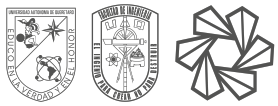


UNIVERSIDAD AUTÓNOMA DE QUERÉTARO
FACULTAD DE INGENIERÍA
DIVISIÓN DE INVESTIGACIÓN Y POSGRADO

Machine Learning: Práctica 4. Regresión y métricas de datos continuos.

Alumno:
Ing. Enrique Mena Camilo

Profesor:
Dr. Marco Antonio Aceves Fernández

Abril 2023

Índice

1. Objetivos

- Implementar regresión polinomial y otro tipo de regresión.
- Obtener métricas de datos continuos:
  - Mean Square Error (MSE)
  - Root Mean Square Error (RMSE)
  - Mean Absolute Error (MAE)
  - Relative Squared Error (RSE)
  - Pearson Correlation Coeficient (PCC)
  - R^2

2. Introducción

La regresión es un tipo de análisis estadístico que se utiliza para modelar la relación entre variables y predecir valores numéricos en función de variables independientes. Los algoritmos de regresión son técnicas computacionales que se utilizan para encontrar la mejor relación matemática entre los datos disponibles y la variable objetivo.

Algunos de los algoritmos de regresión más comunes son:

- **Regresión lineal**: Es uno de los algoritmos de regresión más simples y ampliamente utilizados. Modela la relación entre dos variables continuas mediante una línea recta que mejor se ajusta a los datos disponibles.
- **Regresión polinomial**: Este algoritmo permite modelar relaciones no lineales entre las variables al usar polinomios de grado mayor que uno para ajustar los datos. Puede capturar patrones más complejos en los datos, pero también puede sufrir de sobreajuste si se utiliza con polinomios de alto grado.
- **Regresión Ridge**: Es un algoritmo de regresión que se utiliza para abordar el problema de multicolinealidad en modelos de regresión lineal. La multicolinealidad ocurre cuando hay alta correlación entre las variables predictoras, lo que puede causar problemas en la estimación de los coeficientes de regresión en un modelo lineal.
- **Regresión de redes neuronales**: Este algoritmo utiliza redes neuronales artificiales para modelar la relación entre variables. Puede capturar patrones complejos en datos grandes y complejos, pero también puede requerir una cantidad significativa de datos para su entrenamiento.

Evaluar el desempeño de un modelo de regresión es una parte crítica del proceso de modelado, ya que permite comprender qué tan bien se ajusta el modelo a los datos y cómo se puede mejorar. La evaluación del desempeño de un modelo de regresión implica comparar las predicciones del modelo con los valores reales de la variable objetivo para determinar qué tan preciso es el modelo en sus predicciones. Esto ayuda a determinar la calidad del modelo y su capacidad para generalizar a nuevos datos.

La evaluación del desempeño de un modelo de regresión se puede realizar utilizando varias técnicas y métricas, tales como:

- Error cuadrático medio (MSE).
- Raíz del error cuadrático medio (RMSE).
- Error absoluto medio (MAE).
- Error cuadrático relativo (RSE).

- Coeficiente de correlación de Pearson (PCC).
- Coeficiente de determinación (R^2).

La evaluación del desempeño de un modelo de regresión es una etapa esencial del proceso de modelado, ya que permite comprender la calidad y la capacidad de generalización del modelo, identificar posibles mejoras y tomar decisiones basadas en los resultados del modelo.

3. Marco teórico

3.1. Algoritmos de regresión

3.1.1. Regresión lineal simple

La regresión lineal simple es un método estadístico utilizado para modelar la relación entre dos variables, donde una variable, llamada variable dependiente o de respuesta, se estima en función de otra variable, llamada variable independiente o predictora. En otras palabras, la regresión lineal simple busca encontrar una línea recta que mejor se ajuste a los datos y describa la relación lineal entre las dos variables.

El modelo de regresión lineal simple se puede expresar matemáticamente mediante la siguiente ecuación:

$$y = \beta_0 + \beta_1 * x$$

Donde y es la variable dependiente o de respuesta que se quiere predecir. x es la variable independiente o predictora que se utiliza para predecir la variable dependiente. β_0 es la intersección de la línea de regresión con el eje y, también conocida como el coeficiente de intersección o el término constante. β_1 es la pendiente de la línea de regresión, que representa el cambio en la variable dependiente por cada unidad de cambio en la variable independiente, también conocida como el coeficiente de regresión.

La regresión lineal simple se utiliza para varios propósitos, como la predicción de valores futuros de la variable dependiente, la identificación de la fuerza y dirección de la relación entre las dos variables, la inferencia estadística sobre los coeficientes del modelo, y la evaluación de la bondad de ajuste del modelo a los datos observados. Sin embargo, es importante tener en cuenta las suposiciones y limitaciones de la regresión lineal simple, la normalidad de los errores, y realizar una evaluación adecuada del modelo antes de interpretar los resultados o realizar predicciones basadas en el mismo.

3.1.2. Regresión polinomial

La regresión polinomial es una técnica de análisis de regresión que se utiliza para modelar relaciones no lineales entre variables. Mientras que la regresión lineal simple modela relaciones lineales, la regresión polinomial permite ajustar modelos que capturan relaciones curvilíneas o no lineales entre variables.

El modelo de regresión polinomial se puede expresar matemáticamente mediante la siguiente ecuación:

$$y = \beta_0 + \beta_1 * x + \beta_2 * x^2 + ... + \beta_n * x^n$$

Donde y es la variable dependiente o de respuesta que se quiere predecir. x es la variable independiente o predictora que se utiliza para predecir la variable dependiente. β_0 es la intersección del modelo con el eje y, también conocida como el coeficiente de intersección o el término constante. $\beta_1, \beta_2, ..., \beta_n$ son los coeficientes de regresión que representan las diferentes potencias de la variable independiente x, desde el primer término lineal hasta el término n-ésimo de mayor grado.

La regresión polinomial se utiliza cuando se sospecha que la relación entre las variables no es lineal y puede tener una forma curvilínea o no lineal en los datos. Puede ser útil en situaciones en las que la relación entre las variables es compleja y no se puede describir adecuadamente con un modelo lineal simple. Sin embargo, es importante tener en cuenta que la regresión polinomial puede ser más propensa al sobreajuste, especialmente con polinomios de alto grado, y se deben tomar precauciones para evaluar la calidad del ajuste y la interpretación de los resultados. Además, la elección del grado del polinomio es un aspecto crítico en la regresión polinomial y requiere un enfoque cuidadoso basado en el contexto del problema y la interpretación de los resultados obtenidos.

3.1.3. Regresión lineal múltiple

La regresión lineal múltiple es una técnica de análisis de regresión que se utiliza para modelar la relación entre una variable dependiente y dos o más variables independientes. A diferencia de la regresión lineal simple, que involucra solo una variable independiente, la regresión lineal múltiple permite analizar cómo varias variables independientes se relacionan conjuntamente con la variable dependiente.

El modelo de regresión lineal múltiple se puede expresar matemáticamente mediante la siguiente ecuación:

$$y = \beta_0 + \beta_1 * x_1 + \beta_2 * x_2 + ... + \beta_n * x_n$$

Donde y es la variable dependiente o de respuesta que se quiere predecir. x_1, x_2, ..., x_n son las variables independientes o predictoras que se utilizan para predecir la variable dependiente. β_0 es la intersección del modelo con el eje y, también conocida como el coeficiente de intersección o el término constante. β_1, β_2, ..., β_n son los coeficientes de regresión que representan las relaciones lineales entre las variables independientes y la variable dependiente.

La regresión lineal múltiple se utiliza cuando se sospecha que la relación entre la variable dependiente y las variables independientes es más compleja y no puede ser capturada adecuadamente por un modelo lineal simple. Puede ser útil en situaciones en las que se desea analizar el efecto conjunto de múltiples variables independientes en la variable dependiente, o cuando se busca realizar predicciones más precisas basadas en múltiples predictores. Sin embargo, al igual que con la regresión lineal simple, es importante realizar una evaluación cuidadosa del modelo, incluyendo la interpretación de los coeficientes de regresión, la evaluación de la calidad del ajuste y la validación del modelo, para asegurar la robustez y la validez

de los resultados obtenidos.

3.1.4. Regresión Ridge

La regresión Ridge es una técnica de regresión regularizada que se utiliza para mitigar el problema de multicolinealidad en modelos de regresión lineal múltiple, y así mejorar la estabilidad y el rendimiento de los modelos. La multicolinealidad se refiere a la presencia de alta correlación entre dos o más variables independientes en un modelo de regresión, lo que puede provocar inestabilidad en los coeficientes de regresión y reducir la capacidad de generalización del modelo.

La ecuación matemática de la regresión Ridge se puede expresar como:

$$y = \beta_0 + \beta_1 * x_1 + \beta_2 * x_2 + ... + \beta_n * x_n + \epsilon$$

Donde y es la variable dependiente o de respuesta que se quiere predecir. x_1, x_2, ..., x_n son las variables independientes o predictoras que se utilizan para predecir la variable dependiente. β_0 es la intersección del modelo con el eje y, también conocida como el coeficiente de intersección o el término constante. β_1, $beta_2$, ..., β_n son los coeficientes de regresión que representan las relaciones lineales entre las variables independientes y la variable dependiente. ϵ es el término de error, que representa la variabilidad no explicada por el modelo.

La regresión Ridge se utiliza cuando se sospecha que existe multicolinealidad en los datos y se busca mitigar su efecto en los coeficientes de regresión. Es especialmente útil en situaciones en las que el número de variables independientes es alto o cuando se trabaja con conjuntos de datos de alta dimensionalidad. Sin embargo, al igual que con cualquier técnica de modelado, es importante realizar una evaluación cuidadosa del modelo, incluyendo la interpretación de los coeficientes de regresión, la evaluación de la calidad del ajuste y la validación del modelo, para asegurar la robustez y la validez de los resultados obtenidos.

3.2. Métricas para evaluación de regresión

3.2.1. Error cuadrático medio

El error cuadrático medio (MSE, por sus siglas en inglés) es una medida de evaluación de modelos de regresión que mide la discrepancia promedio entre los valores predichos por el modelo y los valores reales de la variable dependiente. Es el promedio de los errores al cuadrado, lo que implica que los errores negativos y positivos se elevan al cuadrado antes de promediarlos.

La fórmula matemática del error cuadrático medio es:

$$MSE = \frac{\Sigma(y_i - \hat{y}_i)^2}{n}$$

Donde MSE es el error cuadrático medio. n es el número de observaciones en el conjunto de datos. y_i es el valor real de la variable dependiente en la i-ésima observación. $\hat{y}_i$ es el valor predicho por el modelo para la i-ésima observación.

3.2.2. Raíz del error cuadrático medio

La raíz del error cuadrático medio (RMSE, por sus siglas en inglés) es otra medida de evaluación de modelos de regresión que se calcula a partir del error cuadrático medio (MSE) y se utiliza para medir la discrepancia entre los valores predichos por el modelo y los valores reales de la variable dependiente.

La fórmula matemática de la raíz del error cuadrático medio es:

$$RMSE = \sqrt{\frac{\Sigma(y_i - \hat{y}_i)^2}{n}}$$

Donde $RMSE$ es la raíz del error cuadrático medio. n es el número de observaciones en el conjunto de datos. y_i es el valor real de la variable dependiente en la i-ésima observación. $\hat{y}$ es el valor predicho por el modelo para la i-ésima observación.

3.2.3. Error absoluto medio

El error absoluto medio (MAE, por sus siglas en inglés) es una medida de evaluación de modelos de regresión que se utiliza para medir la discrepancia promedio entre los valores predichos por el modelo y los valores reales de la variable dependiente.

La fórmula matemática del MAE es:

$$MAE = \frac{\Sigma|y_i - \hat{y}_i|}{n}$$

Donde MAE es el error absoluto medio. n es el número de observaciones en el conjunto de datos. y_i es el valor real de la variable dependiente en la i-ésima observación. $\hat{y}$ es el valor predicho por el modelo para la i-ésima observación.

3.2.4. Error cuadrático relativo

El error cuadrático relativo (RSE) es una medida de evaluación de modelos de regresión que se utiliza para medir la discrepancia relativa entre los valores predichos por el modelo y los valores reales de la variable dependiente. A diferencia del error cuadrático medio (MSE) y la raíz del error cuadrático medio (RMSE), que se basan en errores absolutos, el RSE considera la proporción de la discrepancia en relación con el valor real de la variable dependiente.

La fórmula matemática del RSE es:

$$RSE = \frac{\Sigma(y_i - \hat{y}_i)^2}{\Sigma(y_i - \bar{y})^2}$$

Donde RSE es el error cuadrático relativo. y_i es el valor real de la variable dependiente en la i-ésima observación. $\hat{y}$ es el valor predicho por el modelo para la i-ésima observación. $\bar{y}$ es el valor promedio en el conjunto de datos.

3.2.5. Coeficiente de correlación de Pearson

El coeficiente de correlación de Pearson (PCC), es una medida estadística que evalúa la fuerza y dirección de la relación lineal entre dos variables continuas. Es ampliamente utilizado para medir la correlación entre dos variables, donde un valor de +1 indica una correlación perfectamente positiva, un valor de -1 indica una correlación perfectamente negativa, y un valor de 0 indica una falta de correlación.

El coeficiente de Pearson se calcula como la covarianza entre las dos variables dividida por el producto de las desviaciones estándar de las dos variables. La fórmula matemática del coeficiente de Pearson es:

$$r = \frac{\Sigma(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\Sigma(x_i - \bar{x})^2 \Sigma(y_i - \bar{y})_i}}$$

Donde r es el coeficiente de Pearson. x_i y y_i son los valores de las dos variables continuas en la i-ésima observación. $\bar{x}$ y $\bar{y}$ son las medias de las dos variables continuas.

3.2.6. Coeficiente de determinación

El coeficiente de determinación (R^2) es una medida estadística que indica la proporción de la variabilidad de una variable dependiente que es explicada por una variable independiente o un modelo de regresión. Es una medida comúnmente utilizada para evaluar la calidad de ajuste de un modelo de regresión y determinar qué tan bien se ajusta el modelo a los datos observados.

El coeficiente de determinación se expresa como un valor entre 0 y 1, donde 0 indica que el modelo no explica ninguna variabilidad de la variable dependiente, y 1 indica que el modelo explica toda la variabilidad de la variable dependiente. Un valor de R2 cercano a 1 indica que el modelo es capaz de explicar una gran proporción de la variabilidad de la variable dependiente, mientras que un valor cercano a 0 indica que el modelo no explica mucha variabilidad.

La fórmula matemática del coeficiente de determinación es:

$$R^2 = 1 - \frac{\Sigma(y_i - \hat{y}_i)^2}{\Sigma(y_i - \bar{y})^2}$$

Donde R^2 es el coeficiente de determinación. y_i es el valor real de la variable dependiente en la i-ésima observación. $\hat{y}$ es el valor predicho por el modelo para la i-ésima observación. $\bar{y}$ es el valor promedio en el conjunto de datos.

4. Materiales y métodos

Para el desarrollo de esta práctica se utilizó el lenguaje de programación Python en su versión 3.10, con el que se diseñó un script para cumplir los objetivos de la práctica.

4.1. Conjunto de datos utilizado

El conjunto de datos utilizado en el desarrollo de esta práctica consta de información relacionada a la disponibilidad y precio de acciones NASDAQ. Dicho conjunto de datos consta de 5 atributos y 1 variable objetivo, sumando un total de 126 instancias sin presencia de datos faltantes.

Los atributos y la variable objetivo se puede categorizar de la siguiente forma:

- Ordinales
  - Date: Fecha del día correspondiente a la instancia.
- Continuos
  - Close/Last (variable objetivo): Precio al final del día.
  - Volume: Cantidad de acciones disponibles.
  - Open: Precio al iniciar el día.
  - High: Precio más alto alcanzado durante el día.
  - Low: Precio más bajo alcanzado durante el día.

4.2. Métricas para evaluación

Tratándose del diseño de modelos de regresión que permitan predecir el valor futuro de las acciones NASDAQ, se optó por realizar una función que genera métricas para datos continuos de las predicciones realizadas por los algoritmos. Dicha función arroja las métricas mostradas a continuación:

- Error cuadrático medio (MSE).
- Raíz del error cuadrático medio (RMSE).
- Error absoluto medio (MAE).
- Error cuadrático relativo (RSE).
- Coeficiente de correlación de Pearson (PCC).
- Coeficiente de determinación (R^2).

4.3. Pre-procesamiento de datos

4.3.1. Limpieza de datos

El conjunto de datos contenía observaciones en formato de texto, por ejemplo, para las observaciones del atributo *Date* podíamos encontrar valores *12/02/2022*, y para el caso de los atributos relacionados a precio podíamos encontrar valores como *$105.54*.

Para lidiar con los valores establecidos como texto de forma nativa, se diseñaron funciones que permiten convertir el atributo *Date* a un formato de fecha nativo de Python, mientras que los atributos relacionados a precio se transformaron a valores tipo *float*. Adicionalmente, se realizó un cambio en el orden de los datos, debido a que en un principio se encontraban de forma descendente (la fecha más reciente en la primer instancia).

Las Tablas 1 y 2 presentan una muestra de los datos originales del conjunto de datos (Tabla 1) y el resultado del proceso de limpieza de datos (Tabla 2).

Tabla 1: Datos originales

Date	Close/Last	Volume	Open	High	Low
12/02/2022	$105.05	7916878	$102.02	$105.54	$101.82
12/01/2022	$103.37	7452313	$102.33	$103.56	$101.95
11/30/2022	$102.2	15000770	$99.05	$102.56	$98.52
11/29/2022	$98.66	4423921	$98.96	$99.33	$98.2
11/28/2022	$98.66	5257862	$98.99	$100.16	$98.56

Tabla 2: Datos transformados

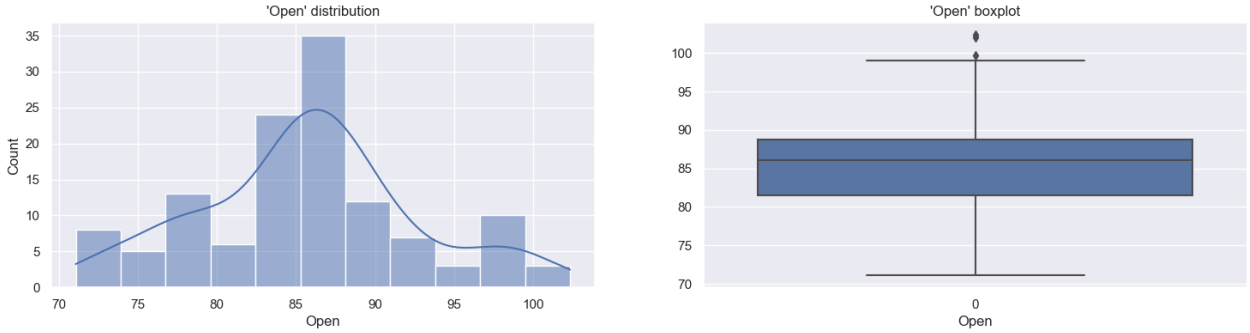
Date	Close/Last	Volume	Open	High	Low
2022-06-06	78.98	7239429	79.70	81.2999	78.53
2022-06-07	79.47	5517101	78.57	79.7500	78.26
2022-06-08	78.47	5178253	78.90	79.7698	78.26
2022-06-09	78.91	13248100	77.97	80.2700	77.73
2022-06-10	75.67	8695476	77.02	77.7900	75.66

4.3.2. Normalización de datos

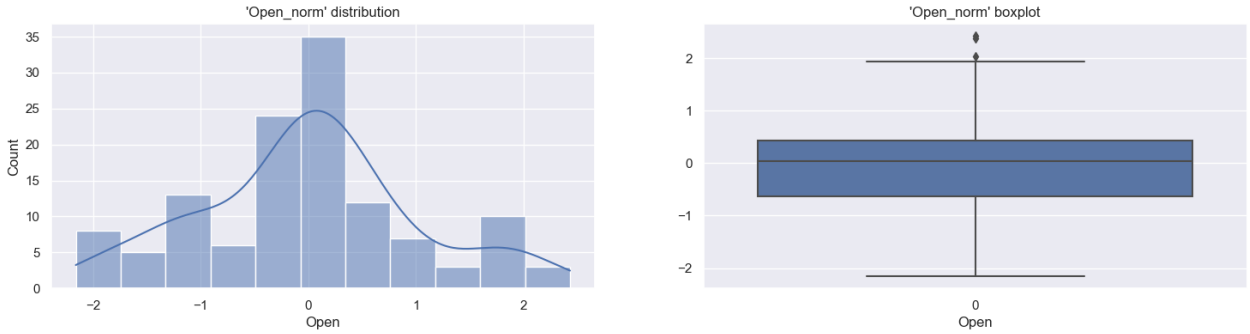
Todos los atributos de tipo continuo se sometieron a un proceso de normalización, donde se utilizaron 2 métodos de normalización para datos continuos, normalización logaritmica (para el atributo *Volume*) y normalización z-score para el resto de los atributos continuos (excluyendo la variable objetivo).

La elección del método de normalización para cada uno de los atributos se realizó mediante un análisis de la distribución de los datos de cada uno de los atributos, donde se llegó a determinar que los atributos relacionados con precios mostraban una tendencia a una distribución normal, mientras que el atributo *Volume* mostraba una distribución sesgada positiva con valores en sus observaciones de la magnitud de millones.

Las Figuras 1 y 2 muestran un ejemplo del proceso de normalización de los atributos continuos. Donde de lado izquierdo se puede observar las gráficas de distribución previas y posterior al proceso de normalización, mientras que del lado derecho se muestran los diagramas de caja correspondientes a los instantes antes mencionados.



(a) Distribución previa a normalizar.



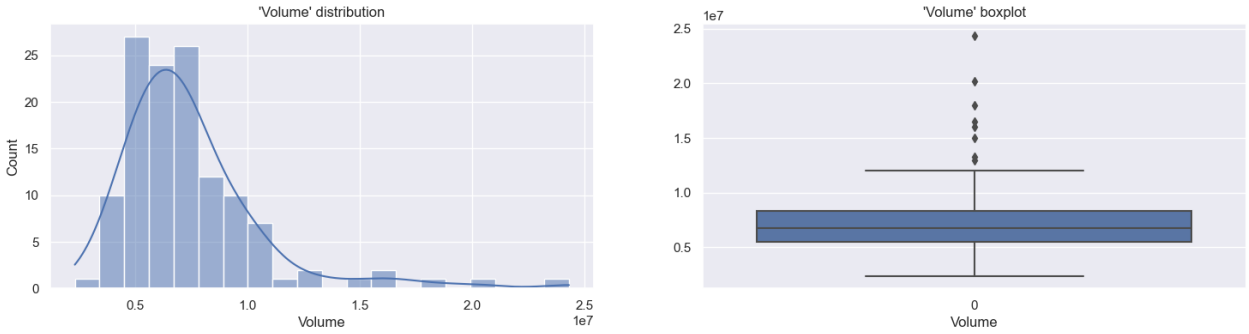
(b) Distribución posterior a normalizar.

Figura 1: Distribución de los datos del atributo *Open*.

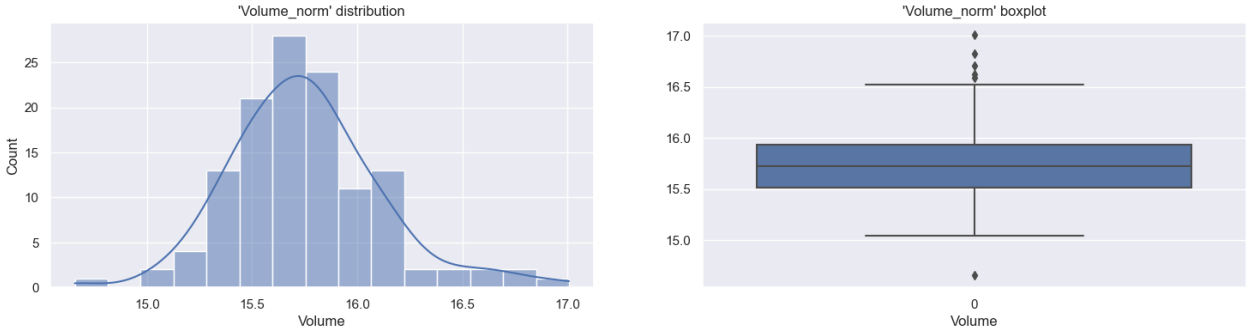
4.3.3. Extracción de características

Previo al desarrollo de los algoritmos de regresión se realizó un proceso de extracción de características, esto con el objetivo de seleccionar los atributos de mayor relación con la variable objetivo.

Junto con este proceso se realizó la segmentación del atributo *Date* en dos atributos: *Month* y *Day*, esto con el objetivo de buscar relaciones cíclicas con la variable objetivo.



(a) Distribución previa a normalizar.



(b) Distribución posterior a normalizar.

Figura 2: Distribución de los datos del atributo *Volume*.

Mediante la obtención de un mapa de correlación (Figura 3) se determinó que los atributos de mayor relación con la variable objetivo son: *Open*, *High*, *Low* y *Month*.

Una vez identificados los atributos de mayor relación con la variable objetivo, se procedió a segmentar el conjunto de datos en una proporción 80:20, tomando como conjunto de entrenamiento a las primeras 100 instancias del conjunto de datos, y las restantes 26 se designaron como conjunto de prueba. La Figura 4 muestra la segmentación de estos datos en la variable objetivo.

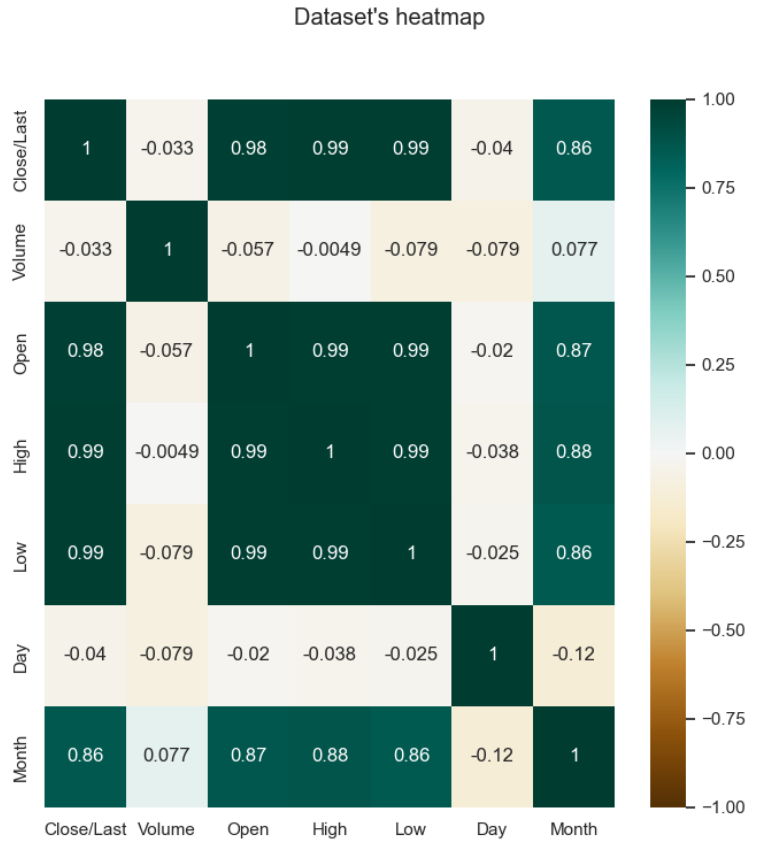


Figura 3: Mapa de correlación del conjunto de datos.

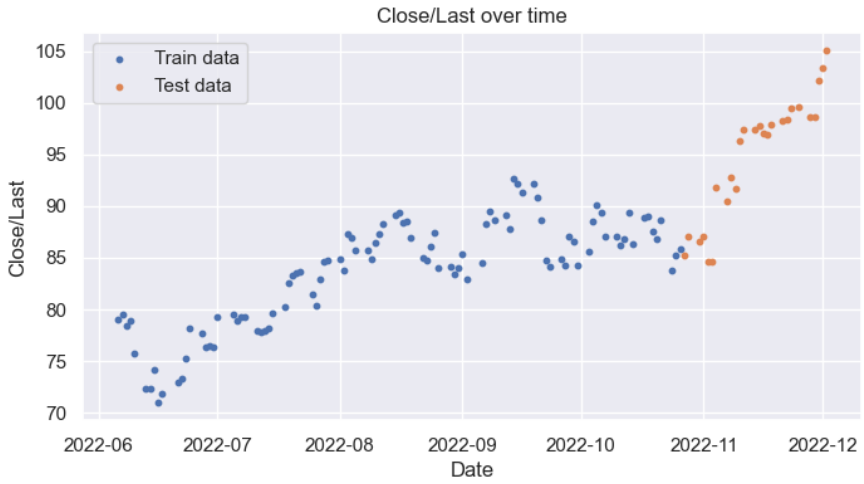


Figura 4: Conjuntos de entrenamiento y prueba vistos respecto al tiempo.

4.4. Algoritmos de regresión

Enfocándose en uno de los objetivos principales de este trabajo, se implementaron 4 algoritmos de regresión, los cuales fueron:

- Regresión lineal simple: Se selecciona debido a que tenemos atributos con alto nivel de correlación. Se seleccionará solamente uno para esta regresión.
- Regresión polinomial. Existe la posibilidad de que la relación entre la variable objetivo y los atributos de alta correlación no sea lineal. Se selecionará un atributo y se modelará como polinomio.
- Regresión lineal múltiple. Retomando el hecho de tener varios atributos con alta correlación, es muy probable que el utilizar todos esos atributos al mismo tiempo favorezca el desempeño.
- Regresión Ridge. Nos permite ajustar el modelo gracias al sistema de penalización, por lo que en casos como el nuestro (varios atributos con alta correlación) será benéfico para reducir los efectos de alta dimensionalidad.

Dichos algoritmos fueron evaluados con la metodología presentada en secciones anteriores.

5. Resultados

Las Figuras 5-8 muestran las predicciones de los modelos de regresión implementados, dichas predicciones se realizaron tanto en el conjunto de prueba con en el conjunto de desarrollo.

De forma general se observan predicciones acertadas, pero al revisar con detalle las métricas obtenidas para cada modelo de regresión (Tabla 3) se puede observar claramente que el modelo que obtuvo predicciones más cercanas a la realidad fue el modelo de regresión lineal múltiple (Figura 7).

Una consideración importante también es el grado del polinomio de la regresión polinomial (Figura 6) se implementó considerando un grado 2, lo cuál puede generar que se disminuya su desempeño pero a su vez previene el sobre-entrenamiento.

Tabla 3: Comparación de métricas de desempeño para cada modelo de regresión.

Algoritmo de regresión	MSE	RMSE	MAE	RSE	PCC	R2
Lineal simple	2.71	1.65	1.28	0.08	0.98	0.92
Polinomial	8.22	2.87	2.40	0.24	0.97	0.76
Lineal múltiple	0.33	0.57	0.49	0.01	1.00	0.99
Ridge	0.64	0.80	0.60	0.02	0.99	0.98

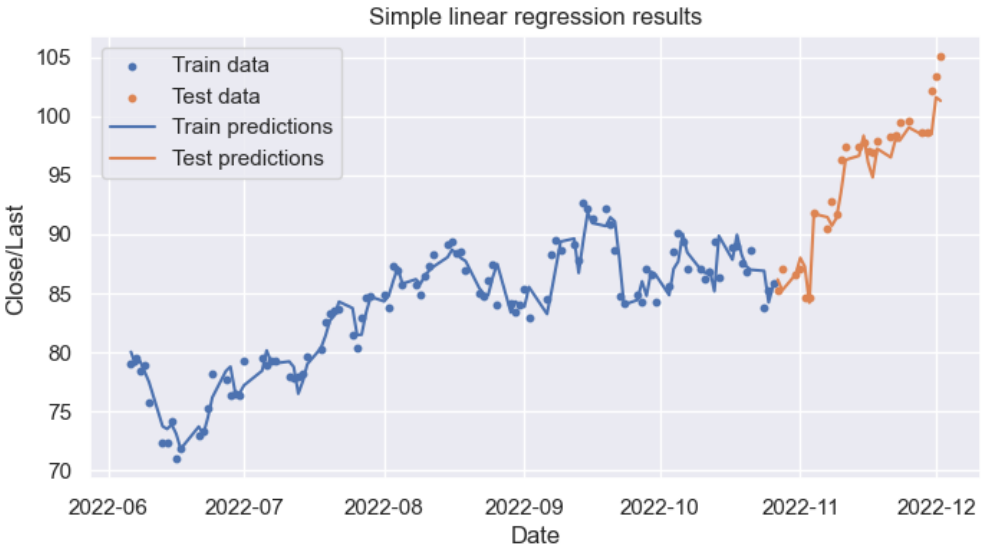


Figura 5: Resultados de regresión lineal simple.

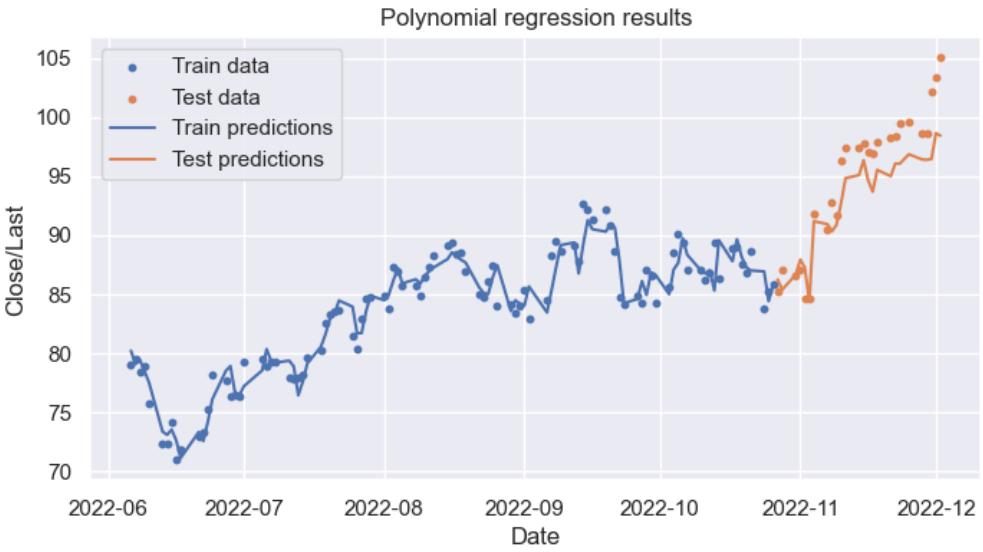


Figura 6: Resultados de regresión polinomial.

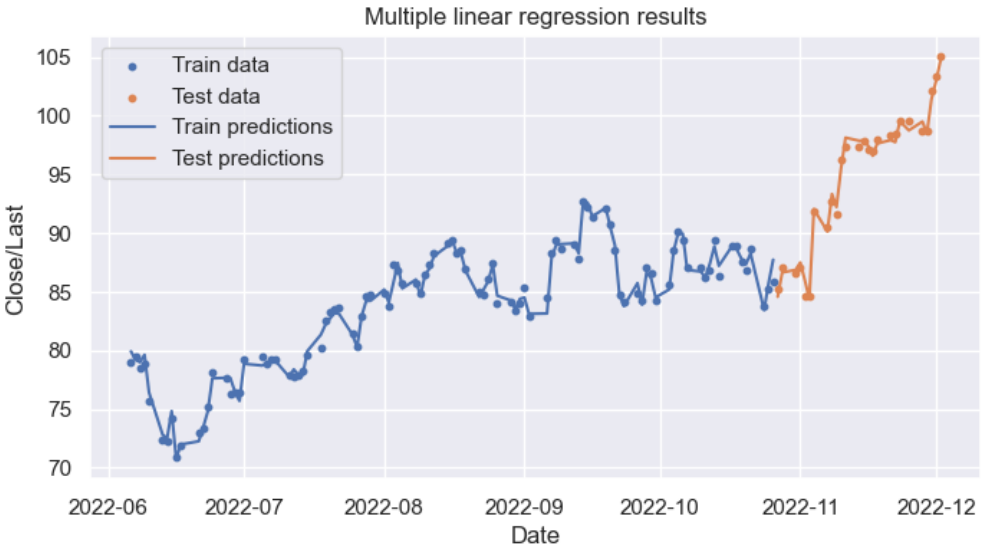


Figura 7: Resultados de regresión lineal multiple.

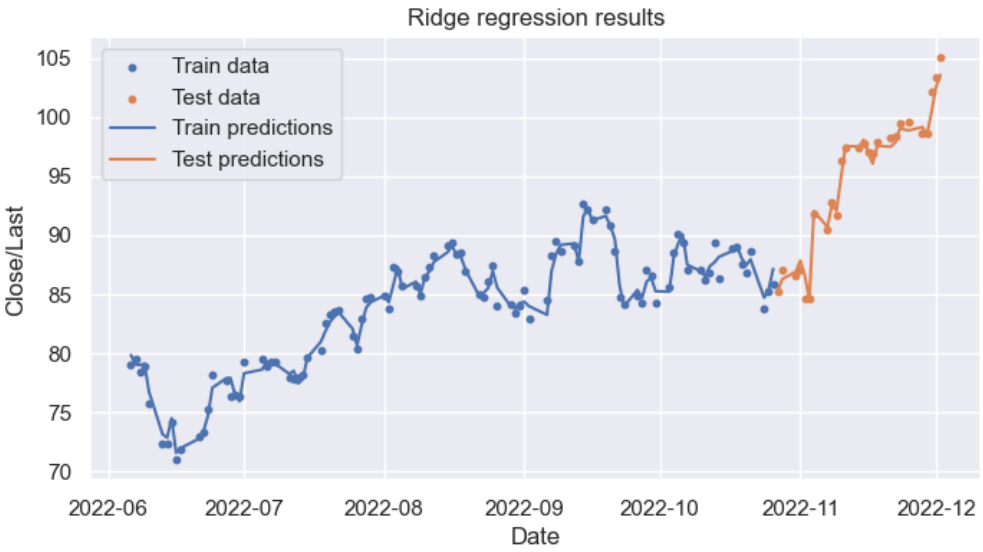


Figura 8: Resultados de regresión Ridge.

6. Conclusiones

Relacionado a la selección de algoritmos de regresión linea. A pesar de existir una gran variedad de algoritmos, podemos considerar que hemos probado los algoritmos adecuados a la situación específica del conjunto de datos, ya que de forma general todos arrojaron resultados favorables. Sería prudente probar otro tipo de algoritmos que permitan considerar atributos externos para mejorar los modelos de regresión.

Respecto al uso de métricas de desempeño para el monitoreo y selección del mejor algoritmo de regresión. Como se muestra en la fundamentación teórica, todas las métricas son útiles para evaluar los algoritmos de regresión, sin embargo, algunas lograr proporcionar información útil para determinados casos de uso. En específico para nuestro caso, donde contamos con una cantidad considerable de valores atípicos, la métrica MAE es de gran utilidad, ya que nos permite obtener una métrica de desempeño que no se verá muy afectada por dichos valores atípicos.

Referencias bibliográficas

[1] M. A. Aceves Fernández, *Inteligencia artificial para programadores con prisa*. Universo de Letras, 2021, ISBN: 9788418854613.

[2] C. M. Bishop, *Pattern Recognition and Machine Learning (Information Science and Statistics)*, 1st ed. Springer, 2007, ISBN: 9780387310732.

[3] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning*. Springer, 2008, ISBN: 9780387848846.

[4] K. P. Murphy, *Machine learning : a probabilistic perspective*. MIT Press, 2013, ISBN: 9780262018029.

A. Código documentado

El código completo y funcional se puede encontrar anexo en el archivo *zip* compartido en conjunto con este reporte, así como en las secciones anexas.

A.1. Practica4_EnriqueMenaCamilo.py

Script implementado para el desarrollo de la práctica.

```python
#!/usr/bin/env python3
"""Machine learning: Practice 4.
Continuous data metrics and regression.
By: Enrique Mena Camilo.
"""

import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LinearRegression, Ridge
from sklearn.preprocessing import PolynomialFeatures

sns.set_theme(style="darkgrid")
pd.set_option("display.max_columns", None)
pd.set_option("display.max_rows", None)

__author__ = "Enrique Mena Camilo"
__email__ = "enriquemece97@gmail.com"


def currency_str_to_float(value: str) -> float:
    """Converts a currency formatted string to a float value.

    :param str value: String to convert, e.g. '£135.04' or '£946,542.094'
    :return float: String converted to float, e.g. 135.09 or 946542.094
    """
    return float(value.replace("$", "").replace(",", "") if isinstance(value, str) else value)


def get_histplot(data: pd.Series, title: str):
    """Generates the histogram of the requested data.

    :param pd.Series data: Data to analyze
    :param str title: Figure title
    """
    fig, ax = plt.subplots(1, 1)
    fig.set_size_inches(8, 4)
    sns.histplot(data, ax=ax, kde=True)
    ax.set_title(f"'{title}' distribution")
    plt.savefig(f"./figures/histplot_{title.lower()}.png", bbox_inches="tight")
    fig.show()


def get_boxplot(data: pd.Series, title: str):
    """Generates the boxplot of the requested data.
```

```
49
50      :param pd.Series data: Data to analyze
51      :param str title: Figure title
52      """
53      fig, ax = plt.subplots(1, 1)
54      fig.set_size_inches(8, 4)
55      sns.boxplot(data, ax=ax)
56      ax.set_title(f"'{title}' boxplot")
57      plt.savefig(f"./figures/boxplot_{title.lower()}.png", bbox_inches="tight")
58      fig.show()
59
60
61  def get_distribution_plots(data: pd.Series, title: str):
62      """Generates the histogram and boxplot of the requested data.
63
64      :param pd.Series data: Data to analyze
65      :param str title: Figure title
66      """
67      fig, axes = plt.subplots(1, 2)
68      fig.set_size_inches(18, 4)
69      sns.histplot(data, ax=axes[0], kde=True)
70      axes[0].set_title(f"'{title}' distribution")
71      sns.boxplot(data, ax=axes[1])
72      axes[1].set_title(f"'{title}' boxplot")
73      axes[1].set_xlabel(data.name)
74      plt.savefig(f"./figures/distribution_{title.lower()}.png", bbox_inches="tight")
75      fig.show()
76
77
78  def plot_predictions(train_set: np.ndarray, test_set: np.ndarray, train_pred: np.ndarray, test_pred: np.ndarray, title: str):
79      """Plots the results of the predictions and the actual data.
80
81      :param np.ndarray train_set: Actual train data
82      :param np.ndarray test_set: Actual test data
83      :param np.ndarray train_pred: Prediected train data
84      :param np.ndarray test_pred: Predictes test data
85      :param str title: Figure title
86      """
87      fig = plt.figure(figsize=(8, 4))
88      plt.scatter(train_set[0], train_set[1], s=10, label="Train data")
89      plt.scatter(test_set[0], test_set[1], s=10, label="Test data")
90      plt.plot(train_set[0], train_pred, linestyle="solid", label="Train predictions")
91      plt.plot(test_set[0], test_pred, linestyle="solid", label="Test predictions")
92      plt.title(f"{title} results")
93      plt.xlabel("Date")
94      plt.ylabel("Close/Last")
95      plt.legend()
96      plt.savefig(f"./figures/{'_'.join(title.lower().split())}_results", bbox_inches="tight")
97      fig.show()
98
99
100 def get_heatmap(data: pd.DataFrame):
101     """Build dataset correlation heatmap.
102
103     :param pd.DataFrame data: Data to analyze
104     """
105     fig, ax = plt.subplots(1)
106     fig.suptitle(f"Dataset's heatmap")
107     fig.set_size_inches(8, 8)
108     sns.heatmap(data=data.corr(), annot=True, cmap='BrBG', vmin=-1, vmax=1, ax=ax)
109     plt.savefig(f"./figures/dataset_heatmap.png", bbox_inches='tight')
```

```
110         fig.show()
111
112
113     def MSE(actual: np.ndarray, predicted: np.ndarray) -> float:
114         """Mean Squared Error (MSE).
115
116         :param np.ndarray actual: Actual data
117         :param np.ndarray predicted: Predicted data
118         """
119         return np.sum((actual - predicted)**2) / len(actual)
120
121
122     def RMSE(actual: np.ndarray, predicted: np.ndarray) -> float:
123         """Root Mean Squared Error (RMSE).
124
125         :param np.ndarray actual: Actual data
126         :param np.ndarray predicted: Predicted data
127         """
128         return np.sqrt(np.sum((actual - predicted)**2) / len(actual))
129
130
131     def MAE(actual: np.ndarray, predicted: np.ndarray) -> float:
132         """Mean Absolute Error (MAE).
133
134         :param np.ndarray actual: Actual data
135         :param np.ndarray predicted: Predicted data
136         """
137         return np.sum(np.abs(actual - predicted)) / len(actual)
138
139
140     def RSE(actual: np.ndarray, predicted: np.ndarray) -> float:
141         """Relative Squared Error (RSE).
142
143         :param np.ndarray actual: Actual data
144         :param np.ndarray predicted: Predicted data
145         """
146         rse_n = np.sum((actual - predicted)**2)
147         rse_d = np.sum((actual - actual.mean())**2)
148         return rse_n / rse_d
149
150
151     def PCC(actual: np.ndarray, predicted: np.ndarray) -> float:
152         """Pearson correlation coefficient (PCC)
153
154         :param np.ndarray actual: Actual data
155         :param np.ndarray predicted: Predicted data
156         """
157         pcc_n = len(actual)*np.sum(actual * predicted) - np.sum(actual)*np.sum(predicted)
158         pcc_d1 = len(actual)*np.sum(actual**2) - np.sum(actual)**2
159         pcc_d2 = len(actual)*np.sum(predicted**2) - np.sum(predicted)**2
160         pcc_d = np.sqrt(pcc_d1 * pcc_d2)
161         return pcc_n / pcc_d
162
163
164     def R2(actual: np.ndarray, predicted: np.ndarray) -> float:
165         """R-squared (R2, R^2).
166
167         :param np.ndarray actual: Actual data
168         :param np.ndarray predicted: Predicted data
169         """
170         r2_n = np.sum((predicted - actual)**2)
```

```python
    r2_d = np.sum((actual -  actual.mean())**2)
    return 1 - (r2_n / r2_d)


def evaluate_regression(actual: np.ndarray, predicted: np.ndarray):
    """Gets all regression evaluation metrics.

    :param np.ndarray actual: Actual data
    :param np.ndarray predicted: Predicted data
    """
    print(f"Mean Square Error: {round(MSE(actual, predicted), 2)}")
    print(f"Root Mean Square Error: {round(RMSE(actual, predicted), 2)}")
    print(f"Mean Absolute Error: {round(MAE(actual, predicted), 2)}")
    print(f"Relative Square Error: {round(RSE(actual, predicted), 2)}")
    print(f"Pearson Correlation Coeficient: {round(PCC(actual, predicted), 2)}")
    print(f"R-squared: {round(R2(actual, predicted), 2)}")


if __name__ == "__main__":
    # === Data cleaning ===
    data = pd.read_csv("../data/Nasdaq.csv")
    print(data.head())

    # It is necessary to give the correct format to the columns
    data["Date"] = pd.to_datetime(data["Date"])
    data["Close/Last"] = data["Close/Last"].apply(currency_str_to_float)
    data["Open"] = data["Open"].apply(currency_str_to_float)
    data["High"] = data["High"].apply(currency_str_to_float)
    data["Low"] = data["Low"].apply(currency_str_to_float)

    print(data.head())

    # The data is sorted descending, we need them sorted ascending
    data = data.sort_values(by="Date", ascending=True, ignore_index=True)
    print(data.head())
    print(data.describe().T)

    # Does the dataset have missing data?
    print(data.info())
    # R: No, the dataset is complete

    # === Data normalization ===
    # What about the distribution of the data?
    columns = [
        {"name": "Volume", "title": "Volume"},
        {"name": "Open", "title": "Open"},
        {"name": "High", "title": "High"},
        {"name": "Low", "title": "Low"}
    ]

    for column in columns:
        get_distribution_plots(data[column["name"]], column["title"])


    # Columns Close/Last, Open, High and Low present a distribution close to a normal distribution, and their range of
    # values could be extended in the future. z-score normalization will be used for these columns.
    # Volume present a positive skewed distribution, and its observations show a magnitude of millions.
    # Log normalization will be used for this column.
    data_norm = data.copy()

    scaler = StandardScaler()
```

```
232     data_norm["Close/Last"] = data_norm["Close/Last"]
233     data_norm["Open"] = scaler.fit_transform(data_norm[["Open"]])
234     data_norm["High"] = scaler.fit_transform(data_norm[["High"]])
235     data_norm["Low"] = scaler.fit_transform(data_norm[["Low"]])
236     data_norm["Volume"] = np.log(data_norm["Volume"])
237
238     columns = [
239         {"name": "Volume", "title": "Volume_norm"},
240         {"name": "Open", "title": "Open_norm"},
241         {"name": "High", "title": "High_norm"},
242         {"name": "Low", "title": "Low_norm"}
243     ]
244
245     for column in columns:
246         get_distribution_plots(data_norm[column["name"]], column["title"])
247
248     # Additionally, We want to use month and day as attributes
249     data_norm["Day"] = data_norm["Date"].apply(lambda x: x.day)
250     data_norm["Month"] = data_norm["Date"].apply(lambda x: x.month)
251
252     print(data_norm.head())
253     print(data_norm.describe().T)
254
255     # === Feature selecction ===
256     # We want to predict Close/Last using regression, which attributes will be most useful to us?
257     get_heatmap(data_norm.drop(columns=["Date"]))
258     # R: We will use Open, High, Low and Month as predictors of Close/Last
259
260     date = data_norm[["Date"]].to_numpy()
261     Y = data_norm[["Close/Last"]].to_numpy()
262     X = data_norm[["Open", "High", "Low", "Month"]].to_numpy()
263
264     fig, axes = plt.subplots(2, 2)
265     fig.set_size_inches(14, 8)
266     fig.suptitle("Relation between target variable andh choiced features")
267
268     axes[0, 0].scatter(X[:, 0], Y)
269     axes[0, 0].set_ylabel("Close/Last")
270     axes[0, 0].set_xlabel("Open")
271
272     axes[0, 1].scatter(X[:, 1], Y)
273     axes[0, 1].set_ylabel("Close/Last")
274     axes[0, 1].set_xlabel("High")
275
276     axes[1, 0].scatter(X[:, 2], Y)
277     axes[1, 0].set_ylabel("Close/Last")
278     axes[1, 0].set_xlabel("Low")
279
280     axes[1, 1].scatter(X[:, 3], Y)
281     axes[1, 1].set_ylabel("Close/Last")
282     axes[1, 1].set_xlabel("Month")
283
284     plt.tight_layout()
285     plt.savefig(f"./figures/correlations.png", bbox_inches="tight")
286     fig.show()
287
288     # Create train and test set
289     split_idx = int(np.floor(len(X) * 0.80))
290     date_train, date_test = date[:split_idx], date[split_idx:]
291     X_train, X_test = X[:split_idx], X[split_idx:]
292     Y_train, Y_test = Y[:split_idx], Y[split_idx:]
```

```
293
294     print(f"Total data: {len(Y)}")
295     print(f"Train data: {len(Y_train)}")
296     print(f"Test data: {len(Y_test)}")
297
298     # === Target variable ===
299     # Let's see the evolution of Close/Last over time
300     fig = plt.figure(figsize=(8, 4))
301     plt.scatter(date_train, Y_train, s=10, label="Train data")
302     plt.scatter(date_test, Y_test, s=10, label="Test data")
303     plt.title("Close/Last over time")
304     plt.xlabel("Date")
305     plt.ylabel("Close/Last")
306     plt.legend()
307     plt.savefig(f"./figures/closelast_time.png", bbox_inches="tight")
308     fig.show()
309
310     # === Simple linear regression ===
311     choiced_feature = 0
312
313     slr = LinearRegression()
314     slr.fit(X_train[:, choiced_feature].reshape(-1, 1), Y_train)
315     slr_train = slr.predict(X_train[:, choiced_feature].reshape(-1, 1))
316     slr_test = slr.predict(X_test[:, choiced_feature].reshape(-1, 1))
317
318     plot_predictions((date_train, Y_train), (date_test, Y_test), slr_train, slr_test, "Simple linear regression")
319
320     print("===== Simple Linear Regression Evaluation =====")
321     evaluate_regression(Y_test, slr_test)
322
323     # === Polynomial regression ===
324     choiced_feature = 0
325     degree = 2
326
327     pr_feature = PolynomialFeatures(degree=degree, include_bias=False)
328     pr_feature = pr_feature.fit_transform(X_train[:, choiced_feature].reshape(-1, 1))
329
330     pr_feature_test = PolynomialFeatures(degree=degree, include_bias=False)
331     pr_feature_test = pr_feature_test.fit_transform(X_test[:, choiced_feature].reshape(-1, 1))
332
333     pr = LinearRegression()
334     pr.fit(pr_feature, Y_train)
335     pr_train = pr.predict(pr_feature)
336     pr_test = pr.predict(pr_feature_test)
337
338     plot_predictions((date_train, Y_train), (date_test, Y_test), pr_train, pr_test, "Polynomial regression")
339
340     print("===== Polynomial Regression Evaluation =====")
341     evaluate_regression(Y_test, pr_test)
342
343     # === Multiple linear regression ===
344     mlr = LinearRegression()
345     mlr.fit(X_train, Y_train)
346     mlr_train = mlr.predict(X_train)
347     mlr_test = mlr.predict(X_test)
348
349     plot_predictions((date_train, Y_train), (date_test, Y_test), mlr_train, mlr_test, "Multiple linear regression")
350
351     print("===== Multiple Linear Regression Evaluation =====")
352     evaluate_regression(Y_test, mlr_test)
353
```

```
354     # === Ridge regression ===
355     rdr = Ridge(alpha=1.0)
356     rdr.fit(X_train, Y_train)
357     rdr_train = rdr.predict(X_train)
358     rdr_test = rdr.predict(X_test)
359
360     plot_predictions((date_train, Y_train), (date_test, Y_test), rdr_train, rdr_test, "Ridge regression")
361
362     print("===== Ridge Regression Evaluation =====")
363     evaluate_regression(Y_test, rdr_test)
364
365     input()
```