# Caso consultoria 1

## 13 de febrero del 2026

Enlace repositorio Github: https://github.com/VictorElias23/Prueba-Tecnica-1

**Caso de Negocio: Estimación de Costos de Equipos para Proyecto de Construcción**

Una empresa del sector constructor se encuentra en la fase de planificación de un proyecto de 36 meses de duración. Esta empresa es responsable de proporcionar la mano de obra, los materiales y los equipos necesarios para la ejecución del proyecto.

La empresa ha contratado a un consultor para que ayude a estimar el costo de dos equipos esenciales para la finalización del proyecto. Estos equipos tienen costos variables según los proveedores, pero es posible establecer un precio base en función del costo de las materias primas que los componen.

**Detalles de los Equipos:**

1. Equipo 1: Composición del Precio: 20% la materia prima X y el resto por la materia prima Y
2. Equipo 2: El precio del equipo 2 está conformado por partes iguales de las materias primas X, Y, Z.

**Beneficios Esperados:**

- Precisión en la estimación de costos, lo que permitirá una mejor planificación financiera del proyecto.
- Optimización de recursos al seleccionar proveedores con la mejor relación costo-beneficio.

## EXPLICACIÓN DEL CASO

La empresa de construcción plantea la necesidad de estimar el costo base de dos equipos que dependen del precio de distintas materias primas. Dichos costos están directamente influenciados por la volatilidad de la materia prima.

En este contexto, se proporcionan datos históricos de precios para tres materias primas, denominadas X, Y y Z, las cuales se utilizan en diferentes proporciones para la fabricación o adquisición de los equipos:

- **Equipo 1**: depende en un 20% de la materia prima X y en un 80% de la materia prima Y.
- **Equipo 2**: utiliza una proporción equivalente de las tres materias primas (33.3% X, 33.3% Y y 33.3% Z).

Dado que los precios de estos insumos presentan variaciones a lo largo del tiempo, se debe definir una metodología que permita analizar su comportamiento histórico y tratar predecir valores futuros que sirvan como **base** para la estimación de costos.

El desarrollo del caso se realizará en el lenguaje de programación Python.

## SUPUESTOS

Se establecen los siguientes supuestos para el desarrollo de este caso:

1. **Continuidad de los datos:** se asume que las tendencias y patrones históricos que muestran los datos proporcionados se mantendrán en el corto y mediano plazo, lo cual es útil para evitar considerar factores externos y extremos o situaciones que puedan afectar el mercado mundial.
2. **Calidad de los datos:** se asume que posterior a la limpieza y normalización de los datos, estos son consistentes y adecuados para su posterior análisis.
3. **Importancia de los datos:** se asume que los datos históricos son representativos del mercado y las tendencias, para así capturar los ciclos y niveles de volatilidad.
4. **Horizonte de la proyección:** se asume que las condiciones de las proyecciones realizadas se mantendrán al menos un año hasta iniciar el proyecto de construcción.

## FORMAS PARA RESOLVER EL CASO

### Exploración de los datos

Se realizó una primera inspección en los datos, y se observaron a simple vista los problemas que tienen los archivos CSV, se considera en modificar el nombre de las columnas a minúsculas, y como parte de los problemas de los archivos "Y" y "Z":

- Y: Tiene un error al momento de guardar los datos, al ser un CSV, los valores están separados por ",", sin embargo, se puede apreciar que se utilizan ";", además, se asumirá que las "," que existen deberían ser "." para formar los números decimales, entonces, eso será corregido.
- Z: Como único problema, es que las columnas están invertidas, por cuestiones de entendimiento y coherencia se invertirán las columnas entre sí.

Por último, también nos aseguramos de que, el tipo de dato de los valores de cada columna coincida con lo que es, una fecha para la columna "date" y un valor numérico para la columna "price". Este proceso se realiza dentro de la función “cargar_y_normalizar_csv”, en donde se manda el path del archivo correspondiente, el separador de los datos y el símbolo del número decimal.

```python
#Carga y Normalización, se especifica el separador y el decimal de los números
def cargar_y_normalizar_csv(path, sep=",", decimal=".", dayfirst=False):
    df = pd.read_csv(path, sep=sep, decimal=decimal) #Lectura de csv
    df.columns = df.columns.str.lower() #minúsculas

    if not {"date", "price"}.issubset(df.columns):
        raise ValueError(f"El archivo {path} no contiene las columnas date o price.")

    #Ordenamiento de columnas por facilidad
    df = df[["date", "price"]]

    #Transformamos la columna date a datetime
    df["date"] = pd.to_datetime(df["date"], dayfirst=dayfirst, errors="coerce")
    #Transformamos la columna price a numérico
    df["price"] = pd.to_numeric(df["price"], errors="coerce")
    return df
```

Posteriormente, se puede realizar un resumen básico de los datos, en donde podemos observar la cantidad de registros que tiene cada archivo de materia prima, el promedio, desviación estándar, valor mínimo, valor máximo, etc. A continuación se presenta dicha tabla:

| Dato | Materia prima X | Materia prima Y | Materia prima Z |
|---|---|---|---|
| **Registros** | 9,144 | 4,485 | 3,565 |
| **Promedio** | 51.32 | 565.45 | 2,037.08 |
| **Desviación estándar** | 32.99 | 145.15 | 372.97 |
| **Percentil 25** | 19.68 | 485 | 1,767.5 |
| **Percentil 50** | 45.94 | 543.67 | 1,974.75 |
| **Percentil 75** | 75.20 | 617.67 | 2,235.75 |
| **Precio mínimo** | 9.64 | 257.5 | 1,421.5 |
| **Precio máximo** | 146.08 | 1,062.37 | 3,984 |
| **Rango (min-max)** | 136.44 | 804.87 | 2,562.50 |

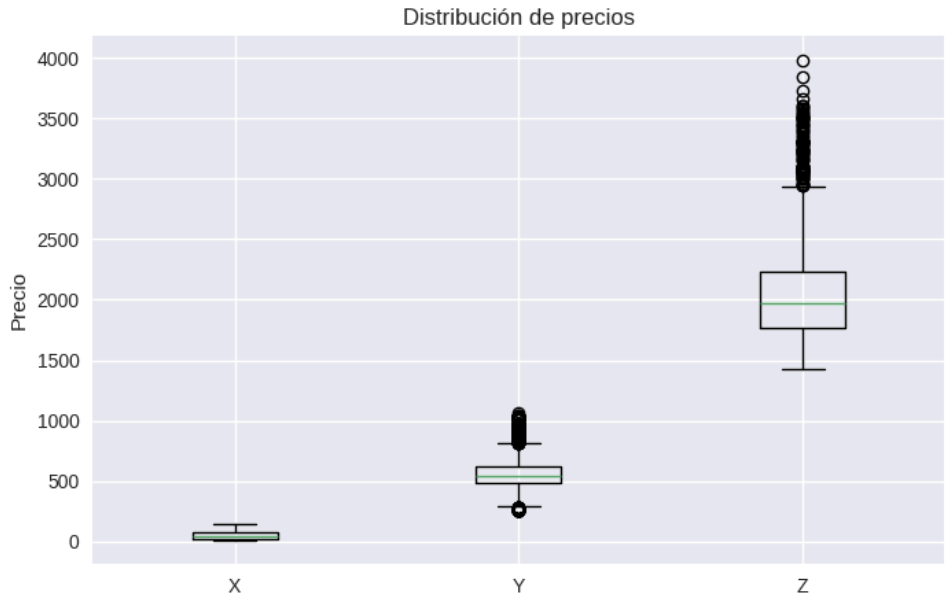
La tabla del resumen de los datos muestra que existen diferencias significativas en el nivel de precios y la variabilidad entre las materias primas. La materia prima "X" presenta un precio promedio bajo en comparación con "Y" y "Z", pero con una desviación estándar elevada relativamente, lo que refleja alta variabilidad. La materia prima "Y" muestra un nivel de precios intermedio con una dispersión moderada, sin embargo si cuenta con varios outliers. La materia prima "Z", además de presentar los precios promedio más altos, también presenta una mayor dispersión (más adelante se puede observar gráficamente con los boxplots), lo que la convierte en el principal factor de posible riesgo al momento de estimar costos de equipos que dependan de esta materia prima.

Esto se realizó con la función “resumen_estadistico”, donde se recibe el ‘dataframe’ y el nombre del archivo de materia prima.

```python
#Análisis descriptivo
#Resumen de los datos
def resumen_estadistico(df, nombre=""):
    print(f"\nResumen estadístico {nombre}")
    estadistica = df["price"].describe()
    print(estadistica)
    rango = df["price"].max() - df["price"].min()
    print(f"Rango (max - min): {rango:.2f}")
    return estadistica
```

## Distribución de los precios

El análisis de distribución de los precios mediante boxplots nos puede mostrar diferencias claras en los rangos y la variabilidad de las materias primas "X", "Y" y "Z".




La materia prima "X" presenta el rango de precios más bajo. La materia prima "Y" presenta una dispersión mayor que la materia prima "X", con una distribución más grande y outliers más marcados. Por otra parte, la materia prima "Z" muestra el mayor rango de precios y la mayor presencia de valores extremos, lo que sugiere que es la materia prima con mayor impacto potencial en el costo final de los equipos y la que introduce mayor "incertidumbre" en una posible estimación base.

Los boxplots fueron generados con la función “boxplots_precios”, mandando los ‘dataframes’, etiquetas y el título.

```python
#Boxplots
def boxplots_precios(dfs, etiquetas, titulo="Distribución de precios"):
    data = [df["price"].dropna() for df in dfs]
    plt.figure(figsize=(8, 5))
    plt.boxplot(data, labels=etiquetas, showfliers=True)
    plt.title(titulo)
    plt.ylabel("Precio")
    plt.show()
```

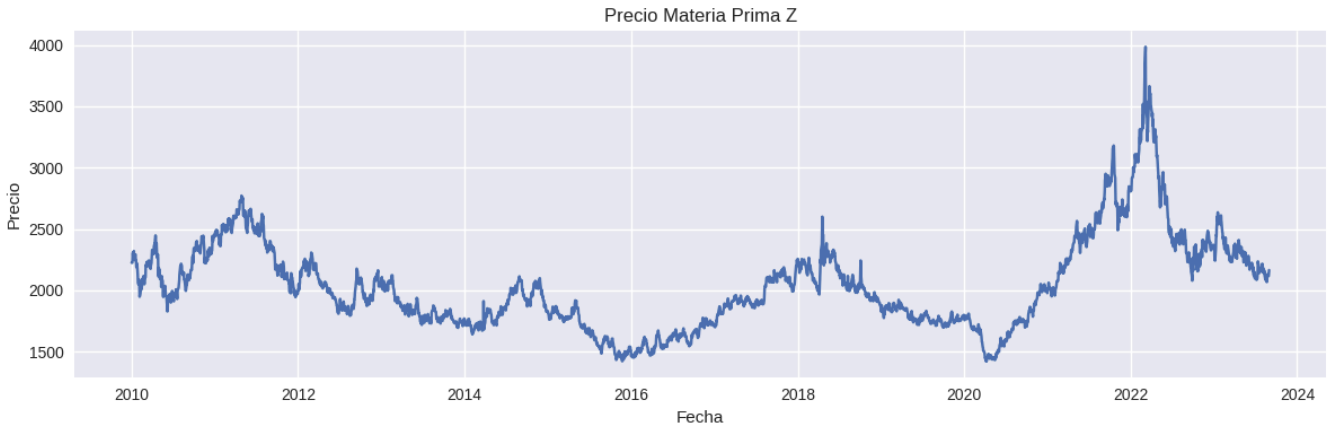
## Series de tiempo

Las series de tiempo nos pueden mostrar si algunas de las tres materias primas presentan comportamientos variables y no fijos a lo largo del tiempo, con períodos de alta volatilidad, que se pueden deber a ciertos eventos que ocurran en ese momento.

1. Serie de tiempo para materia prima X


2. Serie de tiempo para materia prima Y

3. Serie de tiempo para materia prima Z




La **materia prima "X"** muestra una tendencia de alza a largo plazo con picos pronunciados y correcciones. La **materia prima "Y"** presenta un comportamiento "cíclico", con períodos de alza y baja relativamente marcadas, así que posiblemente como la materia prima "Y", esté relacionado con eventos de mercado. Con la **materia prima "Z"**, se observan cambios parecidos a la materia prima "X", sin embargo llevado a precios más altos, evidenciando un mayor riesgo de estimación.

En conjunto, todos estos resultados indican que el uso de un único valor instantáneo para estimar costos puede generar sesgos, por lo que puede resultar más útil basar el resultado en una solución que atrape esta volatilidad y los ciclos que se repiten en la serie de tiempo.

Las series de tiempo fueron generadas por la función “plot_serie_temporal”, recibe el ‘dataframe’ y el nombre del gráfico.

```python
#Serie de tiempo
def plot_serie_temporal(df, nombre=""):
    df_plot = df.sort_values("date")
    plt.figure(figsize=(12, 4))
    plt.plot(df_plot["date"], df_plot["price"])
    plt.title(f"Precio Materia Prima {nombre}")
    plt.xlabel("Fecha")
    plt.ylabel("Precio")
    plt.tight_layout()
    plt.show()
```

## Métodos posibles para resolver el caso

Para abordar el problema de estimar el costo base de los equipos a partir de los precios históricos de las materias primas, se toman en cuenta distintas formas de resolución que parten

del análisis exploratorio de los datos. En un primer nivel, se contemplaron aproximaciones descriptivas basadas en métricas agregadas (por ejemplo, promedios históricos o promedios móviles) como una referencia inicial. Sin embargo, al ser un problema clásico de predicción de series de tiempo, se evaluaron enfoques de modelado de series temporales que permiten capturar de manera explícita la dependencia temporal de los precios, incluyendo modelos clásicos como **ARMA**, **ARIMA** y **SARIMA**.

1. **ARMA (AutoRegressive Moving Average)**

   El modelo **ARMA** combina componentes autorregresivos (AR) y de promedio móvil (MA) para modelar series temporales estacionarias, capturando la dependencia de los valores actuales con respecto a observaciones pasadas y errores previos.

   **Desventajas**:

   - Requiere que la serie sea estacionaria, lo cual no siempre se cumple en datos reales de mercado.
   - No modela explícitamente tendencias ni estacionalidad.
   - Su desempeño puede ser limitado en series con cambios estructurales o alta volatilidad.

2. **ARIMA (AutoRegressive Integrated Moving Average)**

   ARIMA extiende el modelo ARMA incorporando un componente de diferenciación (I) que permite manejar series no estacionarias, capturando tendencias en los datos. Es uno de los modelos más utilizados para el pronóstico de series temporales económicas y financieras, ya que permite trabajar con series que presentan crecimiento o decrecimiento en el tiempo.

   **Desventajas**:

   - Aunque modela tendencias, no incorpora de forma explícita componentes estacionales.
   - Puede requerir un ajuste cuidadoso de hiper parámetros para obtener buenos resultados.

3. **SARIMA (Seasonal ARIMA)**

SARIMA es una extensión de ARIMA que incorpora componentes estacionales, permitiendo modelar patrones recurrentes en el tiempo (por ejemplo, ciclos anuales o mensuales). Este modelo es útil cuando las series temporales presentan comportamientos cíclicos.

**Desventajas**:

- Mayor complejidad en la selección de hiper parámetros (componentes estacionales y no estacionales).
- Mayor costo computacional en comparación con ARMA y ARIMA.
- Riesgo de sobreajuste si no se realiza una validación adecuada de los parámetros.

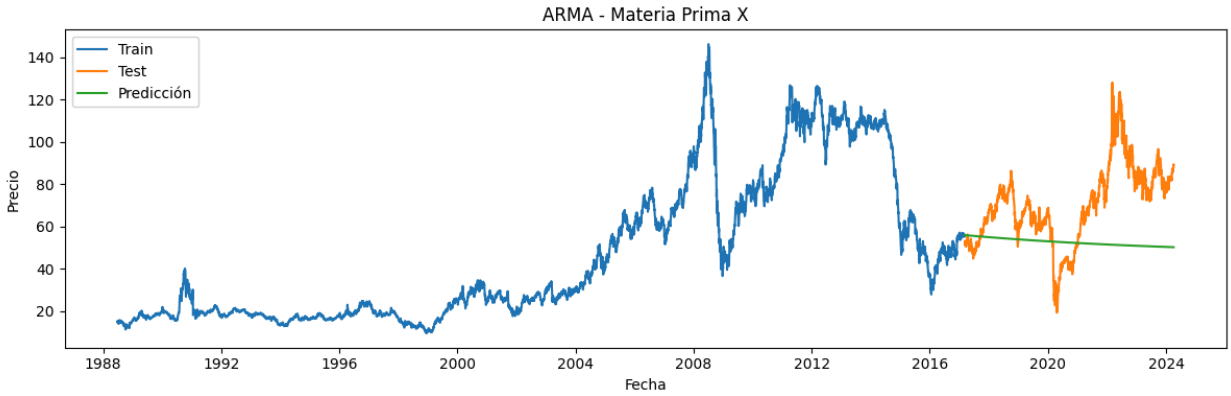
Tomando lo anterior, para cada uno de los modelos considerados (ARMA, ARIMA y SARIMA) se siguió un proceso común con el fin de realizar una comparación justa entre enfoques.

En primer lugar, las series de precios de las materias primas fueron ordenadas temporalmente y divididas en conjuntos de **entrenamiento** y **prueba** utilizando una partición del 80% para entrenamiento y 20% para prueba fuera de muestra. Posteriormente, cada modelo fue entrenado, utilizando configuraciones iniciales de parámetros, y se generaron predicciones sobre el horizonte correspondiente al conjunto de prueba.

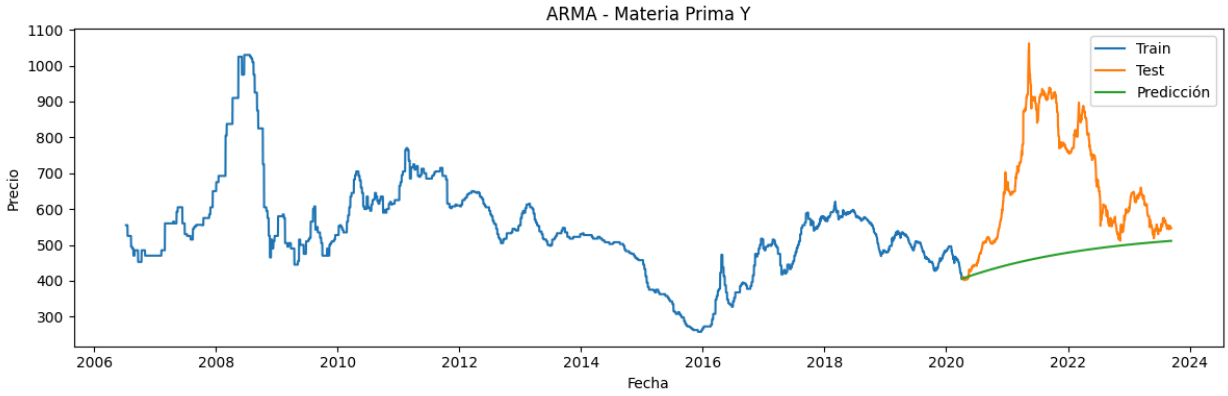
El desempeño de los modelos se evaluó mediante la métrica **RMSE (Raíz del error cuadrático medio)**, complementando el análisis cuantitativo con una inspección visual de las predicciones frente a los valores reales. Este procedimiento generó una manera consistente para conocer la capacidad de cada enfoque.

A continuación se muestran las series de tiempo comparando las predicciones realizadas con los datos reales:

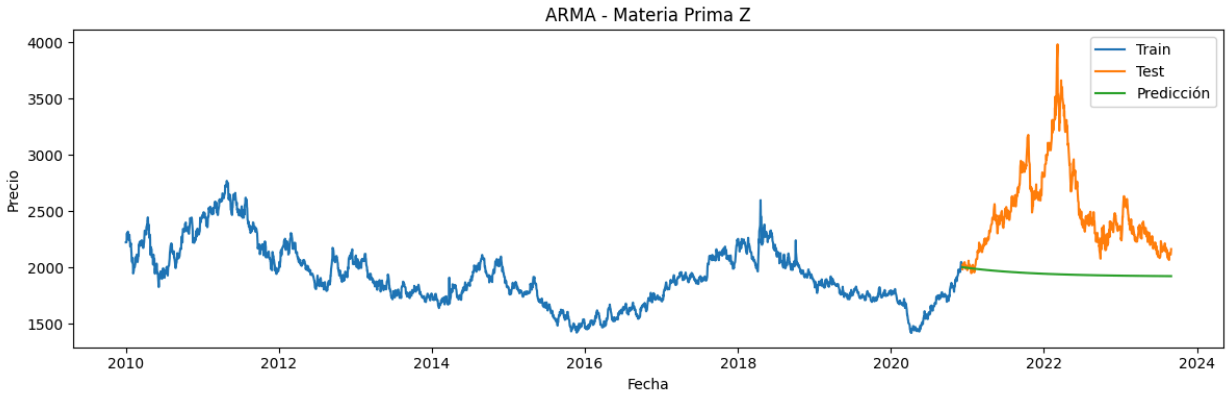
1. **ARMA (AutoRegressive Moving Average)**
    - **Materia prima X**




- **Materia prima Y**

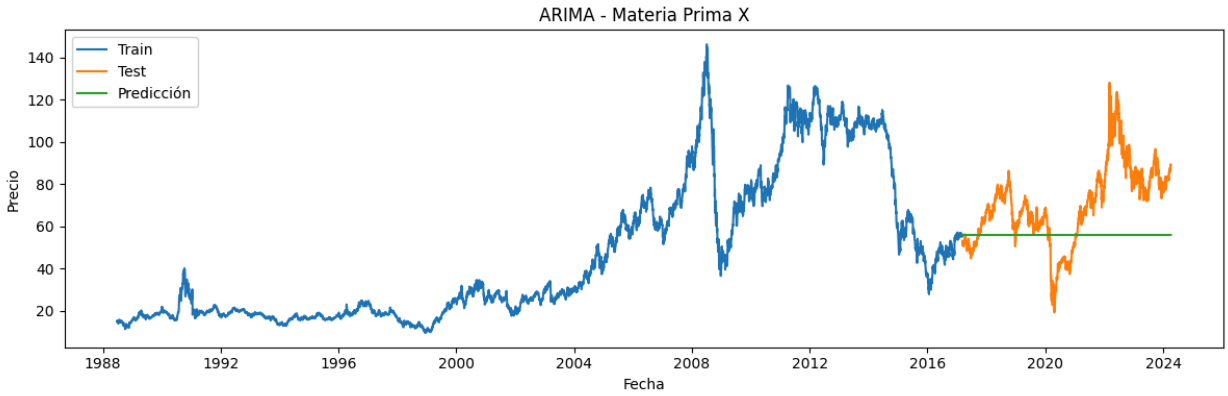



- **Materia prima Z**

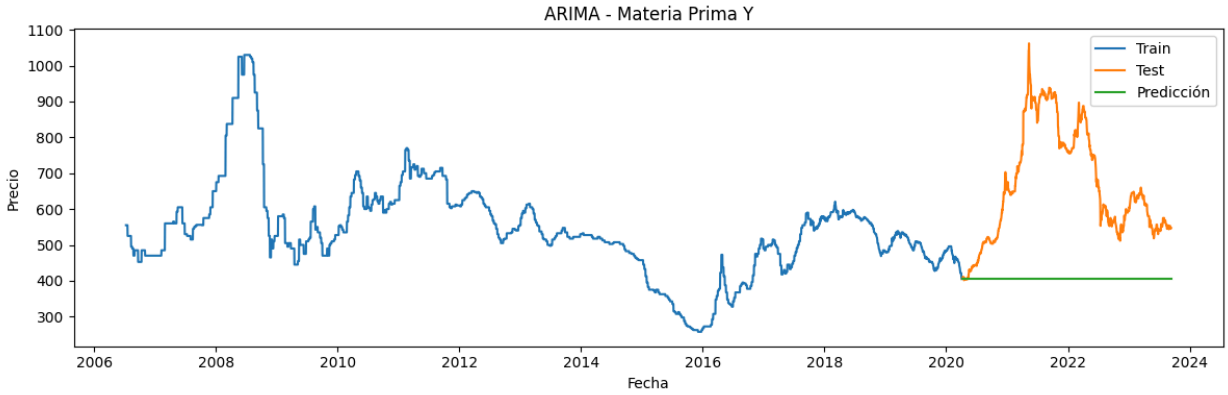


## 2. ARIMA (AutoRegressive Integrated Moving Average)

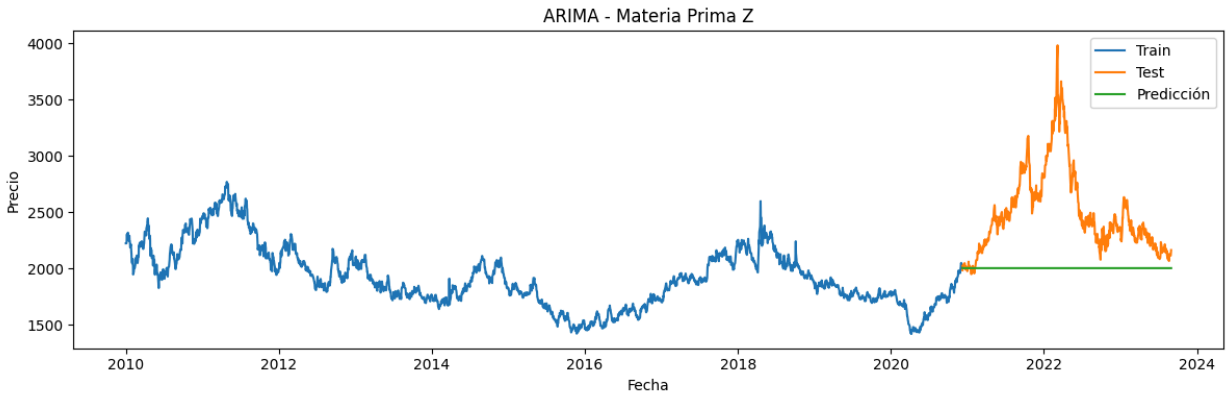
- **Materia prima X**




- **Materia prima Y**

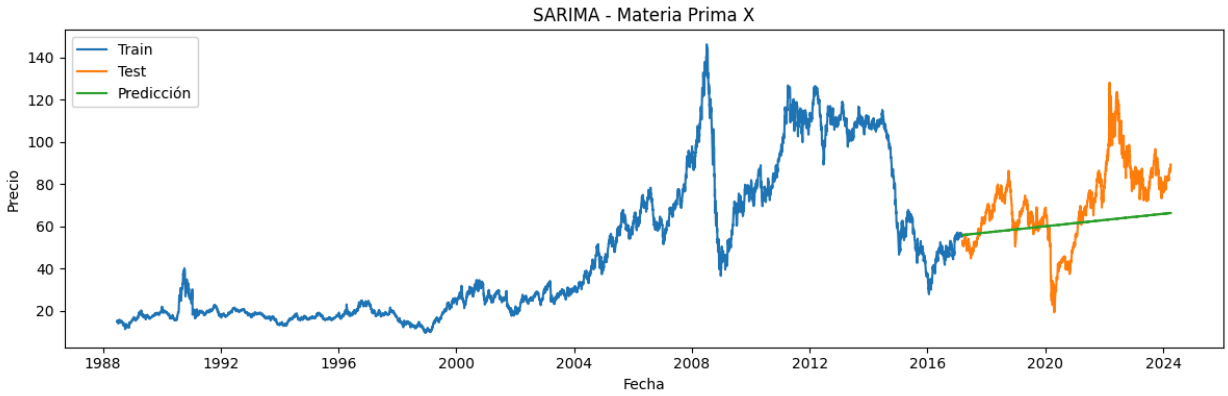



- **Materia prima Z**

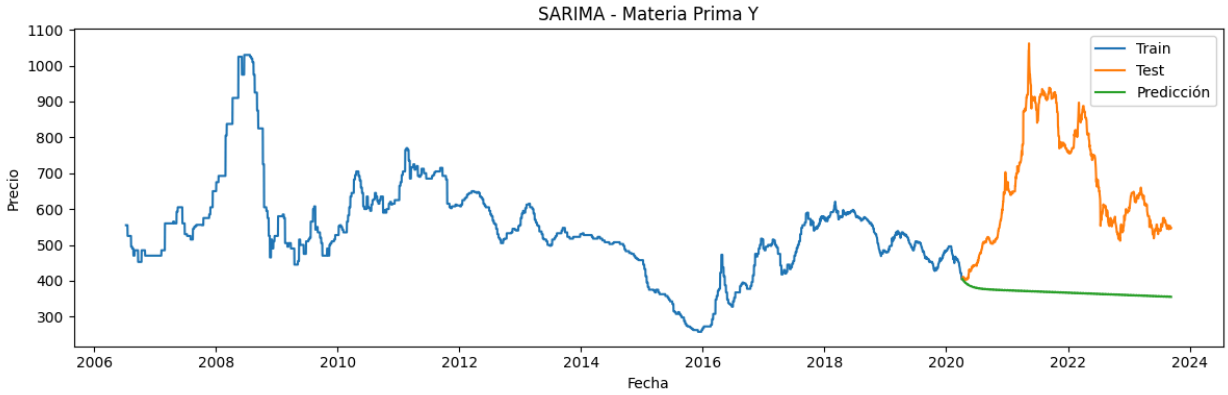


## 3. SARIMA (Seasonal ARIMA)

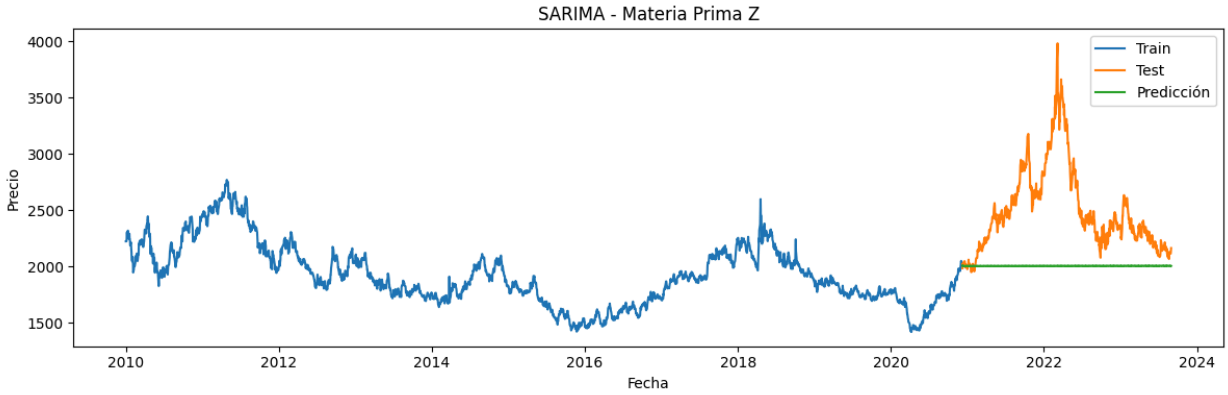
- **Materia prima X**




- **Materia prima Y**




- **Materia prima Z**




Los parámetros de los modelos ARMA, ARIMA y SARIMA se establecieron con valores iniciales estándar con el objetivo de realizar una comparación metodológica entre enfoques de

modelado. No se realizó una optimización de hiper parámetros. La búsqueda sistemática de configuraciones óptimas (p. ej., mediante grid search, AIC/BIC o validación cruzada temporal) se plantea como trabajo futuro.

A continuación se presenta la tabla comparativa con la métrica RMSE de los modelos para cada serie de tiempo de las materias primas:

| Modelo | RMSE Materia prima X | RMSE Materia prima Y | RMSE Materia prima Z |
|---|---|---|---|
| **ARMA** | 26.13 | 247.51 | 671.38 |
| **ARIMA** | 23.32 | 303.60 | 622.04 |
| **SARIMA** | 19.25 | 337.42 | 620.82 |

El desarrollo de la división de los datos fue ejecutada por la función “division_train_test”, donde se recibe el ‘dataframe’ y el tamaño del conjunto de datos de prueba. Posteriormente se realiza la predicción con la función “forecast_model”, que recibe el modelo y el tamaño de la proyección a predecir. Ahora, con la función “rmse” se obtiene la raíz del error cuadrático medio para evaluar los modelos, que recibe los valores reales y los valores predichos. Por último están las 3 funciones donde se realiza el entrenamiento de ARMA, ARIMA y SARIMA, llamadas “entrenar_arma”, “entrenar_arima” y “entrenar_sarima”, las cuales reciben los datos de entrenamiento y su configuración, que este caso fue por default.

```python
#Split de los datos para el entrenamiento de modelos (80%, 20%)
def division_train_test(df, train_size=0.8):
    df = df.sort_values("date") #Ordenamos los datos en orden de la fecha
    n = len(df)
    split_idx = int(n * train_size)
    train = df.iloc[:split_idx] #train
    test = df.iloc[split_idx:] #test
    return train, test


#Métricas y forecast
from sklearn.metrics import mean_squared_error

def forecast_model(model, steps):
    return model.forecast(steps=steps)

#Raíz del error cuadrático medio
def rmse(y_true, y_pred):
    return np.sqrt(mean_squared_error(y_true, y_pred))

from statsmodels.tsa.arima.model import ARIMA
from statsmodels.tsa.statespace.sarimax import SARIMAX

#Modelo ARMA
def entrenar_arma(train_prices, order=(1,0,1)):
    return ARIMA(train_prices, order=order).fit()

#Modelo ARIMA
def entrenar_arima(train_prices, order=(1,1,1)):
    return ARIMA(train_prices, order=order).fit()

#Modelo SARIMA
def entrenar_sarima(train_prices, order=(1,1,1), seasonal_order=(1,1,1,12)):
    return SARIMAX(
        train_prices,
        order=order,
        seasonal_order=seasonal_order,
        enforce_stationarity=False,
        enforce_invertibility=False
    ).fit(disp=False)
```

Para el apartado de la visualización de las series temporales con los datos reales y predichos se utilizó la función “plot_train_test_pred”, en donde se reciben todos los datos divididos por entrenamiento, prueba y predicción, y el título del gráfico.

```python
#Visualización de la serie de tiempo con los datos train, test y pred
def plot_train_test_pred(train_df, test_df, pred, titulo=""):
    plt.figure(figsize=(12, 4))
    plt.plot(train_df["date"], train_df["price"], label="Train")
    plt.plot(test_df["date"], test_df["price"], label="Test")
    plt.plot(test_df["date"], pred, label="Predicción")
    plt.title(titulo)
    plt.xlabel("Fecha")
    plt.ylabel("Precio")
    plt.legend()
    plt.tight_layout()
    plt.show()
```

## OPCIÓN TOMADA PARA ESTA PRUEBA

Se evaluaron los modelos ARMA, ARIMA y SARIMA mediante una validación temporal (80% entrenamiento y 20% prueba) utilizando la métrica RMSE. Los resultados muestran que SARIMA presenta el menor error en las materias primas "X" y "Z", mientras que ARMA obtiene el mejor desempeño en la materia prima "Y".

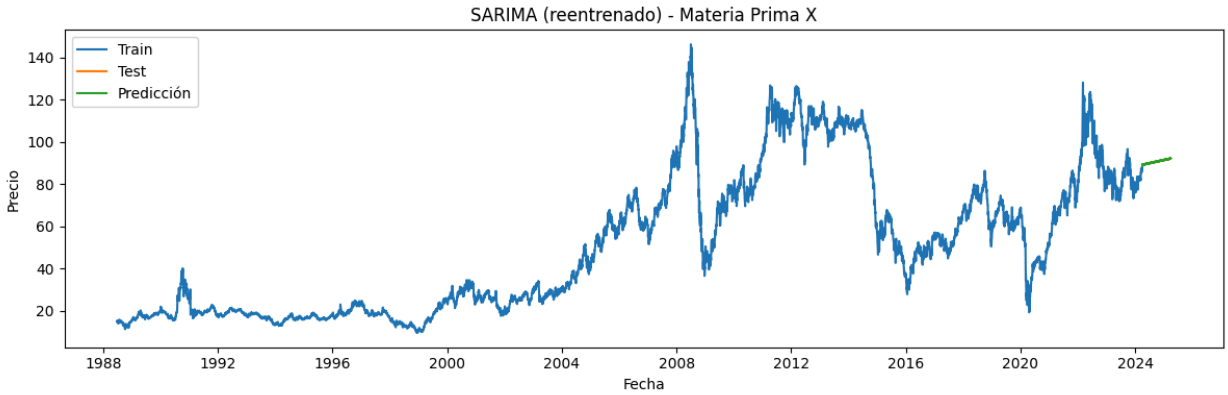
A pesar de que SARIMA no fue el mejor modelo en todas las series, se selecciona como el enfoque principal debido a su mejor desempeño global y a su capacidad para modelar componentes estacionales, los cuales son de gran ayuda dados los comportamientos cíclicos observados en el análisis exploratorio. Se reconoce que una optimización de hiper parámetros podría mejorar el desempeño de SARIMA en las materias primas en general, pero especialmente en "Y".

## RESULTADOS DEL ANÁLISIS DE LOS DATOS Y LOS MODELOS

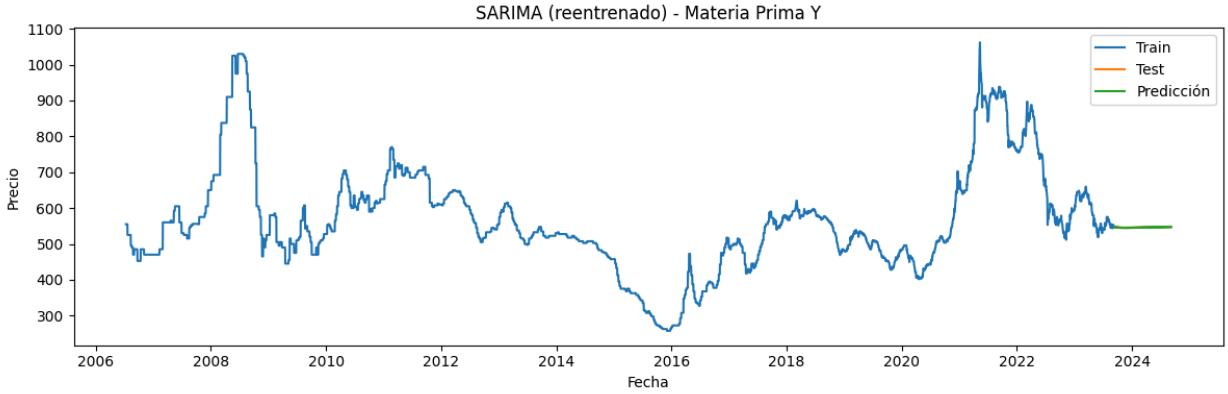
Una vez elegido el modelo a utilizar, en este caso será SARIMA, el enfoque que se usará será reentrenar el modelo SARIMA, pero, ahora será con todos los datos históricos, con el objetivo de obtener una mejor proyección hacia el futuro. Realizaremos una predicción hacía 1 año de cada materia prima para dar un margen de inicio de proyecto y con esta información podremos determinar el valor base de los 2 equipos.

A continuación, se presentan las series temporales junto con la predicción hace 1 año al futuro de cada materia prima.

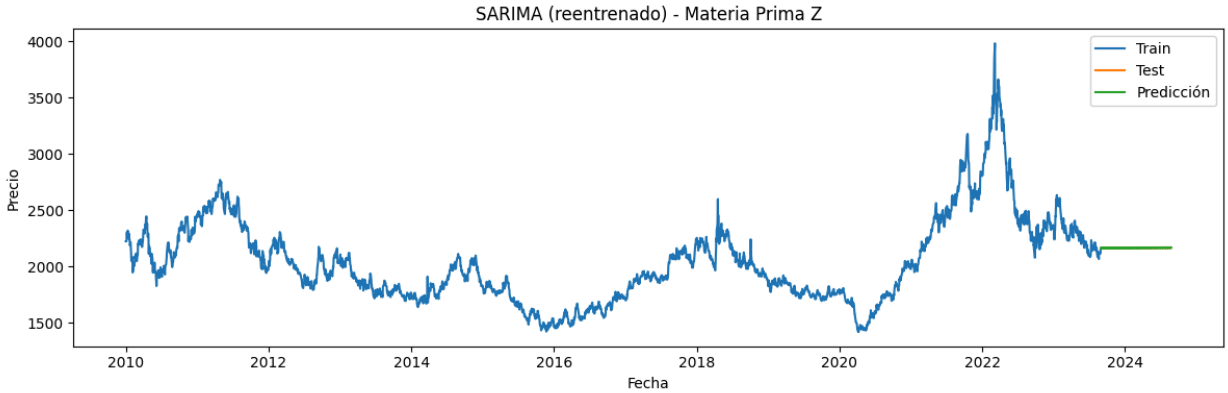
## 1. SARIMA materia prima X




## 2. SARIMA materia prima Y




## 3. SARIMA materia prima Z




A partir de las proyecciones generadas por el modelo SARIMA re-entrenado con el 100% de los datos históricos, se construyeron tres escenarios de precios para cada materia prima. El escenario **promedio** se definió como el valor promedio de las proyecciones, mientras que los

**mejor** escenario y **peor** escenario se obtuvieron a partir de los percentiles 25 y 75, respectivamente. Estos escenarios permiten esperar no solo un resultado del proceso de predicción y proporcionan un rango de referencia para la estimación de costos de los equipos bajo distintos contextos. Además, se proporciona en cada escenario el aumento porcentual del precio de cada equipo en comparación de generar el costo base con los últimos precios reportados.

Tomando en cuenta las fórmulas propuestas desde el inicio para generar un costo base de cada equipo:

- Equipo 1 = $(Precio\_PrimaX \ * \ 0.2) \ + \ (Precio\_PrimaY \ * \ 0.8)$

- Equipo 2 = $\frac{Precio\_PrimaX \ + Precio\_PrimaY + Precio\_PrimaZ}{3}$

Entonces, si tomamos en cuenta de que, los últimos precios de cada materia prima son:

- **Materia prima X (04/04/2024)**: $89.18
- **Materia prima Y (12/09/2023):** $547.33
- **Materia prima Z (31/08/2023):** $2,165.25

El costo base de cada equipo con los últimos valores de cada materia prima quedarían como:

- **Costo equipo 1**: $455.7
- **Costo equipo 2**: $933.92

Ahora, por último, generamos una tabla para reportar las diferencias entre los resultados predichos y los costos que se generaron tomando en cuenta únicamente el último precio reportado:

| Escenario | Equipo 1 | Equipo 2 | Diferencia porcentual con respecto al costo tomando en cuenta el último precio reportado equipo 1 | Diferencia porcentual con respecto al costo tomando en cuenta el último precio reportado equipo 2 |
|---|---|---|---|---|
| **Mejor escenario** | 454.31 | 933.91 | -0.3% | +0.001% |
| **Escenario Promedio** | 455.02 | 934.77 | -0.1% | +0.09% |
| **Peor escenario** | 455.72 | 935.66 | +0.004% | +0.18% |

El entrenamiento realizado con todos los datos históricos para el modelo SARIMA se realizó con las mismas funciones que se presentaron anteriormente, para la visualización de los datos históricos con la predicción final del último año se usó la función “construir_df_forecast” para modificar el ‘dataframe’ y poder agregar el último año predicho, esta función solo recibe los datos históricos y las predicciones. Por último para generar los 3 escenarios propuestos, se usan las funciones “escenarios_desde_forecast”, esta funciona solo para agregar los percentiles a un diccionario de datos, esta función sólo recibe las predicciones, y la función “costos_equipos_por_escenario”, la cual sirve para calcular el costo final de cada equipo usando los valores devueltos por la función anterior.

```python
#Construcción de df para poder una visualización de la serie de tiempo
def construir_df_forecast(serie_full, forecast):
    last_date = serie_full.index.max()
    future_dates = pd.date_range(start=last_date + pd.Timedelta(days=1), periods=len(forecast), freq="D")
    return pd.DataFrame({"date": future_dates, "price": forecast.values})

#Construcción de los 3 escenarios posibles
def escenarios_desde_forecast(forecast):
    valores = np.array(forecast)
    return {
        "optimista": np.percentile(valores, 25),
        "promedio": np.mean(valores),
        "conservador": np.percentile(valores, 75)
    }

#Costos base finales con los 3 escenarios posibles
def costos_equipos_por_escenario(esc_x, esc_y, esc_z):
    resultados = {}
    for escenario in ["optimista", "promedio", "conservador"]:
        px, py, pz = esc_x[escenario], esc_y[escenario], esc_z[escenario]
        resultados[escenario] = {
            "equipo_1": 0.2 * px + 0.8 * py,
            "equipo_2": (px + py + pz) / 3
        }
    return resultados
```

Para finalizar la presentación del código, se muestra el main, en donde se hace uso de todas las funciones para los 3 archivos de materias primas, “X”, “Y” y “Z”.

```python
from funciones import *

# Paths
PATHS = {
    "X": "Datos/X.csv",
    "Y": "Datos/Y.csv",
    "Z": "Datos/Z.csv"
}

def main(PATHS):
    # ————————1: Carga————————
    dfs = {
        "X": cargar_y_normalizar_csv(PATHS["X"]),
        "Y": cargar_y_normalizar_csv(PATHS["Y"], sep=";", decimal=",", dayfirst=True),
        "Z": cargar_y_normalizar_csv(PATHS["Z"])
    }

    # ————————2: EDA————————
    for k, df in dfs.items():
        resumen_estadistico(df, k)
        plot_serie_temporal(df, k)

    boxplots_precios([dfs["X"], dfs["Y"], dfs["Z"]], ["X", "Y", "Z"])

    # ————————3: Split ————————
    splits = {}
    for k, df in dfs.items():
        train, test = division_train_test(df)
        splits[k] = {"train": train, "test": test}
```

```python
# ————————4: Modelos————————
modelos = {}
preds = {}
rmses = {}

for k in ["X", "Y", "Z"]:
    train_df = splits[k]["train"]
    test_df = splits[k]["test"]

    entrada = train_df["price"]
    h = len(test_df)

    arma = entrenar_arma(entrada)
    arima = entrenar_arima(entrada)
    sarima = entrenar_sarima(entrada)

    pred_arma = forecast_model(arma, h)
    pred_arima = forecast_model(arima, h)
    pred_sarima = forecast_model(sarima, h)

    modelos[k] = {"arma": arma, "arima": arima, "sarima": sarima}
    preds[k] = {"arma": pred_arma, "arima": pred_arima, "sarima": pred_sarima}

    rmses[k] = {
        "arma": rmse(test_df["price"], pred_arma),
        "arima": rmse(test_df["price"], pred_arima),
        "sarima": rmse(test_df["price"], pred_sarima),
    }

    print(f"RMSE {k}:",
          rmses[k]["arma"],
          rmses[k]["arima"],
          rmses[k]["sarima"])

    # Visualizaciones
    plot_train_test_pred(train_df, test_df, pred_arma, f"ARMA - Materia Prima {k}")
    plot_train_test_pred(train_df, test_df, pred_arima, f"ARIMA - Materia Prima {k}")
    plot_train_test_pred(train_df, test_df, pred_sarima, f"SARIMA - Materia Prima {k}")
```

```python
# ————————5: Reentrenar SARIMA con datos históricos————————
    forecasts_full = {}
    historicos = {}

    for k, df in dfs.items():
        serie_full = df.sort_values("date").set_index("date")["price"]
        sarima_full = entrenar_sarima(serie_full)

        h = 360  # 1 año
        forecast = sarima_full.forecast(steps=h)

        historicos[k] = df.sort_values("date")[["date", "price"]]
        forecasts_full[k] = {
            "serie_full": serie_full,
            "forecast": forecast,
            "df_forecast": construir_df_forecast(serie_full, forecast)
        }

        plot_train_test_pred(
            historicos[k],
            forecasts_full[k]["df_forecast"],
            forecasts_full[k]["df_forecast"]["price"],
            f"SARIMA (reentrenado) - Materia Prima {k}"
        )

    # ————————6: Escenarios————————
    esc = {
        k: escenarios_desde_forecast(forecasts_full[k]["forecast"])
        for k in ["X", "Y", "Z"]
    }

    resultados_costos = costos_equipos_por_escenario(
        esc["X"], esc["Y"], esc["Z"]
    )

    df_resultados = (
        pd.DataFrame(resultados_costos)
        .T.reset_index()
        .rename(columns={"index": "escenario"})
    )

    return df_resultados


if __name__ == "__main__":
    resultados = main(PATHS)
    print(resultados)
```

## Conclusiones

A partir de las proyecciones generadas por el modelo SARIMA, se calcularon tres escenarios de costos (optimista, promedio y pesimista) para ambos equipos y se compararon con los costos obtenidos utilizando únicamente el último precio reportado de cada materia prima.

Los resultados muestran que las diferencias porcentuales entre ambos enfoques son reducidas para el horizonte de proyección considerado, esto puede deberse principalmente a que, no se experimentó con los hiper parámetros a la hora de entrenar el modelo SARIMA con los datos, sin embargo esto sugiere que el último precio observado constituye una aproximación razonable del costo base en el muy corto plazo. No obstante, el uso de escenarios permite incorporar explícitamente la incertidumbre asociada a la proyección de precios y proporciona un rango de referencia útil para la planeación financiera bajo distintos contextos de mercado.

## FUTUROS AJUSTES O MEJORAS

Si bien la metodología propuesta permite estimar de manera reproducible los costos base de los equipos a partir de datos históricos de precios, si existen varias áreas de mejora visibles que podrían explorarse. En primer lugar, se podría realizar una optimización de los hiper parámetros de los modelos ARIMA y SARIMA, además de evaluar el modelo mediante criterios de información (AIC/BIC) o aplicar un entrenamiento con técnicas de validación cruzada, con el fin de mejorar el desempeño predictivo de los modelos. Asimismo, sería pertinente evaluar horizontes de proyección más amplios y analizar la sensibilidad de los costos ante escenarios de mayor volatilidad.

Además, el análisis podría complementarse mediante la incorporación de más variables relevantes para la formación de precios de las materias primas, lo que permitiría capturar de mejor manera esos picos de precio en las series. También, se podrían comparar los modelos clásicos de series temporales con enfoques basados en deep learning (por ejemplo, redes LSTM), especialmente si en dado caso, se cuenten con datasets más grandes.