modificación de Etiquetas de Variables -->

Para facilitar la comprensión y el manejo del conjunto de datos, se procederá a modificar los nombres de las variables. Este cambio permitirá una organización más clara y una interpretación más precisa de la información.

In [ ]:
```
# Dictionary to rename columns for better readability
rename_col = {
    'Gasolina regular': 'gasoline_regular',    # Renaming 'Gasolina regular'
    'Gasolina superior': 'gasoline_superior',  # Renaming 'Gasolina superior
    'Gas licuado de petróleo': 'liquefied_gas', # Renaming 'Gas licuado de p
    'Diesel': 'diesel'                         # Renaming 'Diesel' to 'diesel
}
```

In [ ]:
```
for i, df in enumerate(dataframes):
  # Use a pandas function to rename the current function
  df = df.rename(columns = rename_col)
  # Change the index name from 'Fecha' to 'date'
  df.rename_axis('date', inplace=True)
  # Ensure all columns are numeric
  df = df.astype('float64')
  # Save changes
  dataframes[i] = df
  print(df.head(2), "\n")
```

```
            gasoline_regular  gasoline_superior  liquefied_gas     diesel
date
2000-01-01         202645.20          308156.82  194410.476190  634667.06
2000-02-01         205530.96          307766.31  174710.552381  642380.66

            gasoline_regular  gasoline_superior  liquefied_gas     diesel
date
2001-01-01         177776.50          373963.96  194065.738095  566101.99
2001-02-01         123115.99          243091.07  170703.380952  489525.80

            gasoline_regular  gasoline_superior  liquefied_gas  diesel
date
2021-01-01             21.11              21.91           99.0   17.61
2021-01-02             21.11              21.91           99.0   17.61
```

(4) Análisis Visual Preliminar de los Datos

In [ ]:
```
# Combine the DataFrames for the interactive selection
dataframe_mapping = {
    "Consumption": dataframes[0],
    "Importation": dataframes[1],
    "Pricing": dataframes[2]
}
```

(1) Análisis Visual de Series Temporales

In [ ]:
```
# Define a mapping of dataset names to y-axis labels
y_label_mapping = {
    "Consumption": "Consumption (Barrel, 42 gallons)",
```

```
    "Importation": "Importation (Barrel, 42 gallons)",
    "Pricing": "Price (GTQ/Gal)"
}
```

In [ ]:

```
def plot_dataframe(name):
    # Get the DataFrame based on the name
    df = dataframe_mapping.get(name)

    # Create a figure widget
    fig = go.FigureWidget()

    # Function to update the graph based on the selected fuel type
    def update_graph(selected_fuel):
        # Clear the current figure
        fig.data = []

        # Add the selected fuel type's line plot
        fig.add_trace(go.Scatter(
            x=df.index,
            y=df[selected_fuel],
            mode='lines',
            name=selected_fuel.replace('_', ' ').title(),
            line=dict(color=palette[['gasoline_regular', 'gasoline_superior'
        ))

        # Update titles and labels
        fig.update_layout(
            title=f'Trends in Petroleum Product {name} in Guatemala',
            title_font=dict(size=16, family='Arial, sans-serif', weight='bol
            xaxis_title='Date',
            yaxis_title=y_label_mapping.get(name, 'Price'),
            xaxis=dict(tickangle=45),  # Rotate x-axis labels for better rea
            yaxis=dict(tickformat=',.0f'),  # Format y-axis ticks with thous
            legend_title='Fuel Type',
            legend=dict(font=dict(size=12)),
            plot_bgcolor='white'  # Background color
        )

    # Create a dropdown to select the fuel type
    fuel_dropdown = widgets.Dropdown(
        options=[
            ('Gasoline Regular', 'gasoline_regular'),
            ('Gasoline Superior', 'gasoline_superior'),
            ('Liquefied Gas', 'liquefied_gas'),
            ('Diesel', 'diesel')
        ],
        value='diesel',  # Default value
        description='Fuel Type:',
    )

    # Set the update function to be called when the dropdown value changes
    fuel_dropdown.observe(lambda change: update_graph(change['new']), names=

    # Initial plot
    update_graph(fuel_dropdown.value)
```

```
    # Display the dropdown and the figure
    display(fuel_dropdown)
    display(fig)
```

(1) Análisis Visual de Serie de Consumo

In [ ]:
```
plot_dataframe("Consumption")
```

```
Dropdown(description='Fuel Type:', index=3, options=(('Gasoline Regular', 'g
asoline_regular'), ('Gasoline Supe…
FigureWidget({
    'data': [{'line': {'color': '#FE6100'},
              'mode': 'lines',
              'name': 'Diesel',
              'type': 'scatter',
              'uid': '495444ba-2a8e-4894-863b-2a6105945cec',
              'x': array([datetime.datetime(2000, 1, 1, 0, 0),
                          datetime.datetime(2000, 2, 1, 0, 0),
                          datetime.datetime(2000, 3, 1, 0, 0), ...,
                          datetime.datetime(2024, 3, 1, 0, 0),
                          datetime.datetime(2024, 4, 1, 0, 0),
                          datetime.datetime(2024, 5, 1, 0, 0)], dtype=objec
t),
              'y': array([ 634667.06,  642380.66,  699807.25, ..., 1393324.5
2, 1428143.44,
                          1401052.37])}],
    'layout': {'legend': {'font': {'size': 12}, 'title': {'text': 'Fuel Typ
e'}},
               'plot_bgcolor': 'white',
               'template': '...',
               'title': {'font': {'family': 'Arial, sans-serif', 'size': 16,
'weight': 'bold'},
                         'text': 'Trends in Petroleum Product Consumption in
Guatemala'},
               'xaxis': {'tickangle': 45, 'title': {'text': 'Date'}},
               'yaxis': {'tickformat': ',.0f', 'title': {'text': 'Consumptio
n (Barrel, 42 gallons)'}}}
})
```

(2) Análisis Visual de Serie de Importación

In [ ]:
```
plot_dataframe("Importation")
```

```
Dropdown(description='Fuel Type:', index=3, options=(('Gasoline Regular', 'g
asoline_regular'), ('Gasoline Supe…
```

```
FigureWidget({
    'data': [{'line': {'color': '#FE6100'},
              'mode': 'lines',
              'name': 'Diesel',
              'type': 'scatter',
              'uid': '1859d38c-4ac5-4208-ad88-d782a87af18c',
              'x': array([datetime.datetime(2001, 1, 1, 0, 0),
                          datetime.datetime(2001, 2, 1, 0, 0),
                          datetime.datetime(2001, 3, 1, 0, 0), ...,
                          datetime.datetime(2024, 3, 1, 0, 0),
                          datetime.datetime(2024, 4, 1, 0, 0),
                          datetime.datetime(2024, 5, 1, 0, 0)], dtype=objec
t),
              'y': array([ 566101.99,  489525.8 ,  575559.68, ..., 1477038.
, 1294706.12,
                          1470870.09])}],
    'layout': {'legend': {'font': {'size': 12}, 'title': {'text': 'Fuel Typ
e'}},
               'plot_bgcolor': 'white',
               'template': '...',
               'title': {'font': {'family': 'Arial, sans-serif', 'size': 16,
'weight': 'bold'},
                         'text': 'Trends in Petroleum Product Importation in
Guatemala'},
               'xaxis': {'tickangle': 45, 'title': {'text': 'Date'}},
               'yaxis': {'tickformat': ',.0f', 'title': {'text': 'Importatio
n (Barrel, 42 gallons)'}}}
})
```

(3) Análisis Visual de Serie de Precios

In [ ]:
```
plot_dataframe("Pricing")
```

```
Dropdown(description='Fuel Type:', index=3, options=(('Gasoline Regular', 'g
asoline_regular'), ('Gasoline Supe…
```

```
FigureWidget({
    'data': [{'line': {'color': '#FE6100'},
              'mode': 'lines',
              'name': 'Diesel',
              'type': 'scatter',
              'uid': '7ddf7914-21e4-4117-b429-b7450683c7a9',
              'x': array([datetime.datetime(2021, 1, 1, 0, 0),
                          datetime.datetime(2021, 1, 2, 0, 0),
                          datetime.datetime(2021, 1, 3, 0, 0), ...,
                          datetime.datetime(2024, 7, 26, 0, 0),
                          datetime.datetime(2024, 7, 27, 0, 0),
                          datetime.datetime(2024, 7, 28, 0, 0)], dtype=objec
t),
              'y': array([17.61, 17.61, 17.61, ..., 28.09, 28.09, 28.09])}],
    'layout': {'legend': {'font': {'size': 12}, 'title': {'text': 'Fuel Typ
e'}},
               'plot_bgcolor': 'white',
               'template': '...',
               'title': {'font': {'family': 'Arial, sans-serif', 'size': 16,
'weight': 'bold'},
                         'text': 'Trends in Petroleum Product Pricing in Gua
temala'},
               'xaxis': {'tickangle': 45, 'title': {'text': 'Date'}},
               'yaxis': {'tickformat': ',.0f', 'title': {'text': 'Price (GT
Q/Gal)'}}}
})
```

Instrucciones para Utilizar el Gráfico ⚙

En estos gráfico interactivos, puedes obesevar diferentes conjuntos de datos relacionados con productos derivados del petróleo en Guatemala, como "Consumo", "Importación" o "Precio". A través del menú desplegable, puedes elegir cuál tipo de combustible deseas visualizar para cada conjunto.

El gráfico muestra las tendencias a lo largo del tiempo para varios tipos de combustibles, incluyendo:

- Gasolina Regular
- Gasolina Superior
- Gas Licuado
- Diésel

Las líneas del gráfico corresponden a cada uno de estos combustibles, y la leyenda te ayuda a identificarlos por color.

El eje x representa las fechas, mientras que el eje y se adapta automáticamente al tipo de dato seleccionado:

- Si seleccionas Consumo o Importación, el eje y mostrará los valores en barriles (42 galones cada uno).
- Si seleccionas Precio, el eje y mostrará los precios en quetzales por galón (GTQ/Gal).

(2) Análisis de la Distribución Mensual y Anual de Series Temporales

In [ ]:
```
# Function to plot maximums interactively
def top_months_interactive(name, detail_level):

    print(f'Starting function for {name} with detail level {detail_level}')

    # Create a new DataFrame that aggregates data by the chosen detail level
    if detail_level == 'Monthly':
        # Aggregate by month and get max for each month
        monthly_max = dataframe_mapping[name].groupby(dataframe_mapping[name
    elif detail_level == 'Yearly':
        # Aggregate by year and get max for each year
        yearly_max = dataframe_mapping[name].resample('Y').max()
        # If you want to display only the years as labels
        monthly_max = yearly_max

    # Plotting
    monthly_max.plot(kind='bar', color=palette, edgecolor='black', width=0.8

    # Set proper names for the legend
    fuel_labels = ['Gasoline Regular', 'Gasoline Superior', 'Liquefied Gas',

    # Formatting the plot
    plt.title(f'Maximum Fuel {name} ({detail_level})', fontsize=16, fontweig
    if detail_level == 'Monthly':
        plt.xlabel('Month', fontsize=14)
        plt.xticks(ticks=range(1, 13), labels=['Jan', 'Feb', 'Mar', 'Apr', '
    else:
        plt.xlabel('Year', fontsize=14)
        plt.xticks(ticks=range(len(monthly_max.index)), labels=monthly_max.i

    plt.ylabel(f'Maximum {name}', fontsize=14)
    plt.legend(fuel_labels, title='Fuel Type', fontsize=12)

    # Set y-axis tick formatting to include thousand separators
    ax = plt.gca()  # Get the current axis
    ax.yaxis.set_major_formatter(mticker.StrMethodFormatter('{x:,.0f}'))  #

    plt.grid(axis='y', linestyle='--', alpha=0.7)
    plt.tight_layout()
    plt.show()
```

In [ ]:
```
# Use the interactive function with the dropdowns
df_selector = widgets.Dropdown(options=dataframe_mapping.keys(), description
detail_selector = widgets.Dropdown(options=['Monthly', 'Yearly'], descriptio

# Create interactive widgets
widgets.interactive(top_months_interactive, name=df_selector, detail_level=d
```

```
interactive(children=(Dropdown(description='Dataset:', options=('Consumptio
n', 'Importation', 'Pricing'), valu…
```

Instrucciones para Utilizar el Gráfico ⚙

En esta gráfica interactiva puedes visualizar los valores máximos de diferentes tipos de combustibles en Guatemala, **seleccionando el nivel de detalle** de tal manera que los datos se muestren ya sea de forma mensual o anual.

Puedes seleccionar el conjunto de datos que quieres analizar, como Consumo, Importación o Precios, usando el menú desplegable.

También puedes elegir el nivel de detalle para la visualización:

- Mensual: Verás los máximos agregados por cada mes.
- Anual: Verás los máximos agregados por cada año.

Los combustibles visualizados incluyen Gasolina Regular, Gasolina Superior, Gas Licuado y Diésel. El gráfico se muestra en formato de barras, donde cada barra representa el valor máximo del tipo de combustible seleccionado en un mes o año. Las etiquetas del eje x se ajustan según la opción seleccionada.

(3) Análisis de Frecuencia por Tipo de Combustible (Gráficos Enlazados)

In [ ]:
```
def interactive_histogram(df, name, units):
    # Create a dropdown to select the fuel type
    fuel_dropdown = widgets.Dropdown(
        options=[
            ('Gasoline Regular', 'gasoline_regular'),
            ('Gasoline Superior', 'gasoline_superior'),
            ('Liquefied Gas', 'liquefied_gas'),
            ('Diesel', 'diesel')
        ],
        value='diesel',  # Default value
        description='Fuel Type:',
    )

    # Create a label to display the selected range
    selected_range_label = widgets.Label(value='Selected Range: None')

    # Create an empty figure widget for dynamic updates
    fig_hist = go.FigureWidget()
    fig_line = go.FigureWidget()

    # Initialize the histogram with the default fuel type
    trace_hist = fig_hist.add_trace(go.Histogram(
        x=df['diesel'],
        name='Diesel',
        marker=dict(color=palette[0])
    ))

    # Initialize the line chart with the default fuel type
    fig_line.add_trace(go.Scatter(
        x=df.index,
        y=df['diesel'],
        mode='lines+markers',
        name='Diesel',
```

```
        line=dict(color=palette[4])
    ))

    # Set up the initial layout for histogram
    fig_hist.update_layout(
        title=f'Diesel {name} Distribution',
        xaxis_title=f'{name} {units}',
        yaxis_title='Count',
        hovermode='x unified',
        dragmode='select'  # Enable selection mode
    )

    # Set up the initial layout for line chart
    fig_line.update_layout(
        title=f'Diesel {name} Over Time',
        xaxis_title='Date',
        yaxis_title=f'{name} {units}',
        hovermode='x unified'
    )

    # Display the dropdown, histogram, and line chart
    display(fuel_dropdown)
    display(fig_hist)
    display(fig_line)
    display(selected_range_label)

    # Function to update the chart when the fuel type is changed
    def update_chart(change):
        fuel_type = change['new']

        # Update the histogram trace data
        with fig_hist.batch_update():
            fig_hist.data[0].x = df[fuel_type]
            fig_hist.data[0].name = fuel_type.replace('_', ' ').capitalize()
            fig_hist.data[0].marker.color = palette[0]

        # Update the line chart trace data
        with fig_line.batch_update():
            fig_line.data[0].y = df[fuel_type]
            fig_line.data[0].name = fuel_type.replace('_', ' ').capitalize()

        # Update the layout titles
        fig_hist.update_layout(
            title=f'{fuel_type.replace("_", " ").title()} {name} Distributio
            xaxis_title=f'{name} {units}'
        )
        fig_line.update_layout(
            title=f'{fuel_type.replace("_", " ").title()} {name} Over Time',
            yaxis_title=f'{name} {units}'
        )

    # Function to update the selected range and line chart
    def update_selected_range(trace, points, selector):
        if points.point_inds:  # Check if any points are selected
            selected_data = df[fuel_dropdown.value].iloc[points.point_inds]
            selected_range = (selected_data.min(), selected_data.max())
```

```
            selected_range_label.value = f'Selected Range: {selected_range}'

            # Filter the DataFrame based on the selected range
            filtered_df = df[(df[fuel_dropdown.value] >= selected_range[0]) &
                             (df[fuel_dropdown.value] <= selected_range[1])]

            # Update the line chart based on the filtered data
            with fig_line.batch_update():
                fig_line.data[0].y = filtered_df[fuel_dropdown.value]
                fig_line.data[0].x = filtered_df.index

        else:
            selected_range_label.value = 'Selected Range: None'

    # Set up observers for dropdown changes and selection updates
    fuel_dropdown.observe(update_chart, names='value')

    # Attach the update function to the histogram trace
    fig_hist.data[0].on_selection(update_selected_range)
```

(1) Distribucición de Conjunto de Datos de Consumo

In [ ]:
```
interactive_histogram(dataframes[0], "Consumption", "(Barrel, 42 gallons)")
```

```
Dropdown(description='Fuel Type:', index=3, options=(('Gasoline Regular', 'g
asoline_regular'), ('Gasoline Supe…
FigureWidget({
    'data': [{'marker': {'color': '#648FFF'},
              'name': 'Diesel',
              'type': 'histogram',
              'uid': '49cd1cde-0e77-4de6-a460-b8fac991412a',
              'x': array([ 634667.06,  642380.66,  699807.25, ..., 1393324.5
2, 1428143.44,
                           1401052.37])}],
    'layout': {'dragmode': 'select',
               'hovermode': 'x unified',
               'template': '...',
               'title': {'text': 'Diesel Consumption Distribution'},
               'xaxis': {'title': {'text': 'Consumption (Barrel, 42 gallon
s)'}},
               'yaxis': {'title': {'text': 'Count'}}}
})
```

```
FigureWidget({
    'data': [{'line': {'color': '#FFB001'},
              'mode': 'lines+markers',
              'name': 'Diesel',
              'type': 'scatter',
              'uid': '56f958ee-4e85-419e-9e09-db6250fd1576',
              'x': array([datetime.datetime(2000, 1, 1, 0, 0),
                          datetime.datetime(2000, 2, 1, 0, 0),
                          datetime.datetime(2000, 3, 1, 0, 0), ...,
                          datetime.datetime(2024, 3, 1, 0, 0),
                          datetime.datetime(2024, 4, 1, 0, 0),
                          datetime.datetime(2024, 5, 1, 0, 0)], dtype=objec
t),
              'y': array([ 634667.06,  642380.66,  699807.25, ..., 1393324.5
2, 1428143.44,
                          1401052.37])}],
    'layout': {'hovermode': 'x unified',
               'template': '...',
               'title': {'text': 'Diesel Consumption Over Time'},
               'xaxis': {'title': {'text': 'Date'}},
               'yaxis': {'title': {'text': 'Consumption (Barrel, 42 gallon
s)'}}}
})
Label(value='Selected Range: None')
```

(2) Distribucición de Conjunto de Datos de Importación

In [ ]:
```
interactive_histogram(dataframes[1], "Importation", "(Barrel, 42 gallons)")
```

```
Dropdown(description='Fuel Type:', index=3, options=(('Gasoline Regular', 'g
asoline_regular'), ('Gasoline Supe…
FigureWidget({
    'data': [{'marker': {'color': '#648FFF'},
              'name': 'Diesel',
              'type': 'histogram',
              'uid': 'cfa89ad5-646e-4be8-81f5-4d1a1813a35a',
              'x': array([ 566101.99,  489525.8 ,  575559.68, ..., 1477038.
, 1294706.12,
                          1470870.09])}],
    'layout': {'dragmode': 'select',
               'hovermode': 'x unified',
               'template': '...',
               'title': {'text': 'Diesel Importation Distribution'},
               'xaxis': {'title': {'text': 'Importation (Barrel, 42 gallon
s)'}},
               'yaxis': {'title': {'text': 'Count'}}}
})
```

```
FigureWidget({
    'data': [{'line': {'color': '#FFB001'},
              'mode': 'lines+markers',
              'name': 'Diesel',
              'type': 'scatter',
              'uid': 'c9433405-d1f3-49a8-82b5-10164fbdae2e',
              'x': array([datetime.datetime(2001, 1, 1, 0, 0),
                          datetime.datetime(2001, 2, 1, 0, 0),
                          datetime.datetime(2001, 3, 1, 0, 0), ...,
                          datetime.datetime(2024, 3, 1, 0, 0),
                          datetime.datetime(2024, 4, 1, 0, 0),
                          datetime.datetime(2024, 5, 1, 0, 0)], dtype=objec
t),
              'y': array([ 566101.99,  489525.8 ,  575559.68, ..., 1477038.
, 1294706.12,
                          1470870.09])}],
    'layout': {'hovermode': 'x unified',
               'template': '...',
               'title': {'text': 'Diesel Importation Over Time'},
               'xaxis': {'title': {'text': 'Date'}},
               'yaxis': {'title': {'text': 'Importation (Barrel, 42 gallon
s)'}}}
})
Label(value='Selected Range: None')
```

(3) Distribucición de Conjunto de Datos de Precio

In [ ]:
```
interactive_histogram(dataframes[2], "Pricing", "(GTQ/Gal)")
```

```
Dropdown(description='Fuel Type:', index=3, options=(('Gasoline Regular', 'g
asoline_regular'), ('Gasoline Supe…
FigureWidget({
    'data': [{'marker': {'color': '#648FFF'},
              'name': 'Diesel',
              'type': 'histogram',
              'uid': 'b47648f7-9e4f-4f09-85d4-a00249a7f1f9',
              'x': array([17.61, 17.61, 17.61, ..., 28.09, 28.09, 28.09])}],
    'layout': {'dragmode': 'select',
               'hovermode': 'x unified',
               'template': '...',
               'title': {'text': 'Diesel Pricing Distribution'},
               'xaxis': {'title': {'text': 'Pricing (GTQ/Gal)'}},
               'yaxis': {'title': {'text': 'Count'}}}
})
```

```
FigureWidget({
    'data': [{'line': {'color': '#FFB001'},
              'mode': 'lines+markers',
              'name': 'Diesel',
              'type': 'scatter',
              'uid': '02488d98-e5c3-4e98-947c-7fcadfe34e38',
              'x': array([datetime.datetime(2021, 1, 1, 0, 0),
                          datetime.datetime(2021, 1, 2, 0, 0),
                          datetime.datetime(2021, 1, 3, 0, 0), ...,
                          datetime.datetime(2024, 7, 26, 0, 0),
                          datetime.datetime(2024, 7, 27, 0, 0),
                          datetime.datetime(2024, 7, 28, 0, 0)], dtype=objec
t),
              'y': array([17.61, 17.61, 17.61, ..., 28.09, 28.09, 28.09])}],
    'layout': {'hovermode': 'x unified',
               'template': '...',
               'title': {'text': 'Diesel Pricing Over Time'},
               'xaxis': {'title': {'text': 'Date'}},
               'yaxis': {'title': {'text': 'Pricing (GTQ/Gal)'}}}
})
Label(value='Selected Range: None')
```

Instrucciones para Utilizar el Gráfico ⚙

En estas visualizaciones interactivas, puedes explorar un histograma que muestra la distribución de diferentes tipos de combustibles en Guatemala.

- Histograma: El histograma muestra la distribución de los tipos de combustible seleccionados. Visualiza con qué frecuencia ocurren diferentes rangos de valores en los datos.
- Gráfico de Líneas: El gráfico de líneas a la derecha muestra la tendencia de los precios de combustible a lo largo del tiempo. Proporciona una vista de serie temporal del tipo de combustible seleccionado.

Cuando selecciones un rango en el histograma, el gráfico de líneas se actualizará automáticamente para mostrar solo los puntos de datos que caen dentro de ese rango seleccionado. Esto te permite analizar las tendencias en los precios del combustible que corresponden a los valores de consumo seleccionados.

(4) Análisis Comparativo de Desempeño de Modelos Predictivos (Consumo de Diesel)

In [ ]:
```python
# Function to plot model comparisons with custom colors
def plot_model_comparisons(mae_results, colors):
    # Create a figure
    fig = make_subplots(rows=1, cols=1)

    # Add traces for each model with custom colors
    for (model, mae), color in zip(mae_results.items(), colors):
        fig.add_trace(
            go.Bar(
```

```
                x=[model],
                y=[mae],
                name=model,
                text=f'MAE: {mae:,.2f}',
                textposition='auto',
                hoverinfo='text',
                marker_color=color  # Set custom color
            )
        )

    # Update layout
    fig.update_layout(
        title='Model Comparisons: MAE of Diesel Consumption Predictions',
        xaxis_title='Model',
        yaxis_title='Mean Absolute Error (MAE)',
        yaxis=dict(title='MAE', zeroline=True),
        barmode='group'
    )

    fig.show()
```

In [ ]:
```
# Sample MAE results for the models
mae_results = {
    'LSTM': 96786.89,
    'ARIMA_1': 85512.12,
    'ARIMA_2': 89274.0
}
```

In [ ]:
```
# Call the function to plot with custom colors
plot_model_comparisons(mae_results, palette)
```

Instrucciones para Utilizar el Gráfico ⚙

Este gráfico de barras interactivo compara el Error Absoluto Medio (MAE) de diferentes modelos predictivos para el consumo de diésel en Guatemala. Cuanto más bajo sea el MAE, mejor será el rendimiento del modelo en la predicción del consumo de diésel.

Una vez que se muestre el gráfico, verás una barra para cada modelo predictivo (por ejemplo, LSTM, ARIMA_1, ARIMA_2). La altura de cada barra representa el valor de MAE para ese modelo. Mueve el cursor sobre cada barra para ver información detallada sobre el modelo, incluido el valor exacto de MAE que se mostrará como un tooltip.

En la parte derecha puedes seleccionar los ajustes de comparación, puedes comparar entre los 3 modelos o solamente 2 de ellos, ¡con la configuración que tú desees!