Instituto Tecnológico de Estudios Superiores de Occidente

Construcción de modelos predictivos del precio del maíz utilizando dos métodos convexos: SVR y regresión lineal múltiple

Departamento de Matemáticas y Física

Optimización convexa

Dr. Juan Diego Sánchez Torres

INTEGRANTES
Emilio Carranza Ávila
Gustavo Ibáñez Sosa
Gabriela Lozano Orozco
Karen Manguart Páez

11 de mayo del 2022

Tabla de contenido

1. Introducción

El maíz es uno de los cultivos de mayor importancia en México, y una parte importante de la economía del país, siendo uno de los productos esenciales de la dieta mexicana. Además, México es uno de los países con mayor cantidad de importación y producción de este grano.

Históricamente los productores mexicanos han tenido que sortear con la volatilidad de los precios, en gran parte causados por la volatilidad de la producción del maíz en Estados Unidos y México, la cual es dependiente de factores exógenos que tienen una repercusión importante en dichos precios.

Si a esto sumamos el incremento de fenómenos climáticos tenemos una receta perfecta para la volatilidad de los precios de este grano, con un potencial a que se incremente a futuro. Por esto, resulta de gran importancia el encontrar métodos para poder predecir su precio.

Además, existe una gran dificultad para su predicción ya que México carece de una señal para el comportamiento de los precios. Esto nos presenta una gran oportunidad para el estudio de los precios y el encontrar un modelo adecuado para la predicción de éste.

Bajo la premisa de que el precio del maíz de Estados Unidos influye de manera directa en el precio del maíz mexicano partiremos de una base de datos compuesta de datos de los precios históricos del maíz en manera de serie de tiempo partiendo de 1959 hasta 2022. Utilizando esta base se pretende el hacer una predicción de los precios a futuro.

Para lograrlo, se decidió aplicar dos modelos distintos de predicción, utilizando dos métodos de optimización convexa, el de *Support vector machine* y el de regresión lineal múltiple. Siendo que la base de datos presenta pocas variables siendo compuesta por las variables *date* y *price*, con 15,800 observaciones a lo largo de 63 años.

Debido a que contamos con pocas variables, pero sí con una serie de tiempo rica en observaciones se tomó la decisión de utilizar algoritmos genéticos para aumentar las posibilidades de encontrar variables que permitan explicar el cambio de precios. Para esto se implementó un transformador simbólico el cual busca encontrar un conjunto de variables con correlación más cercana a cero.

2. Objetivos

El presente trabajo de investigación tiene por objetivo construir dos modelos predictivos de una serie de tiempo del precio del maíz utilizando dos métodos de optimización convexa: SVR y regresión lineal múltiple. Así mismo, se busca calcular métricas de error para poder determinar cuál de los dos métodos predice con mejor exactitud con respecto al tiempo.

3. Marco teórico

3.1 Series de Tiempo

Una serie de tiempo es una secuencia de observaciones, con datos ordenados cronológicamente y espaciados entre sí de forma uniforme; así, los datos con frecuencia son dependientes entre sí. Su principal objetivo es su análisis para realizar pronósticos.

Las predicciones de series de tiempo plantean un problema crucial debido a sus aplicaciones en ingeniería, economía y finanzas. Para solucionar dicha problemática, se han desarrollado técnicas que incluyen modelos estadísticos tradicionales tales como ARIMA (*Autoregressive Integrated Moving Average*), funciones de transferencia, modelos para predicciones de series no lineales como redes neuronales artificiales y *SVR*.

3.2 SVR (*Support Vector Regression*)

El método de regresión de vectores de soporte (SVR) se basa en el principio de minimización de riesgos estructurados. Busca minimizar un límite superior e inferior del error generalizado en vez de encontrar errores empíricos.

Considerando el siguiente conjunto de entrenamiento:

$$T = \{(x_1, y_1), \cdots, (x_l, y_l)\} \in (R^n \times Y)^l,$$
$$\text{donde } x_i \in R^n,\ y_i \in Y = R,\ i = 1, \cdots, l.$$

para el caso lineal de *SVR* se formula el problema como un QPP (Problema de Programación Cuadrática Convexa):

$$\begin{aligned} \min_{w,b,\xi^{(*)}} \quad & \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{l}(\xi_i + \xi_i^*), \\ \text{s.t.} \quad & (w \cdot x_i) + b - y_i \leqslant \varepsilon + \xi_i, i = 1, \cdots, l, \\ & y_i - (w \cdot x_i) - b \leqslant \varepsilon + \xi_i^*, i = 1, \cdots, l, \\ & \xi_i^{(*)} \geq 0, i = 1, \cdots, l, \end{aligned}$$

donde $\xi^{(*)}$ es una variable de holgura y $C > 0$ es un parámetro de penalización.

3.3 Regresión lineal

El método de regresión lineal es un modelo que asume una relación lineal entre las variables de entrada independientes (x) y una variable de salida dependiente (y). Por lo tanto, y puede calcularse a través de una combinación lineal de las variables de entrada. Se pueden clasifican de acuerdo con la cantidad de variables de entrada como regresión lineal simple, cuando existe una variable de entrada; y como regresión lineal múltiple cuando existen más de una variable de entrada.

3.3.1 Regresión lineal múltiple

La regresión lineal múltiple se considera una extensión de la regresión de mínimos cuadrados ordinarios (OLS) con el sustento de involucrar más de una variable explicativa; es utilizada considerablemente en inferencia financiera. La fórmula que describe el modelo es

$$yi=\beta 0+\beta 1xi1+\beta 2xi2+...+\beta pxip+\epsilon$$

donde,
$i = n$ observaciones
y_i = variable dependiente
x_i = variables explicativas
β_0 = intercepto
β_p = coeficientes de pendiente para p variables explicativas
ϵ = termino de error del modelo.

En este caso el problema de optimización tiene por función objetivo:

$$\min || Ax - b||^2$$

donde, A es una matriz de variables explicativas, b de variables de salida y x un vector de pesos.

3.4 Transformador simbólico

Symbolic Transformer es un transformador supervisado que genera distintos ajustes por medio de un algoritmo genético. Se crea una población original aleatoria con la cual se iniciará un proceso de "evolución" para encontrar la mejor población que estará conformada por los individuos con el mejor ajuste y los coeficientes de correlación más cercanos a cero.

El proceso consiste en evaluar cada individuo con una serie de funciones matemáticas las cuales devolverán el nivel de ajuste de cada uno. Estos individuos pasaran a la siguiente generación y así sucesivamente hasta completar el proceso. Una de las grandes ventajas que tiene *Symbolic Transformation* es que nos permite controlar el nivel de complejidad con el que buscará el mejor resultado. Este control se lleva a cabo por medio de sus parámetros definiendo el número de generaciones, el porcentaje de mutaciones, el número de individuos que sobreviven al cambio generacional, entre otros.

Uno de los parámetros que se deben de tener en cuenta es el de *metric*, este parámetro se vuelve relevante en el resultado final ya que con él se define qué coeficiente de correlación se utilizará para corroborar los resultados. Específicamente en la librería gplearn y en la función de *Symbolic Transformer* se cuenta con dos tipos de coeficiente de correlación: Pearson y Spearman.

En el caso de la correlación de Pearson, ésta evalúa la relación lineal entre dos variables continuas, en una relación lineal, las variables se mueven en la misma dirección a un ritmo constante. En el caso de la correlación de Spearman, se evalúa la relación monótona entre dos variables continuas u ordinales, en una relación monótona las variables tienden a moverse en la misma dirección relativa, pero no necesariamente a un ritmo constante.

3.5 MAPE – Error porcentual absoluto medio

El MAPE es un indicador del desempeño de un pronóstico. Mide el tamaño del error absoluto en términos porcentuales y eso lo hace fácil de interpretar. La fórmula para calcularlo es la siguiente:

$$\text{MAPE} = \frac{100\%}{n} \sum_{t=1}^{n} \left| \frac{A_t - F_t}{A_t} \right|$$

Donde A_t es el valor real y F_t es el valor de pronóstico.

4. Implementación y desarrollo

4.1 División de base de datos

Se inició separando la base de datos en dos conjuntos: entrenamiento y prueba (Fig. 1). Así mismo, se definió como única variable explicativa el tiempo y como variable de respuesta el precio del maíz. Debido a que la base de datos tiene un orden cronológico, los datos son dependientes entre sí y la separación de la base de datos se realizó de forma lineal y no aleatoria en una proporción 70-30.

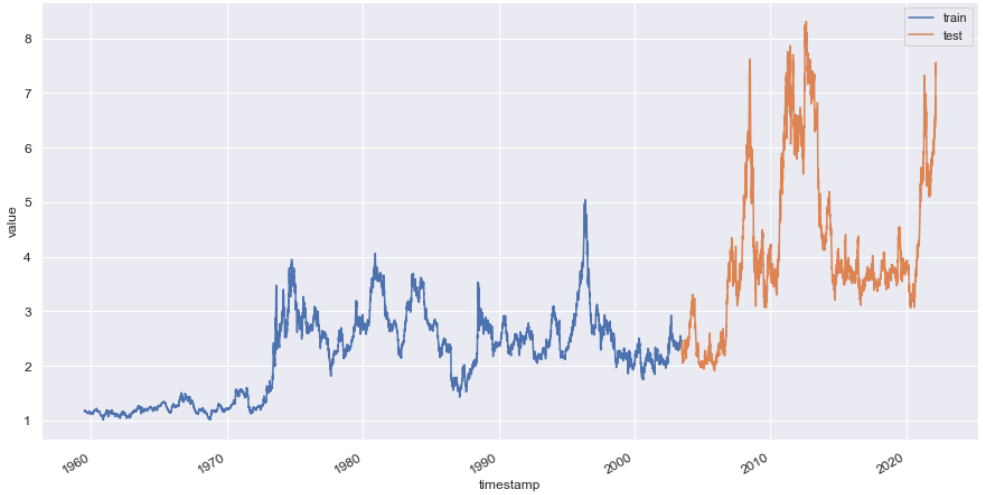

Fig. 1. División lineal (no aleatoria) de la base de datos en conjuntos de entrenamiento y prueba.

Posteriormente, se realizó un preprocesamiento de datos utilizando la normalización *MinMax.* Este método se utilizó para reducir el rango dado de los datos a un rango entre 0 y 1 sin alterar la forma de la distribución original.

4.2 SVR

Se realizó un modelo SVR con un kernel de tipo RBF (*Radial Basis Function*) con un valor de gamma de 0.5, parámetro de penalización (C) de 10, y holgura de 0.05. Los resultados de predicción en el conjunto de prueba se muestran en la figura 2.

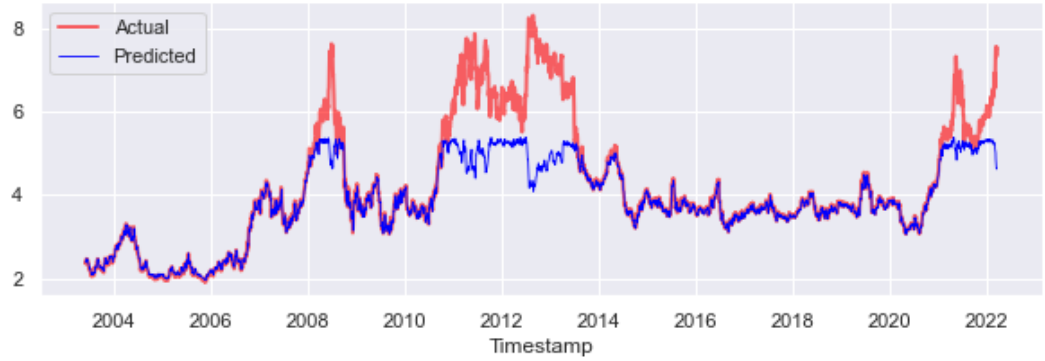

Fig. 2. Gráfico del conjunto de prueba de los valores reales y los valores predichos con gamma 0.01

El MAPE obtenido para el anterior ajuste fue de 5.84%.

A continuación, se muestra la tabla de algunas corridas generadas para este ajuste SVR y sus respectivos resultados.

Tabla 1. Resultados de MAPE de 5 modelos *SVR* de kernel *RBF* variando gamma.

kernel	gamma	C	ε	MAPE (%)
RBF	0.5	10	0.05	5.841
RBF	0.25	10	0.05	4.016
RBF	0.10	10	0.05	3.348
RBF	0.01	10	0.05	**1.973**
RBF	0.0005	10	0	3.146

Como se muestra en la tabla anterior, se encontró que el mejor valor de MAPE fue de 1.97% y se logró utilizando un valor de gamma de 0.01. La siguiente gráfica muestra las predicciones obtenidas comparadas con los valores reales. Es evidente que el ajuste es significativamente mejor con este nuevo valor de gamma encontrado.

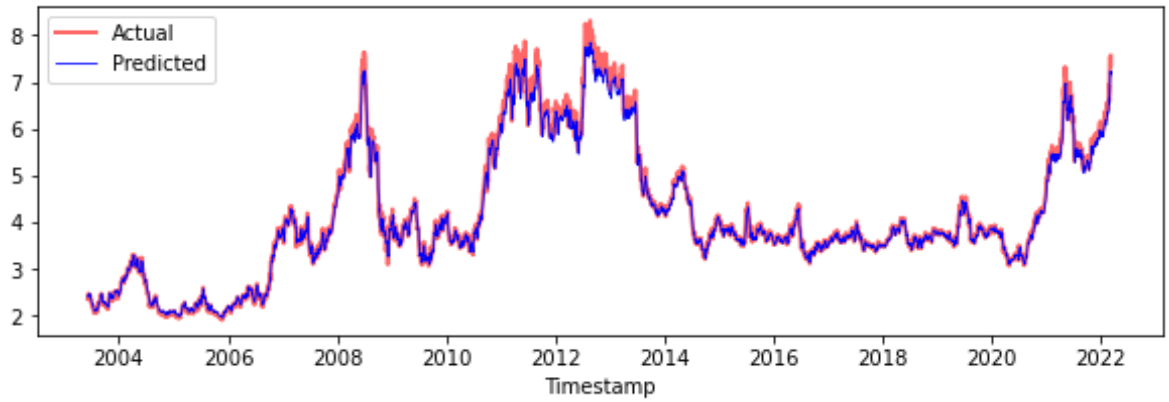

Fig. 3. Gráfico del conjunto de prueba de los valores reales y los valores predichos con gamma 0.01.

4.3 Regresión lineal múltiple

Posteriormente, con el objetivo de comparar el ajuste conseguido con el modelo anterior, se realizó un modelado de naturaleza convexa, a través de una regresión lineal múltiple. La base de datos considerada fue por supuesto la misma, igualmente transformada.

Al tratarse de una serie de tiempo, con una sola variable explicativa, lo primero que se hizo fue correr un transformador simbólico que permitiera encontrar distintas nuevas variables que hicieran posible ajustar los datos a una regresión lineal múltiple.

Una vez con las nuevas variables calculadas, se dividieron los datos en conjuntos de entrenamiento y prueba y se corrió el algoritmo de ajuste de regresión lineal.

Lo anterior se realizó varias veces, con el objetivo de optimizar los parámetros de transformador simbólico. Finalmente, el mejor MAPE obtenido fue 1.59% y se logró con los siguientes parámetros:

- generations=20
- population_size=2000,
- hall_of_fame=100,
- n_components=10,
- function_set=function_set,
- parsimony_coefficient=0.0005,
- max_samples=0.9, verbose=1,
- random_state=0, n_jobs=3,
- metric='spearman'

5. Conclusiones

Siendo el maíz uno de los principales granos producidos y consumidos dentro del país, el poder predecir su precio resulta muy relevante. Se trata de un insumo con un grado alto de volatilidad, y siendo México el segundo importador de maíz del mundo, conocer sus precios es gran importancia para la economía.

Como se puede observar, los resultados de ambos modelos fueron similares, obteniendo el mejor MAPE de los modelos de 1.97% para la regresión vectorial y 1.59% con la regresión lineal múltiple, esto teniendo en cuenta que para resolver el problema de no linealidad se tuvo que utilizar un transformador simbólico para poder crear variables que pudieran integrarse al modelo.

Por otro lado, el SVR pudo resolver el problema de no linealidad desde el inicio, así como obtener resultados con un bajo porcentaje de MAPE de una manera más rápida y sencilla. Esto debido a la naturaleza del SVR, que resulta más flexible comparado a la regresión lineal, además de que al no utilizar el transformador simbólico el tiempo de procesamiento fue mucho menor, requiriendo menos recursos.

6. Referencias

Villavicencio, J. (09 de 05 de 2022). *Introducción a Series de Tiempo.* Obtenido de Instituto de Estadísticas de Puerto Rico: http://www.estadisticas.gobierno.pr/iepr/LinkClick.aspx?fileticket=4_BxecUaZmg%3D

Velázques, J., Olaya, Y., & Franco, C. (2010). Predicción de series temporales usando máquinas de vestores de soporte. *Ingeniare*, 64-75.

Ping-Feng, P., Kuo-Ping, L., & Chi-Shen, L. (2009). Pronóstico de series de tiempo mediante un modelo de regresión de vector de soporte estacional. *Elsevier Ltd.* , 4261–4265.

Xuchan, J., Manjin, C., Yuhong, X., Fuqiang, Q., & Yingjie, T. (2014). Regresión vectorial y análisis de series temporales para el Pronóstico del requerimiento total de agua de Bayannur. *Elsevier*, 523 – 531 .

Brownlee, J. (09 de 05 de 2022). *Machine Learning Mastery.* Obtenido de Linear Regression for Machine Learning: https://machinelearningmastery.com/linear-regression-for-machine-learning/#:~:text=Linear%20regression%20is%20a%20linear,the%20input%20variables%20(x).

Hayes, A. (09 de 05 de 2022). *Investiopedia.* Obtenido de Regresión Lineal Múltiple: https://www.investopedia.com/terms/m/mlr.asp#:~:text=Key%20Takeaways-,Multiple%20linear%20regression%20(MLR)%2C%20also%20known%20simply%20as%20multiple,uses%20just%20one%20explanatory%20variable.

Minitab. (2022). *Relaciones lineales, no lineales y monótonas*. Obtenido de Support Minitab: https://support.minitab.com/es-mx/minitab/19/help-and-how-to/statistics/basic-statistics/supporting-topics/basics/linear-nonlinear-and-monotonic-relationships/#:~:text=En%20una%20relaci%C3%B3n%20mon%C3%B3tona%2C%20las,direcci%C3%B3n%20a%20un%20ritmo%20con

Minitab. (2022). *Una comparación de los métodos de correlación de Pearson y Spearman-Minitab*. Obtenido de Support Minitab: https://support.minitab.com/es-mx/minitab/18/help-and-how-to/statistics/basic-statistics/supporting-topics/correlation-and-covariance/a-comparison-of-the-pearson-and-spearman-correlation-methods/#:~:text=Los%20coeficientes%20de%20correlaci%C3%B3n%20de%20P

gplearn. (2022). *Symbolic Transformer*. Obtenido de gplearn API reference: https://gplearn.readthedocs.io/en/stable/reference.html#gplearn.genetic.SymbolicTransformer

7. Anexos

Anexo 1. Código SVR

```
# %% Import libraries
import os
import warnings
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import datetime as dt
import math
from sklearn.svm import SVR
from sklearn.preprocessing import MinMaxScaler


# %% Load data_frame

df=pd.read_csv("Desktop/MCD/Optimizacion/Proyecto/corn-prices")
df=df.dropna()
df['date'] = pd.to_datetime(df['date'], format='%d/%m/%Y')
df2=df.set_index(df['date'], drop=False, append=False, inplace=False,
verify_integrity=False).drop('date', 1)
    #Allocate train and test start periods
train_start_dt = '1959-07-01 00:00:00'
test_start_dt = '2003-05-30 00:00:00'
    #Visualize differences
df2[(df2.index < test_start_dt) & (df2.index >=
train_start_dt)][['value']].rename(columns={'value':'train'}) \
    .join(df2[test_start_dt:][['value']].rename(columns={'value':'test'}),
how='outer') \
    .plot(y=['train', 'test'], figsize=(15, 8), fontsize=12)
plt.xlabel('timestamp', fontsize=12)
plt.ylabel('value', fontsize=12)
plt.show()


#%% Prepare data fro training

train = df2.copy()[(df2.index >= train_start_dt) & (df2.index <
test_start_dt)][['value']]
test = df2.copy()[df2.index >= test_start_dt][['value']]
    #Scale the training data to be in the range (0, 1):
scaler = MinMaxScaler()
train['value'] = scaler.fit_transform(train)
test['value'] = scaler.transform(test)
    #Create data with time-steps (transform the input data to be of the form
[batch, timesteps])
    #Converting to numpy arrays
train_data = train.values
test_data = test.values
    #take timesteps = 5
timesteps=5
```

```
        #Converting training and test data to 2D tensor using nested list
comprehension:
train_data_timesteps=np.array([[j for j in train_data[i:i+timesteps]] for i
in range(0,len(train_data)-timesteps+1)])[:,:,0]
test_data_timesteps=np.array([[j for j in test_data[i:i+timesteps]] for i in
range(0,len(test_data)-timesteps+1)])[:,:,0]
    #Selecting inputs and outputs from training and testing data:
x_train, y_train = train_data_timesteps[:,:timesteps-
1],train_data_timesteps[:,[timesteps-1]]
x_test, y_test = test_data_timesteps[:,:timesteps-
1],test_data_timesteps[:,[timesteps-1]]


#%% Implement SVR

model = SVR(kernel='rbf',gamma=0.5, C=10, epsilon = 0.05)
    #Fit the model on training data
model.fit(x_train, y_train[:,0])
    #Make model prediction
y_train_pred = model.predict(x_train).reshape(-1,1)
y_test_pred = model.predict(x_test).reshape(-1,1)


#%% Evaluate model
    # Scaling the predictions
y_test_pred = scaler.inverse_transform(y_test_pred)
    # Scaling the original values
y_test = scaler.inverse_transform(y_test)
    #Check model performance on training and testing data
train_timestamps = df2[(df2.index < test_start_dt) & (df2.index >=
train_start_dt)].index[timesteps-1:]
test_timestamps = df2[test_start_dt:].index[timesteps-1:]
    #Plot the predictions for testing data
plt.figure(figsize=(10,3))
plt.plot(test_timestamps, y_test, color = 'red', linewidth=2.0, alpha = 0.6)
plt.plot(test_timestamps, y_test_pred, color = 'blue', linewidth=0.8)
plt.legend(['Actual','Predicted'])
plt.xlabel('Timestamp')
plt.show()

#%% Metrics

def mape(actual, pred):
    actual, pred = np.array(actual), np.array(pred)
    return np.mean(np.abs((actual - pred) / actual)) * 100

print('MAPE for testing data: ', mape(y_test, y_test_pred), '%')
```

Anexo 2. Código Symbolic Transformer

```
# %% Import libraries

import numpy as np
import pandas as pd
import datetime as dt
import math
from sklearn.preprocessing import MinMaxScaler
import numpy as np
import pandas as pd
from sklearn.linear_model import Ridge
from sklearn.utils import check_random_state
from sklearn.preprocessing import LabelEncoder, OneHotEncoder
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from gplearn.genetic import SymbolicTransformer, SymbolicRegressor

# %%Load data_frame

df=pd.read_csv("Desktop/MCD/Optimizacion/Proyecto/corn-prices")
df=df.dropna()
df['date'] = pd.to_datetime(df['date'], format='%d/%m/%Y')
df=df.set_index(df['date'], drop=False, append=False, inplace=False,
verify_integrity=False).drop('date', 1)


# %% Time Series Data Preparation

    #Scale data frame to be in the range (0, 1)
scaler = MinMaxScaler()
df['value'] = scaler.fit_transform(df)

    #Convert to numpy array
df_data= df.values
    #Create data with time-steps (transform the input data to be of the form
[batch, timesteps])
        #take timesteps = 5
timesteps=5
    #Converting training and test data to 2D tensor using nested list
comprehension:
df_data_timesteps=np.array([[j for j in df_data[i:i+timesteps]] for i in
range(0,len(df_data)-timesteps+1)])[:,:,0]
    #Split x and y
x, y= df_data_timesteps[:,:timesteps-1],df_data_timesteps[:,[timesteps-1]]
x_raw=x.copy()
```

```
# %% Symbolic Transformer

    #Scale x to be in the range (0, 1)
x= scaler.fit_transform(x)
x_raw= scaler.inverse_transform(x)
NUM = 15796 #Row number of df
est = Ridge()
est.fit(x[:NUM, :], y[:NUM])
function_set = ['add', 'sub', 'mul', 'div',
                                'sqrt', 'log', 'abs', 'neg', 'inv',
                                'max', 'min', 'sin', 'cos', 'tan']
gp = SymbolicTransformer(generations=20, population_size=2000,
                         hall_of_fame=100, n_components=10,
                         function_set=function_set,
                         parsimony_coefficient=0.0005,
                         max_samples=0.9, verbose=1,
                         random_state=0, n_jobs=3,
                         metric='spearman')
gp.fit(x[:NUM, :], y[:NUM])
gp_features = gp.transform(x)
x = np.hstack((x, gp_features))

# %% Split x_train, y_train, x_test, y_test 70:30

x_train= x[0:11057,:]
y_train=y[0:11057,:]
x_test= x[11057:15796,:]
y_test=y[11057:15796,:]

# %% Train Linear Regression Model

from sklearn.linear_model import LinearRegression
regressor = LinearRegression()
regressor.fit(x_train, y_train)


print(regressor.intercept_)
print(regressor.coef_)
print(f'r_sqr value: {regressor.score(x_train, y_train)}')

#%% Predictions
y_pred= regressor.predict(x_test)

#%% Metrics
def mape(actual, pred):
    actual, pred = np.array(actual), np.array(pred)
    return np.mean(np.abs((actual - pred) / actual)) * 100

from sklearn import metrics
print('Mean Absolute Error:', metrics.mean_absolute_error(y_test, y_pred))
print('Mean Squared Error:', metrics.mean_squared_error(y_test, y_pred))
print('Root Mean Squared Error:', np.sqrt(metrics.mean_squared_error(y_test,
y_pred)))
print('MAPE for testing data: ', mape(y_test, y_pred), '%')
```

Anexo 3. Repositorio de proyecto

https://github.com/GLOZANOO/Final_Project_Optimizaci-n.git