PRONÓSTICO DE VENTAS DIARIAS UTILIZANDO FACEBOOK PROPHET Y XGBOOST: CASO DE UNA CAFETERÍA.

ALUMNO: RAFAEL FARÍAS
FACULTAD DE INGENIERÍA
MAGÍSTER EN DATA SCIENCE
UNIVERSIDAD DEL DESARROLLO
rfariasp@udd.cl
www.linkedin.com/in/rfariasp/

PROFESOR: JOAQUÍN VILLAGRA
FACULTAD DE INGENIERÍA
UNIVERSIDAD DEL DESARROLLO
joaquin.villagra@udd.cl
www.linkedin.com/in/joaqunv/

22 de febrero de 2021

ABSTRACT

Generar predicciones es algo de lo cual los negocios dependen en estos tiempos. El no poder pronosticar comportamientos de los indicadores aumenta la probabilidad de fracaso o de menores rentabilidades.

Es por eso que este documento busca generar el interés, en entender la información de ventas de una PYME, el caso de una cafetería, y a partir de esta, probar dos metodologías (modelos) de regresión que permitan obtener predicciones, utilizando técnicas de Machine Learning Avanzadas; cuyo objetivo general sea mejorar la rentabilidad del negocio, con el fin específico de dividir mayores utilidades entre los accionistas, la sociedad y el medio ambiente. Esto se espera lograr prediciendo niveles de ventas para poder determinar el uso de materias primas necesarias para la semana siguiente.

El dataset está conformado por ventas diarias desde Enero del año 2019 al presente. Estos datos son obtenidos de fuentes propias.

***K*eywords** Gestión · Cafeterías · Machine Learning · PYME · Prophet · XGBoost

1. INTRODUCCIÓN

Las Pymes son un pilar fundamental del desarrollo económico sustentable, porque son generadoras de riqueza, además de ser entes dinámicos que identifican, explotan y desarrollan nuevas actividades productivas que no sólo benefician a los *sharholders*[1], sino que también a todos los *stakeholders*[2].

Lamentablemente son organizaciones que no se adaptan rápidamente a las nuevas tecnologías[4], a pesar de que su planeación y organización no requiere de mucha burocracia en comparación con las grandes empresas. Estas

[1] Shareholders es aquella persona natural o jurídica que es propietaria de acciones de los distintos tipos de sociedades anónimas o comanditarias que pueden existir en el marco jurídico de cada país.

[2] Stakeholder es cualquier individuo u organización que, de alguna manera, es impactado por las acciones de determinada empresa.

organizaciones tienen que perdurar en los mercados de alta competencia y para ello deben alcanzar un desarrollo empresarial que se los permita.

Una de las fuentes de este desarrollo empresarial y tal como lo indica Andrew Ng en su artículo *AI Transformation Playbook*[1], es la Inteligencia Artificial, la cual está destinada a transformar la gestión en toda la industria.

La importancia estratégica de la información como lo señala Porter[5] se está extendiendo por toda la economía y ninguna empresa podrá escapar a sus efectos. Está afectando a todo el proceso mediante el cual una empresa crea los productos. Es más, está redefiniendo el producto en sí: el conjunto integral de bienes físicos, servicios e información con que las empresas proporcionan valor a sus clientes.

En este caso específico los esfuerzos estarán destinados a pronosticar los ingresos de los próximos siete días en una cafetería, no sólo para aumentar la eficiencia en los pedidos de materias primas, sino que además aumentar la repartición de dividendos a favor de los dueños, sociedad y el medio ambiente.

El problema será afrontado analizando el comportamiento de dos modelos predictores.

Por un lado Prophet[6], que es un acercamiento práctico para predecir a escala, que combina modelos configurables con análisis en los ciclos, para determinar el rendimiento del análisis. Este modelo utiliza series de tiempo descomponibles[3] con tres componentes principales: tendencia, estacionalidad y días festivos que se combinan en la siguiente ecuación:

$$y(t) = g(t) + s(t) + h(t) + et \tag{1}$$

En donde g (t) es la función de tendencia que modela cambios no periódicos en el valor de la serie de tiempo, s (t) representa cambios periódicos (por ejemplo, estacionalidad semanal y anual), y h (t) representa los efectos de las vacaciones que ocurren en horarios potencialmente irregulares durante uno o más días. El término de error t representa cualquier cambio idiosincrásico que no se adapta al modelo.

Por otro lado se modela la predicción a través del método de XGboost[8], el cual es una de las técnicas más utilizas para la solución de problemas de Machine Learning para varios tipos de aplicaciones[2]. El *Tree boosting* ha demostrado entregar excelentes resultados para *el estado del arte* en varias comparativas de regresiones[7].

Quizás el factor más importante detrás del éxito de XGBoost es su escalabilidad en todos los escenarios. El sistema se ejecuta cerca de diez veces más rápido que las soluciones populares existentes en una sola máquina y escala a miles de millones de ejemplos en configuraciones distribuidas o con memoria limitada, por lo que es una excelente herramienta para PYMES que no cuentan con potentes máquinas de procesamiento.

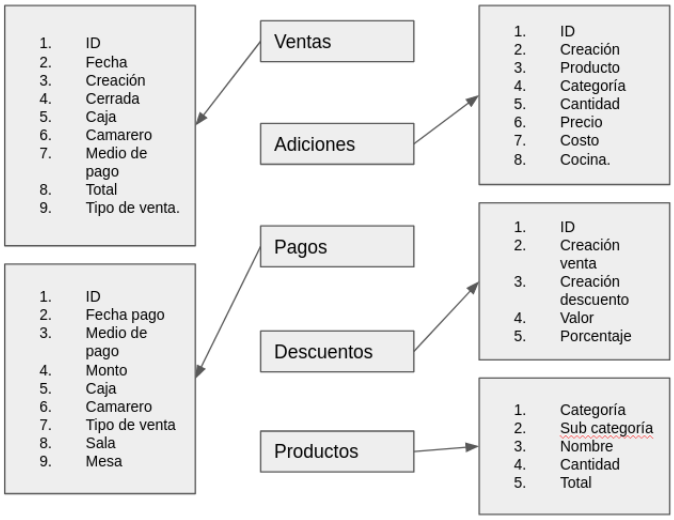
2. ACERCA DEL DATASET

Los datos a utilizar en este proyecto corresponden a información obtenida de Café Miranda Chile[3] en donde se cuenta con los datos de ventas obtenidos desde el 01 de Enero del año 2019. Este dataset no tiene datos nulos

La información concerniente a datos de ventas, se obtiene del dataset explicado en la Figura 1 y contiene dos variables continuas: Fecha y Monto Total.

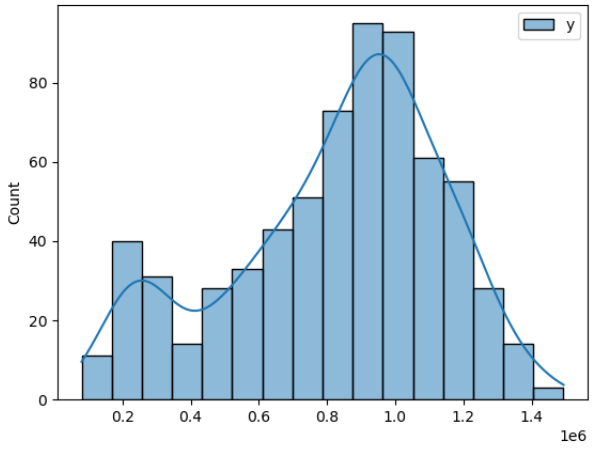
[3] http://www.cafemiranda.cl

Figura 1: Representación gráfica del Dataset principal (Elaboración propia, 2020)



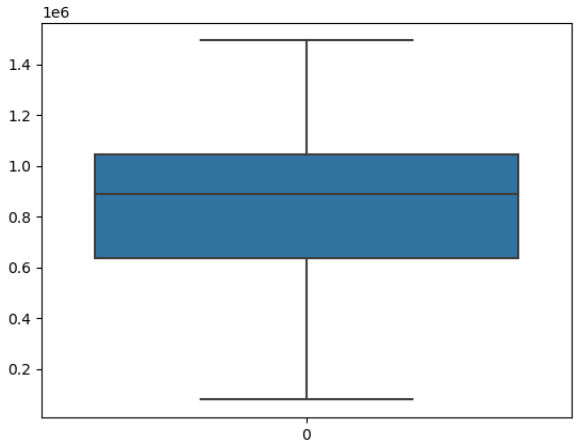
Los datos cuentan con una distribución normal tal como se muestra en la Figura 2, por lo que la mayoría de los datos se encuentran cercanos a la media. Quizás se puede observar valor que pueden ser considerados como *outliers*, sin embargo se toma la desición de no eliminarlos al ser un dataset pequeño.

Figura 2: Distribución de los datos (Elaboración propia, 2020)



Además se pueda notar en la Figura 3 que existe una diferencia importante entre el valor menor y mayor, sin embargo se conservan todos los datos para entender la tendencia principalmente relacionada a la estacionalidad semanal, es decir, la diferencias delnivel de ventas entre los distintos días de la semana. Por otro lado en este gráfico se puede notar una media cercana a los 825.000 mil pesos con un Q1 bordeando los 637.000 mil pesos y un Q3 cercano a 1.042.000 pesos.

Figura 3: Gráfico de Boxplot (Elaboración propia, 2020)



3. OBJETIVOS

Como objetivo general, se pretende estudiar el comportamiento de dos modelos de Machine Learning en la predicción de ventas para una cafetería y compararlos a través del *Mean Absolute Error*[4]

Las tareas que se realizan son: 1). Probar ambos modelos en *default*. 2). Realizar una búsqueda de los mejores hiperparámetros. 3). Probar ambos modelos con los nuevos hiperparámetros. 4). Obtener las predicciones. 5). Indicar cuál es el modelo con el menor MAE (*Mean Absolute Error*). Se escoge MAE ya que permite entender claramente el monto promedio de error en cada predicción.

4. RESULTADOS

Los experimentos realizados fueron los siguientes: 1). Prophet en *default.* 2). Prophet con hiperparámetros. 3). XGBoost en *default.* 4). XGBoost con hiperparámetros.

Los resultados obtenidos se muestran en el siguiente Cuadro 1:

Modelo	MAE
Prophet Default	191997
Prophet Hiper	161688
XGBoost Default	229543
XGBoost Hiper	213611

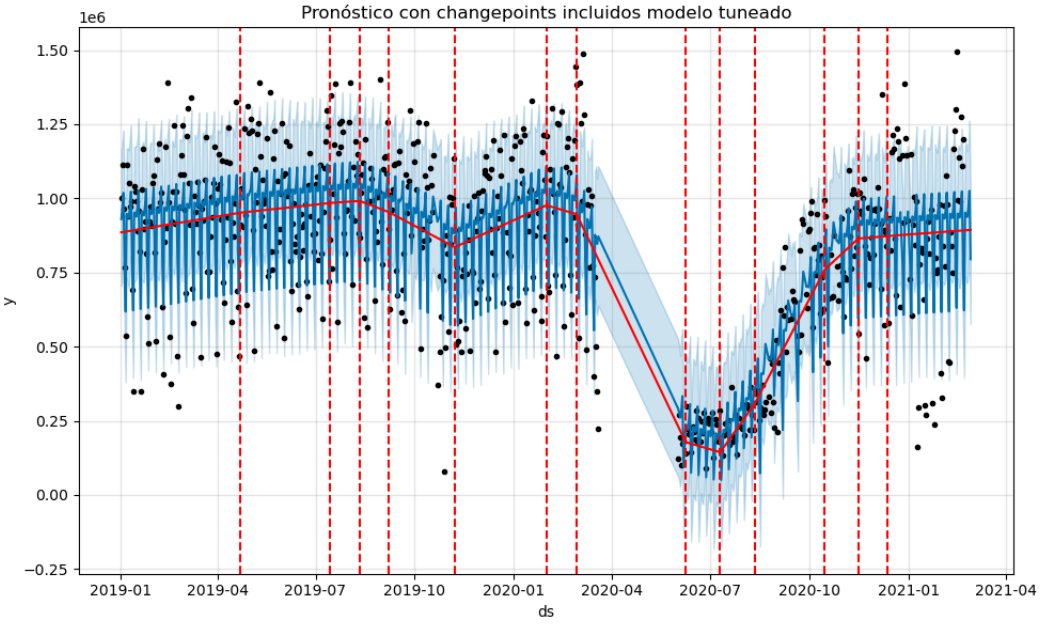
Cuadro 1: MAE de los modelos estudiados.

Se puede observar entonces que el modelo con el menor MAE es Prophet Hiper con un valor de 161688 pesos.

En la siguiente Figura 4 podemos observar el comportamiento de los datos observados en color azul y de la tendencia encontrada en color rojo del modelo Prophet Hiper.

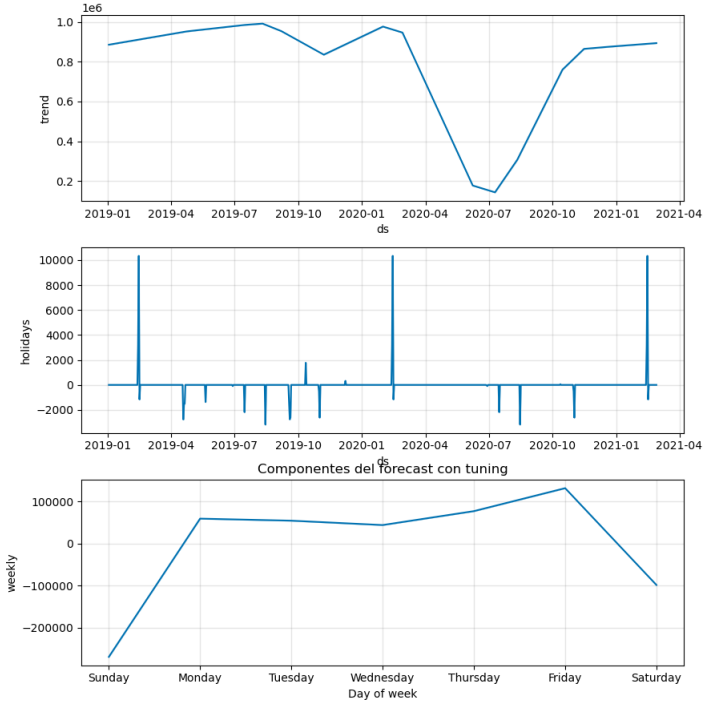
[4]En estadística, el error absoluto medio es una medida de la diferencia entre dos variables continuas, en este caso entre el valor del pronóstico y el valor observado.

Figura 4: Pronóstico con Changepoints - Prophet Hiper (Elaboración propia, 2020)



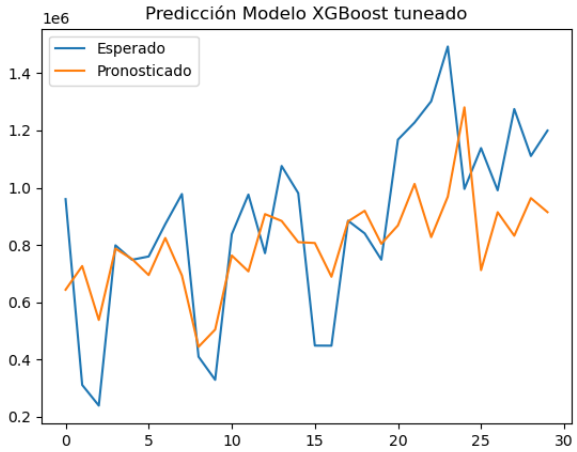
Por otro lado Prophet Hiper también nos entrega la siguiente Figura 5 en donde se puede apreciar los componentes principales del pronóstico entre ellos la tendencia, los días festivos y la estacionalidad de los datos.

Figura 5: Componentes principales- Prophet Hiper (Elaboración propia, 2020)



Por último la siguiente Figura 6 se puede observar las líneas de tendencia tanto para lo esperado como para lo pronosticado con el modelo XGBoost Hiper.

Figura 6: Pronóstico- XGBoost Hiper (Elaboración propia, 2020)



5. CONCLUSIONES

Del estudio realizado se puede concluir que el mejor modelo que pronostica los siguientes siete días y comparado a través del MAE, es Prophet Hiper con un resultado de 161688 pesos, seguido del modelo Prophet Default el cual obtiene un MAE de 191997 pesos. Dentro de los modelos XGBoost el mejor modelo fue el XGBoost Hiper con un MAE de 213611 pesos.

En otras palabras el modelo Prophet tiene un 24 por ciento menos de MAE que el mejor modelo de XGBoost.

Además se puede concluir que la búsqueda de hiperparámetros permite mejorar los modelos en un 15 por ciento para Prophet y en un 7 por ciento para XGboost.

Para ahondar más en este informe pueden revisar el código en el siguiente punto o bien visitar el GitHub[5] del alumno.

6. CÓDIGO

```
################################################################################
# Importamos las librerias necesarias.
################################################################################
import pandas as pd
from fbprophet import Prophet
from fbprophet.plot import add_changepoints_to_plot
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from numpy import asarray
from pandas import read_csv
from pandas import DataFrame
from pandas import concat
from sklearn.metrics import mean_absolute_error
from xgboost import XGBRegressor
from matplotlib import pyplot
from fbprophet.diagnostics import performance_metrics, cross_validation
from fbprophet.plot import plot_cross_validation_metric
import itertools
import numpy as np
```

[5] https://github.com/Rfariaspoblete/tarea$_M LA$

```
import seaborn as sns
import random
from sklearn.model_selection import RandomizedSearchCV, GridSearchCV
from sklearn.model_selection import StratifiedKFold

random.seed(1234)

#############################################################################
# IO
#############################################################################


file_path = '/home/rafaelfarias/Dropbox/Postgrados/MDS/MLA/Tarea MLA/data/Tarea MLA - Hoja 2.csv'
df = pd.read_csv(file_path)

# Le doy formato a la fecha para poder trabajarla.
df['ds'] = pd.to_datetime(df['ds'], format='%d/%m/%Y')
df = df.sort_values(by=['ds'], ascending=True)
df.reset_index(drop=True, inplace=True)

# Guardo el df ordenado para usarlo en el pron stico del modelo mejor.
df.to_csv(
    r'/home/rafaelfarias/Dropbox/Postgrados/MDS/MLA/Tarea MLA/data/datos2.csv',
    index=False)

path2 = '/home/rafaelfarias/Dropbox/Postgrados/MDS/MLA/Tarea MLA/data/datos2.csv'

#############################################################################
# Estadistica descriptiva.
#############################################################################

# Graficamos los datos.
graf = sns.histplot(data=df, kde='true')
plt.show()

outliers = sns.boxplot(data=df['y'])
plt.show()

# Describo los datos.
describe = df.describe()

# Le indicamos al modelos los d as "importantes"
dias_especiales = pd.DataFrame({
    'holiday': 'dias_especiales',
    'ds': pd.to_datetime(['2018-02-14', '2019-02-14', '2020-02-14',
                          '2021-02-14', '2022-02-14'], format='%Y-%m-%d'),
    'lower_window': -1,
    'upper_window': 1,
})
holidays = dias_especiales

#############################################################################
# Modelo Prophet sin tuning.
#############################################################################

# Creamos el modelo Prophet y le hacemos un fit.
m = Prophet(holidays=holidays, weekly_seasonality=True,
            daily_seasonality=False,
            yearly_seasonality=False, n_changepoints=20)
m.add_country_holidays(country_name='Chile')
m.fit(df)

# Se indica cu les ser n los futures.
future = m.make_future_dataframe(periods=7)
```

```
future.tail()

# Forecast
forecast = m.predict(future)
forecast[['ds', 'yhat', 'yhat_lower', 'yhat_upper']].tail(14)

# Se grafican los componentes del forecast (trend, weekly, yearly)
fig2 = m.plot_components(forecast)
plt.title('Componentes del forecast sin tuning')
plt.show()

# Se grafica cuando se producen los mayores cambios en la tendencia.
fig3 = m.plot(forecast)
a = add_changepoints_to_plot(fig3.gca(), m, forecast)
plt.title('Pron stico con changepoints modelo sin tunear')
plt.show()

# Crossvalidation.
df_cv = cross_validation(m, initial='30 days', horizon='7 days',
                         parallel='processes', period='1 days')

df_p = performance_metrics(df_cv, rolling_window=1)

fig4 = plot_cross_validation_metric(df_cv, metric='mae')
plt.title('MAE del modelo sin tuning')
plt.show()

print('MAE de Prophet sin tuning: %.3f' % df_p['mae'])

##############################################################################
# Hiperparametros
##############################################################################

# param_grid = {
#     'changepoint_prior_scale': [0.4, 0.5],
#
#     'changepoint_range': [0.8, 0.9],
#
#     'seasonality_prior_scale': [0.001, 0.01, 0.1, 0.5, 0.7, 1, 5, 10],
#
#     'holidays_prior_scale': [0.001, 0.01, 0.1, 0.5, 0.7, 1, 5, 10]
# }
#
# # Se genera toda la combinaci n de parametros.
# all_params = [dict(zip(param_grid.keys(), v))
#               for v in itertools.product(*param_grid.values())]
# maes = []  # Store the maes for each params here
#
# # Se usa cross validation para evaluar los parametros.
# for params in all_params:
#     m = Prophet(**params, yearly_seasonality=False, daily_seasonality=False,
#                 weekly_seasonality=True,
#                 holidays=holidays, n_changepoints=20)
#     m.add_country_holidays(country_name='Chile')
#     m.fit(df)  # Fit model with given params
#     df_cv = cross_validation(m, initial='30 days', horizon='7 days',
#                              parallel='processes', period='1 days')
#     df_p = performance_metrics(df_cv, rolling_window=1)
#     maes.append(df_p['mae'].values[0])
#
# # Encontrando los mejores parametros.
# tuning_results = pd.DataFrame(all_params)
# tuning_results['mae'] = maes
#
# # Se imprime el mejor par metro.
```

```
# best_params = all_params[np.argmin(maes)]
# best_params
# Darle index para lograr automatizar el paso de m s abajo

##############################################################################
# Modelo Prophet con hiperparametros.
##############################################################################

m2 = Prophet(changepoint_prior_scale=0.5, changepoint_range=0.9,
             seasonality_prior_scale=0.1, holidays_prior_scale=0.01,
             weekly_seasonality=True, holidays=holidays,
             yearly_seasonality=False, daily_seasonality=False,
             n_changepoints=22, interval_width=0.8)
m2.add_country_holidays(country_name='Chile')
m2.fit(df)

# Se indica cuales seran los futures y el periodo hacia adelante.
future2 = m2.make_future_dataframe(periods=7)
future2.tail()

# Forecast
forecast2 = m2.predict(future2)
forecast2[['ds', 'yhat', 'yhat_lower', 'yhat_upper']].tail(14)

forecast2[['yhat_upper']].sum()

# Se grafican los componentes del forecast (trend, weekly, yearly)
fig6 = m2.plot_components(forecast2)
plt.title('Componentes del forecast con tuning')
plt.show()

# Se grafica cuando se producen los mayores cambios en la tendencia.
fig7 = m2.plot(forecast2)
a2 = add_changepoints_to_plot(fig7.gca(), m2, forecast2)
plt.title('Pron stico con changepoints incluidos modelo tuneado')
plt.show()

# Crossvalidation
df_cv2 = cross_validation(m2, initial='30 days', horizon='7 days',
                          parallel='processes', period='1 days')

df_p2 = performance_metrics(df_cv2, rolling_window=1)

fig8 = plot_cross_validation_metric(df_cv2, metric='mae')
plt.title('MAE del modelo tuneado')
plt.show()

print('MAE de Prophet tuneado: %.3f' % df_p2['mae'])


##############################################################################
# XGBoost 1 sin tuning.
##############################################################################
# Series a aprendizaje supervisado.
def series_to_supervised(data, n_in=1, n_out=1, dropnan=True):
    n_vars = 1 if type(data) is list else data.shape[1]
    df = DataFrame(data)
    cols = list()
    # input sequence (t-n, ... t-1)
    for i in range(n_in, 0, -1):
        cols.append(df.shift(i))
    # forecast sequence (t, t+1, ... t+n)
    for i in range(0, n_out):
        cols.append(df.shift(-i))
    # put it all together
```

```
    agg = concat(cols, axis=1)
    # drop rows with NaN values
    if dropnan:
        agg.dropna(inplace=True)
    return agg.values


# train/test sets
def train_test_split(data, n_test):
    return data[:-n_test, :], data[-n_test:, :]


# Fit y un paso de prediccion.
def xgboost_forecast(train, testX):
    # transformando listas en arreglos.
    train = asarray(train)
    # split into input and output columns
    trainX, trainy = train[:, :-1], train[:, -1]
    # fit al modelo.
    model = XGBRegressor(objective='reg:squarederror', max_depth=2)
    model.fit(trainX, trainy)
    # un paso de predicci n.
    yhat = model.predict(asarray([testX]))
    return yhat[0]


# Walk-forward validacinn para la data sin variaciones.
def walk_forward_validation(data, n_test):
    predictions = list()
    # separando el dataset
    train, test = train_test_split(data, n_test)
    # Creando history con los datos de entrenamiento.
    history = [x for x in train]
    # Pasos de cada time-step el set de testeo.
    for i in range(len(test)):
        # separar el testeo entre input y output
        testX, testy = test[i, :-1], test[i, -1]
        # Fit al modelo en History y haciendo una predicci n.
        yhat = xgboost_forecast(history, testX)
        # Guardando el forecast en una lista de predicciones.
        predictions.append(yhat)
        # agregando la obervacion actual de history al siguiente loop.
        history.append(test[i])
        # Resumiendo el progreso.
        print('>expected=%.1f, predicted=%.1f' % (testy, yhat))
    # Estimando el error de la predicci n.
    error = mean_absolute_error(test[:, -1], predictions)
    return error, test[:, -1], predictions


# Pronosticando proximo dia tomando en cuenta los 30 dias anteriores.

# Cargando el dataset.
series = read_csv(path2, header=0, index_col=0)
values = series.values

# transform the time series data into supervised learning
data = series_to_supervised(values, n_in=6)

# Evaluando.
mae, y, yhat = walk_forward_validation(data, 30)

# Graficando esperado vs pronosticado.
pyplot.plot(y, label='Esperado')
pyplot.plot(yhat, label='Pronosticado')
```

```
pyplot.title('Predicci n Modelo XGBoost sin tuning')
pyplot.legend()
pyplot.show()

# Pronosticando proximo dia tomando en cuenta los 30 d as anteriores.
# Cargando el dataset
series = read_csv(path2, header=0, index_col=0)
values = series.values

# Transformando las series de tiempo en aprendizaje supervisado.
train = series_to_supervised(values, n_in=30)

# Separando entre input y outputs.
trainX, trainy = train[:, :-1], train[:, -1]

# Fit al modelo.
model = XGBRegressor(objective='reg:squarederror', gpu_id=-1, max_depth=2)

model.fit(trainX, trainy)

# Input para la predicci n.
row = values[-30:].flatten()

# Un paso de predicci n.
yhat = model.predict(asarray([row]))
print('Input: %s, Predicted with XGBoost sin tuning: %.3f' % (row, yhat[0]))


##############################################################################
# XGBoost 2 con tuning.
##############################################################################
# # Buscamos los mejores hiperparametros dentro del listado params.
# xgb = XGBRegressor(objective='reg:squarederror', gpu_id=-1)
#
# params = {
#     'min_child_weight': [5, 10, 40, 60],
#     'gamma': [0.5, 1.5, 3, 6],
#     'subsample': [0.6, 0.8, 1],
#     'colsample_bytree': [0.6, 0.8, 1],
#     'max_depth': [10, 20, 30],
#     'learning_rate': [0.3, 0.5, 0.8, 1],
#     'objective': ['reg:squarederror'],
# }
#
# param_comb = 100
# folds = 2
# skf = StratifiedKFold(n_splits=folds, shuffle=True)
# random_search = RandomizedSearchCV(xgb, param_distributions=params,
#                                    n_iter=param_comb,
#                                    scoring='neg_mean_absolute_error',
#                                    n_jobs=4,
#                                    cv=skf.split(trainX, trainy),
#                                    verbose=3)
#
# random_search.fit(trainX, trainy)
#
# print(random_search.cv_results_)
# print('\n Best estimator:')
# print(random_search.best_estimator_)
# print(
#     '\n Best normalized gini score for %d-fold search with %d parameter
    combinations:' % (
#     folds, param_comb))
# print(random_search.best_score_ * 2 - 1)
# print('\n Best hyperparameters:')
```

```
# print(random_search.best_params_)
# results = pd.DataFrame(random_search.cv_results_)


# Se crea el nuevo modelo con los hiperparametros encontrados.
def xgboost_forecast(train, testX):
    # transformando lista en arreglo.
    train = asarray(train)
    # split into input and output columns
    trainX, trainy = train[:, :-1], train[:, -1]
    # fit model
    model = XGBRegressor(base_score=0.5, booster='gbtree', colsample_bylevel=1,
                         colsample_bynode=1, colsample_bytree=1, gamma=0.5,
                         gpu_id=-1,
                         importance_type='gain', interaction_constraints='',
                         learning_rate=0.3, max_delta_step=0, max_depth=10,
                         min_child_weight=40,
                         monotone_constraints='()',
                         n_estimators=100, n_jobs=16, num_parallel_tree=1,
                         random_state=0,
                         reg_alpha=0, reg_lambda=1, scale_pos_weight=1,
                         subsample=1,
                         tree_method='exact', validate_parameters=1,
                         verbosity=None, objective='reg:squarederror',
                         eval_metric='mae')
    model.fit(trainX, trainy)
    # make a one-step prediction
    yhat = model.predict(asarray([testX]))
    return yhat[0]


# Prediccion.

model2 = XGBRegressor(base_score=0.5, booster='gbtree', colsample_bylevel=1,
                      colsample_bynode=1, colsample_bytree=1, gamma=0.5,
                      gpu_id=-1,
                      importance_type='gain', interaction_constraints='',
                      learning_rate=0.3, max_delta_step=0, max_depth=10,
                      min_child_weight=40,
                      monotone_constraints='()',
                      n_estimators=100, n_jobs=16, num_parallel_tree=1,
                      random_state=0,
                      reg_alpha=0, reg_lambda=1, scale_pos_weight=1,
                      subsample=1,
                      tree_method='exact', validate_parameters=1,
                      verbosity=None, objective='reg:squarederror',
                      eval_metric='mae')

model2.fit(trainX, trainy)
series2 = read_csv(path2, header=0, index_col=0)
values2 = series2.values
data = series_to_supervised(values2, n_in=6)
mae2, y2, yhat2 = walk_forward_validation(data, 30)

# plot expected vs predicted
pyplot.plot(y2, label='Esperado')
pyplot.plot(yhat2, label='Pronosticado')
pyplot.title('Predicci n Modelo XGBoost tuneado')
pyplot.legend()
pyplot.show()

# Pronosticando proximo dia tomando en cuenta los 30 dias anteriores y el mejor
# modelo encontrado.

# Input para la nueva predicci n.
```

```
row = values2[-30:].flatten()

# Un paso de prediccion.
yhat = model2.predict(asarray([row]))
print('Input: %s, Predicted with XGBoost con tuning: %.3f' % (row, yhat[0]))

# Resultados finales.
print('MAE de Prophet Default: %.3f' % df_p['mae'])
print('MAE de Prophet Hiper: %.3f' % df_p2['mae'])
print('MAE de XGBoost Default: %.3f' % mae)
print('MAE de XGBoost Hiper: %.3f' % mae2)
```

Listing 1: Código Tarea MLA

Referencias

[1] Andrew Ng. AI Tranformation Playbook (Working paper, Landing AI, 2018), https://landing.ai/ai-transformation-playbook/

[2] J. Friedman. Greedy function approximation: a gradient boosting machine. Annals of Statistics, 29(5):1189–1232, 2001.

[3] Harvey, A. Peters, S. (1990), 'Estimation procedures for structural time series models', Journal of Forecasting 9, 89–108.

[4] De la Hoz Domínguez , E. J., Fontalvo Herrera , T. J., y Mendoza Mendoza, A. A. (2020). Aprendizaje automático y PYMES: Oportunidades para el mejoramiento del proceso de toma de decisiones. Investigación E Innovación En Ingenierías, 8(1), 21-36. https://doi.org/10.17081/invinno.8.1.3506

[5] Michael E. Porter, Victor E. Millar, Ser Competitivo, novena edición, Cap 3., (2009) Harvard Business Press,

[6] Taylor SJ, Letham B. 2017. Forecasting at scale. PeerJ Preprints 5:e3190v2. https://doi.org/10.7287/peerj.preprints.3190v2

[7] P. Li. Robust Logitboost and adaptive base class (ABC) Logitboost. In Proceedings of the Twenty-Sixth Conference Annual Conference on Uncertainty in Artificial Intelligence (UAI'10), pages 302–311, 2010.

[8] KDD '16: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data MiningAugust 2016 Pages 785–794 https://doi.org/10.1145/2939672.2939785