
Estimación de costos de Equipos para Proyecto de Construcción usando Machine Learning

Juan Peñaloza Quintana

Abstract Se ha requerido estimar los costos de dos equipos esenciales para un proyecto de construcción, con una duración de 36 meses. El cliente (empresa constructora) debe proporcionar los equipos necesarios, y el análisis se centra en la estimación del precio futuro de dos equipos en específico, cuyos precios dependen del valor de mercado de las materias primas X , Y , Z . El equipo 1 está compuesto en un 20% por la materia prima X y un 80% por la materia prima Y . El equipo 2 está compuesto por iguales proporciones de estas. El objetivo de este estudio es optimizar el monto de inversión que la empresa constructora debe realizar en la adquisición de los equipos en el futuro. Para esto, se usarán técnicas de análisis de series temporales (Teoría: Ver Anexo 1.) para predecir los precios de las materias primas X , Y y Z en los próximos 36 meses. Se reporta como resultado que el momento óptimo para la compra de los equipos es aproximadamente en el mes 12, contando desde la actualidad. Si los equipos se requieren inmediatamente, se puede gestionar con mecanismos como Contratos de Ajuste de Precios¹ o Pago diferido

Supuestos

- Se cuenta con datos históricos diarios de precios de las materias primas X , Y y Z .
- Los datos históricos se recopilan desde junio 1988, noviembre 2006 y enero 2010 para X , Y y Z , respectivamente.
- Los datos históricos se recopilan hasta abril 2024, diciembre 2023 y agosto 2023 para X , Y y Z , respectivamente.
- Cada materia prima es susceptible a distinta volatilidad².

Metodología

- **Preprocesamiento de los datos:** Los datos históricos fueron analizados y preparados, asegurando que no hubiese valores faltantes ni inconsistencias. Se estandarizó el formato de las fechas como . YYYY-MM-DD y los separadores decimales.
- **Trabajo en la suite de Azure:** Se utilizó la suite de Azure para el análisis de series temporales de la materia prima X . Se obtuvo un modelo ARIMA ajustado y

1. Greene 2018.

2. Asteriou and Hall 2011.

luego, en un entorno local, se llevaron a cabo las predicciones para los próximos 36 meses.

- **Trabajo en entorno local:** Para hacer forecasting de las materias primas Y y Z , se utilizó el modelo XGBoost Hyndman and Athanasopoulos 2018, el cual permite trabajar con series temporales no estacionarias y con múltiples variables predictoras.
- **Evaluar resultados:** Con medidas de dispersión como el error cuadrático medio (MSE) y el error absoluto medio (MAE)³, se evaluaron las predicciones obtenidas.
- **Consideraciones finales:** Desarrollo de las conclusiones y consideración de las posibles modificaciones futuras en pos de enriquecer la exactitud y precisión de este análisis.

Análisis

Preprocesamiento de los datos

En primer lugar, los datasets de las materias primas X , Y y Z diferían en el formato de las fechas, orden de las columnas, y separadores del formato CSV. En un notebook (para el código completo Ver **Anexo 2**) se realiza el preprocesamiento necesario para que estén listos para ser trabajados

```
>>>
df1 = pd.read_csv('Datos/X.csv')
df2 = pd.read_csv('Datos/Y.csv', delimiter=';')
df3 = pd.read_csv('Datos/Z.csv')
# ...
df1 = df1.iloc[::-1]
# resetear el indice de df1
df1 = df1.reset_index(drop=True)
# ...
df2['Date'] = df2['Date'].str.replace('/', '-')
df2['Price'] = df2['Price'].str.replace(',', '.').astype(float)
df2['Date'] = pd.to_datetime(df2['Date'])
df2['Date'] = pd.to_datetime(df2['Date'],
    format='%d/%m/%Y').dt.strftime('%Y-%m-%d')
<<<
```

Entrenamiento en Azure Machine Learning

Azure cloud tiene una serie de herramientas que van desde no-code hasta ejecución de scripts locales. En este caso, se utilizó ML Studio, el cual permite el entrenamiento

3. Hyndman and Athanasopoulos 2018.

y evaluación con *multiples modelos* y distingue el mejor a partir de la métrica de evaluación seleccionada por el desarrollador. Se requiere una suscripción de azure, en la cual se creará un workspace⁴ en el que estarán los recursos a usar (almacenamiento, cómputo, etc).

La carga de los datasets, y almacenamiento de los datasets, el aprovisionamiento de los recursos de cómputo, monitoreo de trabajos, etc; está en el (**Anexo 3**). El procedimiento lo desarrollé en la interfaz gráfica de Azure AutoML, donde se selecciona el dataset, se elige la variable objetivo, se selecciona el tipo de predicción y se lanzan los experimentos; pero se genera el código que se ejecuta en el entorno de Azure. A continuación presento un extracto.

```
>>>
# Import the required libraries
from azure.identity import DefaultAzureCredential
from azure.ai.ml import MLClient
from azure.ai.ml.entities import AmlCompute

# The workspace information from the previous experiment has been pre-filled
subscription_id = "b0108ced-f6be-4641-917c-8e9e1cf2c8d9"
resource_group = "modelo_ARIMA"
workspace_name = "workspace_arima_3011"

credential = DefaultAzureCredential()
ml_client = MLClient(credential, subscription_id, resource_group, workspace_name)
workspace = ml_client.workspaces.get(name=ml_client.workspace_name)
print(ml_client.workspace_name, workspace.resource_group, workspace.location)
    ml_client.connections._subscription_id, sep = '\n')
#
# ...
# Choose a name for your CPU cluster
cluster_name = "cpu-cluster"

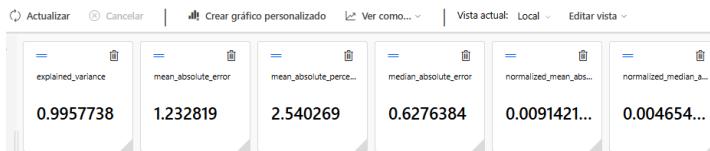
# Verify that cluster does not exist already
try:
    cluster = ml_client.compute.get(cluster_name)
    print('Found existing cluster, use it.')
except Exception:
    compute = AmlCompute(name=cluster_name, size='STANDARD_DS4_V2',
                          max_instances=4)
    cluster = ml_client.compute.begin_create_or_update(compute)
#
from azure.ai.ml import command, Input
```

4. <https://learn.microsoft.com/en-us/azure/machine-learning/quickstart-create-resources>

```
# To test with new training / validation datasets, replace
# the default dataset id(s)/uri(s) taken from parent run below
command_str = 'python script.py --training_dataset_uri azureml://locations
command_job = command(
    code=project_folder,
    command=command_str,
    tags=dict(automl_child_run_id='exp-98-r2-para-x_2'),
    environment='AzureML-ai-ml-automl:7',
    compute='cpu-cluster',
    experiment_name='Default')

returned_job = ml_client.create_or_update(command_job)
returned_job.studio_url
<<<
```

Luego de los experimentos, se obtiene el mejor modelo, que en este caso fue Arimax. Las métricas principales se muestran en la figura:



Notes: El modelo puede descargarse desde Azure para despliegue o uso en local

FIGURE 1. El valor de mean absolute error (MAE) representa un porcentaje pequeño de las magnitudes de los precios de la materia prima X.

Trabajo en entorno local

El entrenamiento para las materias primas Y y Z se realizó en notebooks ejecutados en un entorno local. Se utilizó la clase `XGBRegressor` de la librería `xgboost` para entrenar los modelos y hacer las predicciones, a través del método **recursive forecasting**⁵. A continuación, se muestra un extracto del código utilizado para el entrenamiento y la predicción de los precios de la materia prima Y. El código para z es equivalente.

>>>

5. <https://cienciadedatos.net/documentos/py56-forecasting-time-series-with-xgboost>

```
% from sklearn.model_selection import train_test_split
% from sklearn.metrics import mean_squared_error
% import xgboost as xgb
% from sklearn.preprocessing import StandardScaler
# ...
df['Date'] = pd.to_datetime(df['Date'])

# Establecer la columna 'fecha' como índice
df.set_index('Date', inplace=True)
# ...
# Crear características de serie temporal (lag features)
def create_lag_features(df, lags=5):
    for lag in range(1, lags + 1):
        df[f'lag_{lag}'] = df['Price'].shift(lag)
    df = df.dropna() # Eliminar filas con valores nulos
    return df

# Crear características de lag
df = create_lag_features(df, lags=5)
# Dividir los datos en conjunto de entrenamiento y prueba
train_size = int(len(df) * 0.7)
train, test = df.iloc[:train_size], df.iloc[train_size:]
# Separar las características (X) y el target (y)
X_train = train.drop(columns=['Price'])
y_train = train['Price']
X_test = test.drop(columns=['Price'])
y_test = test['Price']
# ...
# Create forecaster
# =====
end_validation = '2023-12-09'

window_features = RollingFeatures(
    stats=["coef_variation"],
    window_sizes=4200)
forecaster = ForecasterRecursive(
    regressor      = XGBRegressor(random_state=15926,
                                   enable_categorical=True),
    lags          = 3000,
    window_features = window_features
)
```

6 Prueba Técnica DataKnow

```
# Train forecaster
# =====
forecaster.fit(y=df.loc[:end_validation, 'Price'])

# Calcular el error cuadrático medio (MSE) -> El resultado fue 61.2
mse = mean_squared_error(y_train, predictions)
print(f"Error cuadrático medio (MSE): {mse}")

# predicción_año_n = forecaster.predict(steps=(365*n))
<<<
```

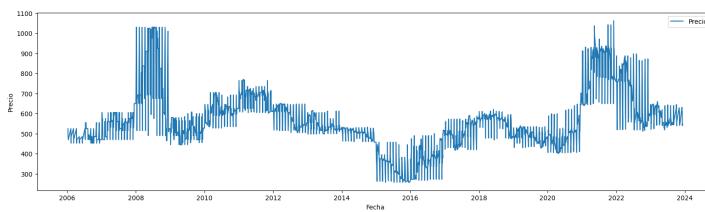
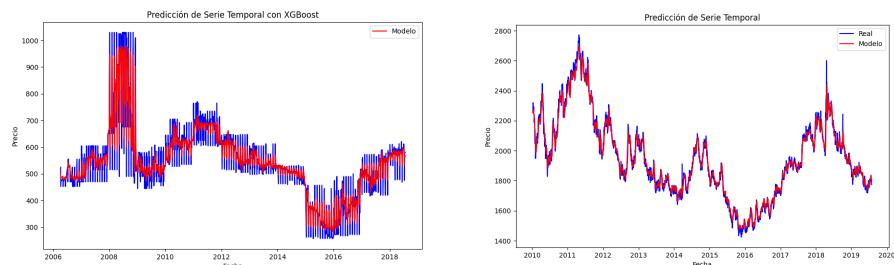


FIGURE 2. Las variaciones bruscas del precio que se ven en esta figura explican que el mse (error cuadrático medio) obtenida fue de 61.2, lo cual es un valor relativamente alto para los valores registrados como precio en el dataset de Y.

Veamos ahora la comparación de las predicciones realizadas para los días registrados en el dataset con los valores históricos, para las materias primas Y y Z (Para el gráfico del precio para X ver Anexo 2.).



En secciones posteriores se discutirán maneras técnicas de abordar series temporales con características de alta volatilidad, como la de la materia prima Y

Con la suficiente cantidad de arboles de estimación en la definición del algoritmo, los resultados se ciñen a los registros para Z.
Nota: Cuidar el sobreajuste

Teniendo los modelos entrenados podemos hacer las predicciones para los próximos 36 meses. Teniendo en cuenta que un dataset tiene datos hasta 2023 mientras que los otros dos tienen datos hasta 2024 (año actual), en dicho modelo se harán las predicciones para 'sus' próximos 48 meses. En la siguiente tabla se consignan los valores obtenidos para los meses 0 (actualidad), 12, 24 y 36.

TABLE 1

Materia Prima	<i>Actualidad</i>	<i>12 meses</i>	<i>24 meses</i>	<i>36 meses</i>
$X \pm 1.73$	89.2	82.8	88.2	79.5
$Y \pm 12.4$	547.3	614.1	589.3	568.9
$Z \pm 32.49$	2165.3	1834.1	2246.7	2483.5

Notes: La medida de confianza que se usa para reportar cada predicción es el intervalo de predicción, que se calcula a partir de la **desviación estándar de los residuos** (Ver Anexo 4) y el *mse* de cada modelo.

Veamos ahora el calculo de los costos de los equipos 1 y 2 para el futuro (tabla: 2).

$$\text{Costo Equipo 1} = \frac{20}{100} \cdot \text{Precio } X + \frac{80}{100} \cdot \text{Precio } Y$$

$$\text{Costo Equipo 2} = \frac{1}{3} \left(\text{Precio } X + \text{Precio } Y + \text{Precio } Z \right)$$

TABLE 2

	<i>Actualidad</i>	<i>12 meses</i>	<i>24 meses</i>	<i>36 meses</i>
Equipo 1	455.7 ± 13.4	507.8 ± 12.5	489.08 ± 15.2	471.05 ± 11.47
Equipo 2	933.92 ± 76.3	1023.5 ± 87.3	1000.2 ± 83.5	978.5 ± 81.1
TOTAL	1389.62 ± 32.66	1351.5 ± 36.54	1463.83 ± 38.12	1515.4 ± 43.82

Notes: Los \pm se calculan usando la fórmula que involucra las derivadas parciales de la función de costo con respecto a cada variable, después de hacer una multiplicación que considere no solo la incertidumbre de la materia prima sino también las magnitudes de cada punto de datos involucrado.

Consideraciones finales

En conclusión, el momento óptimo para la adquisición de los equipos es en 12 meses de desarrollo del proyecto, pues los costos de los equipos 1 y 2 son menores en comparación con los costos actuales y futuros. Sin embargo, es importante tener en cuenta que las predicciones realizadas tienen un margen de error, por lo que se recomienda monitorear los precios de las materias primas⁶ y ajustar las estrategias de inversión en consecuencia. Por ejemplo, reentrenar los modelos en el transcurso del futuro. También se podría agregar una medida de la inflación prevista por el banco mundial (u organismos similares) en el país donde se desarrolla el proyecto; además de una medida de la volatilidad del mercado⁷ (Histórica y esperada) para las materias primas.

Conclusión

El análisis de series temporales abarca una amplia variedad de métodos, desde los tradicionales enfoques estadísticos (como ARIMA o GARCH) hasta técnicas más modernas de aprendizaje automático (como redes neuronales y árboles de decisión). La elección del enfoque depende de la naturaleza de los datos⁸, los objetivos del análisis (predicción, comprensión de patrones) y las características específicas⁹ de la serie temporal que se está analizando (estacionalidad, volatilidad, tendencia, etc.).

6. Linton 2013.

7. Andersen et al. 2003.

8. Campbell, Lo, and MacKinlay 1997.

9. Shumway and Stoffer 2017.

Supplementary Material

(Más contenido relevante:

<<https://cienciadedatos.net/documentos/py56-forecasting-time-series-with-xgboost>>

References

- Andersen, Torben G., Tim Bollerslev, Francis X. Diebold, and Paul Labys. 2003. Measuring Financial Volatility. *Review of Economics and Statistics* 85 (4):793–808.
- Asteriou, Dimitrios, and Stephen G. Hall. 2011. *Applied Econometrics*. 4th. Palgrave Macmillan.
- Campbell, John Y., Andrew W. Lo, and A. Craig MacKinlay. 1997. *The Econometrics of Financial Markets*. 2nd. Princeton University Press.
- Greene, William H. 2018. *Econometric Analysis*. 8th. Pearson.
- Hyndman, Rob J., and George Athanasopoulos. 2018. *Forecasting: Principles and Practice*. OTexts.
- Linton, Oliver. 2013. *Financial Econometrics: A Re-Appraisal*. Cambridge University Press.
- Shumway, Robert H., and David S. Stoffer. 2017. *Time Series Analysis and Its Applications: With R Examples*. 4th. Springer.

Anexo 1.

El estudio de series temporales es un tema extenso, por lo tanto se tratará solamente modelos de Machine Learning: XGBoost y ARIMAX. XGBoost es un algoritmo de aprendizaje automático basado en árboles de decisión que se utiliza comúnmente para problemas de regresión. Su formalización matemática se basa en la minimización de una función de pérdida, de la siguiente forma:

$$\mathcal{L} = \sum_{i=1}^n L(y_i, \hat{y}_i) + \sum_{k=1}^K \Omega(h_k)$$

XGBoost utiliza una aproximación de segundo orden (incluyendo derivadas primeras y segundas) para optimizar la función de pérdida. Para cada iteración k , se calcula la mejora γ_k que minimiza la función objetivo:

$$\gamma_k = \arg \min_{\gamma} \left[\sum_{i \in I_k} L(y_i, F_{k-1}(x_i) + \gamma) \right] + \Omega(\gamma)$$

XGBoost construye árboles de decisión de manera aditiva. Cada árbol nuevo se agrega al modelo existente para corregir los errores residuales. La construcción de cada árbol implica:

División de Nodos: Selección de la mejor característica y punto de división que maximice la ganancia de información.

Asignación de Pesos: Determinación de los pesos óptimos para cada hoja, minimizando la función objetivo localmente.

3. Poda: Eliminación de ramas que no contribuyen significativamente al modelo, según los parámetros de regularización.

Por otro lado, ARIMAX es un modelo de regresión lineal que incorpora términos autorregresivos (AR), de medias móviles (MA) y diferenciación (I) para modelar series temporales. Antes de profundizar en ARIMAX, es esencial comprender los componentes básicos de ARIMA:

- AR (AutoRegressive): Captura la relación lineal entre una observación actual y un número de observaciones anteriores. - I (Integrated): Indica el número de veces que la serie debe ser diferenciada para alcanzar la estacionariedad.

- MA (Moving Average): Modela el error de predicción como una combinación lineal de errores pasados. El modelo ARIMAX puede expresarse como:

$$\Phi(B)(1 - B)^d Y_t = \Theta(B)\epsilon_t + \beta_1 X_{1,t} + \beta_2 X_{2,t} + \cdots + \beta_k X_{k,t}$$

Donde:

- B es el operador de rezago ($BY_t = Y_{t-1}$).
- $\Phi(B) = 1 - \phi_1 B - \phi_2 B^2 - \cdots - \phi_p B^p$ representa la componente autorregresiva.
- $\Theta(B) = 1 + \theta_1 B + \theta_2 B^2 + \cdots + \theta_q B^q$ representa la parte de media móvil.
- ϵ_t es el término de error aleatorio, generalmente asumido como ruido blanco ($\epsilon_t \sim N(0, \sigma^2)$).
- $\beta_1, \beta_2, \dots, \beta_k$ son los coeficientes asociados a las variables exógenas $X_{1,t}, X_{2,t}, \dots, X_{k,t}$.

ARIMAX es usado en multiples campos, tales como:

- **Economía:** Modelado de indicadores macroeconómicos influenciados por variables exógenas como tasas de interés.
- **Marketing:** Predicción de ventas basada en campañas publicitarias u otras variables de marketing.
- **Marketing:** Predicción de ventas basada en campañas publicitarias u otras variables de marketing.
- **Ingeniería:** Monitoreo de procesos industriales con influencias externas.

Anexo 2.

El preprocesado de los datos se realizó con el siguiente código (completo):

```
>>>
# leer archivos (3 excels) contenidos en la carpeta 'datos'
df1 = pd.read_csv('Datos/X.csv')
df2 = pd.read_csv('Datos/Y.csv', delimiter=';')
df3 = pd.read_csv('Datos/Z.csv')
# ...
df1 = df1.iloc[::-1]
# resetear el indice de df1
df1 = df1.reset_index(drop=True)

# cambiar los / por - en la columna 'Date' y convertir la columna 'Price'
df2['Date'] = df2['Date'].str.replace('/', '-')
df2['Price'] = df2['Price'].str.replace(',', '.').astype(float)

# convertir la columna 'Date' a tipo datetime
df2['Date'] = pd.to_datetime(df2['Date'])
# pasar del formato dd/mm/yyyy a yyyy-mm-dd
df2['Date'] = pd.to_datetime(df2['Date'], format='%d/%m/%Y').dt.strftime('%Y-%m-%d')

# exportar los dataframes a archivos csv
df1.to_csv('Datos/X_modificado.csv', index=True)
df2.to_csv('Datos/Y_modificado.csv', index=True)
df3.to_csv('Datos/Z_modificado.csv', index=True)
<<<
```

Veamos con mejor tamaño la gráfica de ajuste del modelo XGBoost a la variable Y , a pesar de que esto no corrija la presencia de cambios abruptos.

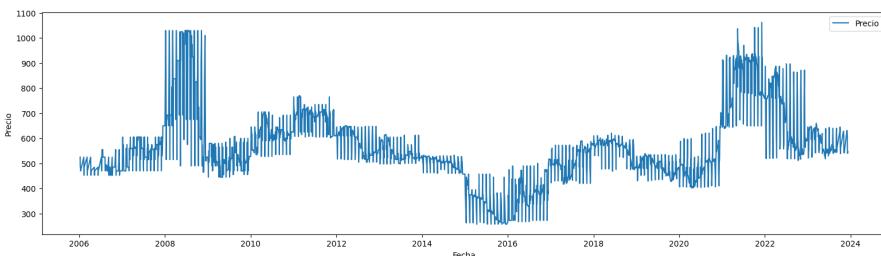
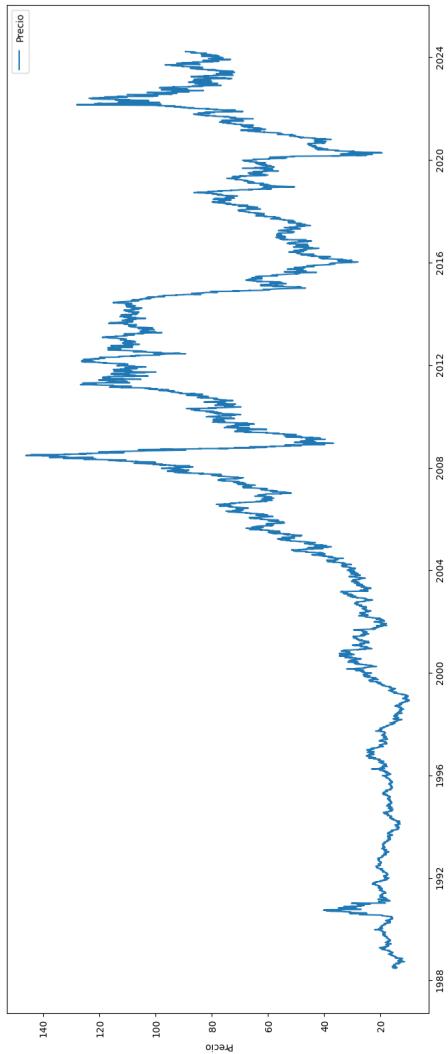


FIGURE 3



Notes:

FIGURE 4. Con el fin de ahorrar espacio en el análisis se muestra aquí, como apéndice, el registro histórico de precios para la variable X.

Anexo 3.

1 Train using Azure Machine Learning Compute

- Connect to an Azure Machine Learning Workspace
- Use existing compute target or create new
- Configure & Run command

1.1 Prerequisites

Please ensure Azure Machine Learning Python SDK v2 is installed on the machine running Jupyter.

1.2 Connect to a Workspace

Initialize a workspace object from the previous experiment.

```
[ ]: # Import the required libraries
from azure.identity import DefaultAzureCredential
from azure.ai.ml import MLClient

# The workspace information from the previous experiment has been pre-filled for you.
subscription_id = "b0108ced-f6be-4641-917c-8e9e1cf2c8d9"
resource_group = "modelo_ARIMA"
workspace_name = "workspace_arima_3011"

credential = DefaultAzureCredential()
ml_client = MLClient(credential, subscription_id, resource_group, workspace_name)
workspace = ml_client.workspaces.get(name=ml_client.workspace_name)
print(ml_client.workspace_name, workspace.resource_group, workspace.location, ml_client.connections_subscription_id, sep = '\n')
```

1.2.1 Create project directory

Create a directory that will contain the training script that you will need access to on the remote resource.

```
[ ]: import os
import shutil

project_folder = os.path.join(".", "code_folder")
os.makedirs(project_folder, exist_ok=True)
shutil.copy('script.py', project_folder)
```

1.2.2 Use existing compute target or create new (Basic)

Azure Machine Learning Compute is managed compute infrastructure that allows the user to easily create single to multi-node compute of the appropriate VM Family. It is created **within your workspace region** and is a resource that can be used by other users in your workspace. It autoscales by default to the max_nodes, when a job is submitted, and executes in a containerized environment packaging the dependencies as specified by the user.

Since it is managed compute, job scheduling and cluster management are handled internally by Azure Machine Learning service.

A compute cluster can be created using the `AmlCompute` class. Some of the key parameters of this class are:

- `size` - The VM size to use for the cluster. For more information, see [Supported VM series and sizes](#).
- `max_instances` - The maximum number of nodes to use on the cluster. Default is 1.

```
[ ]: from azure.ai.ml.entities import AmlCompute

# Choose a name for your CPU cluster
cluster_name = "cpu-cluster"

# Verify that cluster does not exist already
try:
    cluster = ml_client.compute.get(cluster_name)
    print("Found existing cluster, use it.")
except Exception:
    compute = AmlCompute(name=cluster_name, size='STANDARD_DS4_V2',
                          max_instances=4)
    cluster = ml_client.compute.begin_create_or_update(compute)
```

1.2.3 Configure & Run

The environment and compute has been pre-filled from the original training job. More information can be found here:

`command: https://docs.microsoft.com/en-us/python/api/azure-ai-ml/azure.ai.ml?view=azure-python-preview#azure-ai-ml-command`

`environment: https://docs.microsoft.com/en-us/azure/machine-learning/resource-curated-environments#automated-ml-automl`

16 Prueba Técnica DataKnow

```
compute: https://docs.microsoft.com/en-us/python/api/azure-ai-ml/azure.ai.ml.entities.amlcompute?view=azure-python-preview

[ ]: # To test the script with an environment referenced by a custom yaml file, uncomm
    ↵ uncomment the following lines and replace the `conda_file` value with the path to the yaml file.
    # Set the value of `environment` in the `command` job below to `env`.

    # env = Environment(
    #     name="automl-tabular-env",
    #     description="environment for automl inference",
    #     image="mcr.microsoft.com/azureml/openmpi4.1.0-ubuntu20.04:20210727.v1",
    #     conda_file="conda.yaml",
    # )

[ ]: from azure.ai.ml import command, Input

# To test with new training / validation datasets, replace the default dataset_id(s)/uri(s) taken from parent run below
command_str = 'python script.py --training_dataset_uri azurreml://locations/eastus/workspaces/036f090e-319e-4836-bac1-a5ad6f8e86a0/data/train-XA/versions/1 --validation_dataset_uri azurreml://locations/eastus/workspaces/036f090e-319e-4836-bac1-a5ad6f8e86a0/data/test-XA/versions/1'
command_job = command(
    code=project_folder,
    command=command_str,
    tags=dict(automl_child_run_id='exp-98-r2-para-x_2'),
    environment='AzureML-ai-ml-automl:7',
    compute='cpu-cluster',
    experiment_name='Default')

returned_job = ml_client.create_or_update(command_job)
returned_job.studio_url
```

1.2.4 Initialize MLFlow Client

The metrics and artifacts for the run can be accessed via the MLFlow interface. Initialize the MLFlow client here, and set the backend as Azure ML, via. the MLFlow Client.

IMPORTANT, you need to have installed the latest MLFlow packages with:

```
pip install azureml-mlflow
```

```
pip install mlflow
```

```
[ ]: # %pip install azureml-mlflow
# %pip install mlflow
```

```
[ ]: import mlflow

# Obtain the tracking URL from MLClient
MLFLOW_TRACKING_URI = ml_client.workspaces.get(
    name=ml_client.workspace_name
).mlflow_tracking_uri

# Set the MLFLOW TRACKING URI

mlflow.set_tracking_uri(MLFLOW_TRACKING_URI)

# Retrieve the metrics logged to the run.
from mlflow.tracking.client import MlflowClient

# Initialize MLFlow client
mlflow_client = MlflowClient()
mlflow_run = mlflow_client.get_run(returned_job.name)
mlflow_run.data.metrics
```

1.2.5 Download Fitted Model

Download the resulting fitted model to the local folder in `local_dir`.

```
[ ]: # import os

# Create local folder
# local_dir = "./artifact_downloads"
# if not os.path.exists(local_dir):
#     os.makedirs(local_dir)
# Download run's artifacts/outputs
# local_path = mlflow_client.download_artifacts(
#     mlflow_run.info.run_id, "outputs", local_dir# )
# print("Artifacts downloaded in: {}".format(local_path))
# print("Artifacts: {}".format(os.listdir(local_path)))
```

Anexo 4.

Las medidas de dispersión usadas fueron, básicamente, la desviación estándar y el error cuadrático medio (MSE). La desviación estándar se define como la raíz cuadrada de la varianza, y mide la dispersión de los datos alrededor de la media. Por otro lado, el MSE es una medida de la diferencia entre los valores observados y los valores predichos por un modelo.

$$\text{Desviación Estándar} = \sqrt{\frac{1}{n} \sum_{i=1}^n (x_i - \bar{x})^2}$$

$$\text{MSE} = \frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i)^2$$

En estadística, un **intervalo de predicción** proporciona un rango dentro del cual se espera que caiga una observación futura individual con una cierta probabilidad. A diferencia de los intervalos de confianza, que se utilizan para estimar parámetros poblacionales, los intervalos de predicción están diseñados para predecir valores individuales futuros.

Definición Formal

Sea Y_{nuevo} una nueva observación que se desea predecir. Un intervalo de predicción de nivel $(1 - \alpha)$ para Y_{nuevo} está definido como el intervalo $[L, U]$ tal que:

$$P(L \leq Y_{\text{nuevo}} \leq U) = 1 - \alpha$$

Donde:

- P denota la probabilidad.
- L es el límite inferior del intervalo de predicción.
- U es el límite superior del intervalo de predicción.
- α es el nivel de significancia (por ejemplo, $\alpha = 0.05$ para un intervalo de predicción del 95%).

Cálculo del Intervalo de Predicción

En el contexto de un modelo de regresión lineal, el intervalo de predicción para una nueva observación Y_{nuevo} dada una nueva variable independiente $\mathbf{x}_{\text{nuevo}}$ se calcula de la siguiente manera:

$$\hat{Y}_{\text{nuevo}} \pm t_{\alpha/2, n-p} \cdot s \cdot \sqrt{1 + \mathbf{x}_{\text{nuevo}}^\top (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{x}_{\text{nuevo}}}$$

Donde:

- \hat{Y}_{nuevo} es la predicción puntual para Y_{nuevo} .
- $t_{\alpha/2, n-p}$ es el valor crítico de la distribución t de Student con $(n - p)$ grados de libertad.

- s es la estimación de la desviación estándar del error.
- \mathbf{X} es la matriz de diseño de las variables independientes en el modelo de regresión.
- n es el número de observaciones.
- p es el número de parámetros en el modelo.

Interpretación

El intervalo de predicción proporciona un rango donde se espera que caiga una nueva observación futura con una probabilidad especificada. Es especialmente útil en aplicaciones donde se necesita predecir valores individuales más que estimar parámetros poblacionales.

Diferencias con el Intervalo de Confianza

- **Intervalo de Confianza:** Estima un parámetro poblacional (como la media) y proporciona un rango donde se espera que se encuentre el parámetro con cierta probabilidad.
 - **Intervalo de Predicción:** Predice un valor individual futuro y proporciona un rango donde se espera que caiga esa observación con cierta probabilidad.
- La implementación en python es la siguiente:

```
>>>
# usar el dataframe predictions para calcular el intervalo de confianza
model = sm.OLS(y_train, X_train).fit()
prediccion = model.get_prediction(test)
# crear un dataset que tenga la columna 'Price' con las predicciones
prediccion = prediccion.summary_frame(alpha=0.05)

# renombrar la columna mean a 'Price'
prediccion = prediccion.rename(columns={'mean': 'Price'})
# renombrar la columna obs_ci_lower a 'lower_bound'
prediccion = prediccion.rename(columns={'obs_ci_lower': 'lower_bound'})
# renombrar la columna obs_ci_upper a 'upper_bound'
prediccion = prediccion.rename(columns={'obs_ci_upper': 'upper_bound'})
# seleccionar solamente las columnas 'Price', 'lower_bound'
# y 'upper_bound'
prediccion = prediccion[['Price', 'lower_bound', 'upper_bound']]

# calculo para cada fila del dataframe el intervalo de confianza
prediccion['intervalo'] = prediccion.apply(
lambda x: (
x['Price'] - x['lower_bound'], x['upper_bound'] - x['Price']
),
axis=1)
<<<
```