

Avance 2:

Ingeniería de Características

(Modelos Base)

Presentado por Equipo 15:

Jose Fabricio Barahona Amaya

Andrés Eduardo Figueroa García

Isaac Francisco Viramontes Castillo

Profesor Titular: Dr. Luis Eduardo Falcón Morales

Proyecto Integrador | Fecha: 12/05/2024

Índice

| | |
|---|----|
| Índice | 2 |
| Optimización de Portafolio de Inversión | 3 |
| Antecedentes | 3 |
| Entendimiento del Negocio | 3 |
| Formulación del Problema | 3 |
| Contexto | 4 |
| Objetivos..... | 5 |
| General..... | 5 |
| Específicos | 6 |
| Preguntas de Negocio | 6 |
| Involucrados | 6 |
| Obtención de los Datos..... | 6 |
| Noticias | 6 |
| Bing News API | 6 |
| Google News Web Scrapping | 10 |
| Entendimiento de los Datos..... | 13 |
| Descripción de los datos..... | 13 |
| Identificación de variables | 17 |
| Análisis exploratorio de datos..... | 18 |
| Valores Faltantes:..... | 18 |
| Estadísticas resumidas del Conjunto de Datos..... | 18 |
| Valores Atípicos | 19 |
| Otras observaciones con respecto al análisis de datos | 22 |
| Registro de noticias..... | 24 |
| Anexos | 36 |
| Github – Equipo 15..... | 36 |
| Script Python | 36 |
| CSV Generado | 36 |
| Bibliografía | 37 |

Optimización de Portafolio de Inversión

Este proyecto pretende implementar una alternativa por medio del uso de los sistemas apoyados por la inteligencia artificial, para asistir en la toma de decisiones financieras orientadas a valorar acciones empresas, su utilidad y practicidad para áreas financieras y personas particulares que compran y venden acciones en mercados de valores para obtener beneficios sobre sus inversiones.

Antecedentes

- Empresa: TeamUp Costa Rica, está en búsqueda de implementar productos innovadores basados en inteligencia artificial, están interesados en poner el mercado los servicios de asesoría en inversiones financieras con instrumentos disponibles dentro y fuera del país en donde opera. Debido a que este proyecto no utiliza datos privados de la compañía, no se requerirá algún tipo de acuerdo de confidencialidad. La empresa se define como una empresa consultora con una red de profesionales con experiencia, conocimientos y la mejor actitud de servicio en servicios de asesoría estratégica, definición de propuestas ganadoras (concursos), arquitectura empresarial, asesoría en procesos de innovación y transformación digital.
- Sector Industrial al que Pertenece: Información en Medios Masivos - Edición de periódicos, revistas, libros, directorios, software y otros materiales
- Lugar de Aplicación: Belén, Heredia, Costa Rica.

Entendimiento del Negocio

Formulación del Problema

El problema está en implementar una solución por medio de la cual se pueda conformar un portafolio de inversión óptimo, el cual en la medida de lo posible pueda predecir comportamientos que se pueden acercar a la realidad en los mercados de capitales y dar soluciones en tiempos útiles emulando el comportamiento de un experto financiero y con la capacidad procesar la información histórica, proveniente de datos estructurados (precios de mercados) y no estructurados (noticias). El modelo te inteligencia artificial se implementará de forma en la que se pueda formular un portafolio en base a la asistencia del computador, en donde se pueda recomendar a los inversionistas un portafolio que les permita obtener mejores beneficios al menor riesgo. También es se contextualizará la investigación a la utilidad en la práctica para inversionistas de Latinoamérica, una región en la que varios países carecen de mercados de capitales locales y opciones de inversión rentables.

Contexto

Los mercados de capitales son una opción de financiamiento para las empresas que desean financiarse vendiendo parte del capital de la empresa por medio de un instrumento llamado acción o por medio de la emisión de deuda. Por otro lado, el comprador o inversionista se vuelve parcialmente propietario de una parte de la compañía en vez de volverse un acreedor y representa una oportunidad de inversión como alternativa al mercado de dinero en el cual suele invertir en bonos que para el emisor son deuda. (Vázquez, 2012, pp. 55-79)

Para que una empresa pueda acceder a vender acciones o emitir deuda en un mercado organizado, esta debe cumplir con una serie de normas y certificaciones que el mercado exige con el fin de trasladar confiabilidad y atracción por para los inversionistas. Las empresas salen al mercado fijando un precio para cada acción el cual ha sido fijado mediante una serie de estudios y métodos de valuación que no siempre pueden ser el precio justo, en algunos casos el precio de la acción luego de la oferta pública inicial aumenta o disminuye drásticamente.

Cuando una corporación emite acciones o deuda por primera vez o agrega acciones al mercado (emite un nuevo paquete de acciones), estas se negocian en el mercado primario y son negociados al precio fijo que el emisor estimó “el más justo”. Una vez colocadas todas las acciones emitidas en el mercado los propietarios de las acciones pueden revenderlas, lo que se lleva a cabo en un mercado secundario.

Los precios de las acciones en mercados secundarios ya no son fijados en base a estudios de valoración si no que son establecidos mediante la oferta y la demanda de la misma, así como la decisión de la empresa de emitir o retirar acciones del mercado mediante la compra de acciones a los accionistas. Hay empresas que son oportunistas y hacen un seguimiento cercano del precio de mercado de sus acciones para emitir nuevas acciones y captar los recursos directamente y quitar esa cuota en parte el mercado secundario. Estas decisiones pueden saturar el mercado y hacer que las acciones bajen de precio.

El método de valoración en mercados secundarios por tanto en las bolsas de valores más importantes del planeta está dado por la negociación entre el que vende y el que compra, o también llamado método de subasta de doble punta. Este método de valuación de acciones es el implementado en las bolsas de valores más grandes del mundo incluyendo la bolsa NYSE (New York Stock Exchange) en la cual cotizan la mayoría de las empresas en Estados Unidos de América.

El riesgo juega un papel importante en el mercado de acciones ya que los dividendos de las acciones van acordes del éxito o el fracaso de la empresa. Por tanto, existe el peligro de perder en su totalidad una inversión realizada con alto riesgo, la cual pudo haber sido atractiva ya que a mayor riesgo mayor ganancias.

Un portafolio financiero es entonces la colección de activos con las cuales cuenta una persona o empresa de los cuales obtiene una utilidad financiera que podría provenir de varias fuentes: intereses en el caso de bonos, depósitos en mercados de dinero, dividendos pagados por acciones en mercados de capitales, la venta de acciones o transacciones en mercados de derivados.

Harry M. Markowitz planteó su teoría del portafolio, en la que, mediante datos históricos, aplicación de covarianzas estadísticas, evaluación de expectativas (ya que el precio de una acción va según lo que esta retornará en el futuro) y valoración del riesgo, optimiza un portafolio de inversión maximizando las ganancias y diversificando el riesgo.

La teoría del portafolio de Markowitz es utilizada por los inversionistas para ayudar a la toma de decisiones de inversión. Sin embargo, el saber las tendencias de los precios de las acciones, sería para los inversionistas una información valiosa para sus finanzas, ya que podrían tomar decisiones adecuadas en el momento indicado. El resultado de la aplicación de la teoría de Markowitz es determinístico ya que se basa en datos históricos y aplicación de estadística.

Los seres humanos generalmente toman decisiones no determinísticas y sorprendentes. Por ejemplo, un ser humano es capaz de identificar a una identidad de otro ser humano con solo mirar a los ojos, ver algún rasgo físico, un patrón de caminado, escuchar una voz o inclusive con solo ver una sombra. Esto es algo que definitivamente le da una ventaja grandísima al ser humano por sobre los sistemas computacionales y modelo matemático alguno. Las decisiones de compra de acciones en algunos casos se vuelven subjetivas.

En los últimos años las ciencias de la computación han sido responsables de representar comportamientos sociales complejos mediante simulaciones y aplicaciones que implementan inteligencia artificial. Los mercados financieros están en la mira de los científicos ya que representan un comportamiento fundamental en el sistema capitalista.

Objetivos

General

- Analizar la importancia y utilidad de invertir en mercados de capitales y la introducción de modelos de inteligencia artificial en la asistencia para la inversión en mercados financieros con el fin que los inversionistas puedan tomar decisiones oportunas y efectivas por medio de la elaboración un portafolio de inversión exitoso donde se maximizan las ganancias y se diversifica el riesgo y su aplicación en economías que carecen de mercados capitales como es caso de la economía hondureña.

Específicos

- Implementar un modelo para configurar un portafolio de inversión apoyado en teoría de portafolio elaborada por Harry M. Markowitz y evaluar su desempeño y utilización.
- Comparar los diferentes modelos de diseño de portafolio.
- Implementar ejecuciones prácticas asistidas por inteligencia artificial en un escenario al alcance.
- Evaluar los resultados de la implementación experimental asistida por computadora de los diferentes modelos.

Preguntas de Negocio

- ¿Qué tan confiables son los resultados de las simulaciones y aplicación de inteligencia artificial a los mercados bursátiles y como se comprueba su efectividad?
- ¿Cómo puede asistir la minería de texto o procesamiento de lenguaje natural en el comportamiento de los mercados financieros y en la toma de decisiones de inversión?

Involucrados

TeamUp:

- Luis Carlos Rivas García - Gerente General

Claustro de profesores:

- Dr. Luis Eduardo Falcón – Profesor Titular

Ejecutores:

- Equipo 15 de Proyecto Integrador

Obtención de los Datos

Noticias

Bing News API

Para la obtención de noticias en un primer momento se quiso hacer uso de un API dentro de Azure para Bing News. De esta prueba se obtuvieron datos preliminares para la entrega pasada. Sin embargo, los créditos de Azure consumidos fueron más de los esperados, por lo que se decidió cambiar de estrategia para la obtención de noticias. De las pruebas anteriores se comparte el siguiente código:

```
import requests  
import json
```

```

import time
import pandas as pd
import yfinance as yf

from datetime import date, timedelta
from google.colab import userdata

df_SP500 = pd.read_csv('S&P500_List.csv')

SP500_symbol = df_SP500['Symbol'].to_list()
SP500_name = df_SP500['Security'].to_list()

daily_register_news = {
    'Symbol': [],
    'Name': [],
    'Source_1': [],
    'Name_1': [],
    'Description_1': [],
    'Source_2': [],
    'Name_2': [],
    'Description_2': [],
    'Source_3': [],
    'Name_3': [],
    'Description_3': [],
    'Status_News': []
}

subscription_key = userdata.get('api_key')
search_url = userdata.get('endpoint') + "/v7.0/search"
headers = {"Ocp-Apim-Subscription-Key" : subscription_key}

for i in range(len(SP500_symbol)):
    rankingResponse = {}
    time.sleep(0.02)
    stock = SP500_symbol[i]
    name = SP500_name[i]

    daily_register_news['Symbol'].append(stock)
    daily_register_news['Name'].append(name)

    # print(name)
    if i % 50 == 0:
        print(f"{int(i/5)}")

    query = f'{name}'
    params = {
        'q': query,
        'count':10,
        'offset': 0,
        'mkt': 'en-US',
        'freshness': 'Month',
        "responseFilter": "News",
        "sortBy": "Relevance"
    }
    response = requests.get(search_url, headers=headers, params=params)
    results = response.json()
    try:
        rankingResponse = results['rankingResponse']
        # print(f"query: {query}; rankingResponse: {rankingResponse}")

```

```

except:
    rankingResponse = {}

if rankingResponse == {}:
    query = f'{stock}'
    params = {
        'q': query,
        'count':10,
        'offset': 0,
        'mkt': 'en-US',
        'freshness': 'Month',
        "responseFilter": "News",
        "sortBy": "Relevance "
    }
    response = requests.get(search_url, headers=headers, params=params)
    results = response.json()
    try:
        rankingResponse = results['rankingResponse']
        # print(f"query: {query}; rankingResponse: {rankingResponse}")
    except:
        rankingResponse = {}

if rankingResponse == {}:
    query = f'{name} {stock}'
    params = {
        'q': query,
        'count':10,
        'offset': 0,
        'mkt': 'en-US',
        'freshness': 'Month',
        "responseFilter": "News",
        "sortBy": "Relevance "
    }
    response = requests.get(search_url, headers=headers, params=params)
    results = response.json()
    try:
        rankingResponse = results['rankingResponse']
        # print(f"query: {query}; rankingResponse: {rankingResponse}")
    except:
        rankingResponse = {}

if rankingResponse == {}:
    query = f'{stock} {name}'
    params = {
        'q': query,
        'count':10,
        'offset': 0,
        'mkt': 'en-US',
        'freshness': 'Month',
        "responseFilter": "News",
        "sortBy": "Relevance "
    }
    response = requests.get(search_url, headers=headers, params=params)
    results = response.json()
    try:
        rankingResponse = results['rankingResponse']
    
```

```

# print(f"query: {query}; rankingResponse: {rankingResponse}")
except:
    rankingResponse = {}

if rankingResponse == {}:
    query = f'NASDAQ {name}'
    params = {
        'q': query,
        'count': 10,
        'offset': 0,
        'mkt': 'en-US',
        'freshness': 'Month',
        "responseFilter": "News",
        "sortBy": "Relevance"
    }
    response = requests.get(search_url, headers=headers, params=params)
    results = response.json()
    try:
        rankingResponse = results['rankingResponse']
        # print(f"query: {query}; rankingResponse: {rankingResponse}")
    except:
        rankingResponse = {}

if rankingResponse == {}:
    query = f'stocks {name}'
    params = {
        'q': query,
        'count': 10,
        'offset': 0,
        'mkt': 'en-US',
        'freshness': 'Month',
        "responseFilter": "News",
        "sortBy": "Relevance"
    }
    response = requests.get(search_url, headers=headers, params=params)
    results = response.json()
    try:
        rankingResponse = results['rankingResponse']
        # print(f"query: {query}; rankingResponse: {rankingResponse}")
    except:
        rankingResponse = {}

if rankingResponse == {}:
    query = f'news {name}'
    params = {
        'q': query,
        'count': 10,
        'offset': 0,
        'mkt': 'en-US',
        'freshness': 'Month',
        # "responseFilter": "News",
        "sortBy": "Relevance"
    }
    response = requests.get(search_url, headers=headers, params=params)
    results = response.json()
    try:

```

```

rankingResponse = results['rankingResponse']
# print(f"query: {query}; rankingResponse: {rankingResponse}")
except:
    rankingResponse = {}

if rankingResponse == {}:
    for j in range(0, 3):
        daily_register_news[f'Source_{j+1}'].append(' ')
        daily_register_news[f'Name_{j+1}'].append(' ')
        daily_register_news[f'Description_{j+1}'].append(' ')

    daily_register_news['Status_News'].append('NOK')
    print(f"{name}: {results}")

else:
    for j in range(0, 3):
        try:
            top = results['news']['value'][j]

            daily_register_news[f'Source_{j+1}'].append(top['contractualRules'][0]['text'])
            daily_register_news[f'Name_{j+1}'].append(top['name'])
            daily_register_news[f'Description_{j+1}'].append(top['description'])

        except:
            daily_register_news[f'Source_{j+1}'].append(' ')
            daily_register_news[f'Name_{j+1}'].append(' ')
            daily_register_news[f'Description_{j+1}'].append(' ')

    daily_register_news['Status_News'].append('OK')

df_news = pd.DataFrame(daily_register_news)
df_news.to_csv('Register_240426.csv', index=False)

```

Google News Web Scrapping

Haciendo uso de Google News se puede hacer búsqueda de las noticias, incluso filtrando por el periodo de tiempo en que se desea hacer la búsqueda. Esto se hace con el siguiente query.

Q = StockName after:YYYY-MM-DD before:YYYY-MM-DD

Esto da como resultado algo similar a lo mostrado en la siguiente imagen:

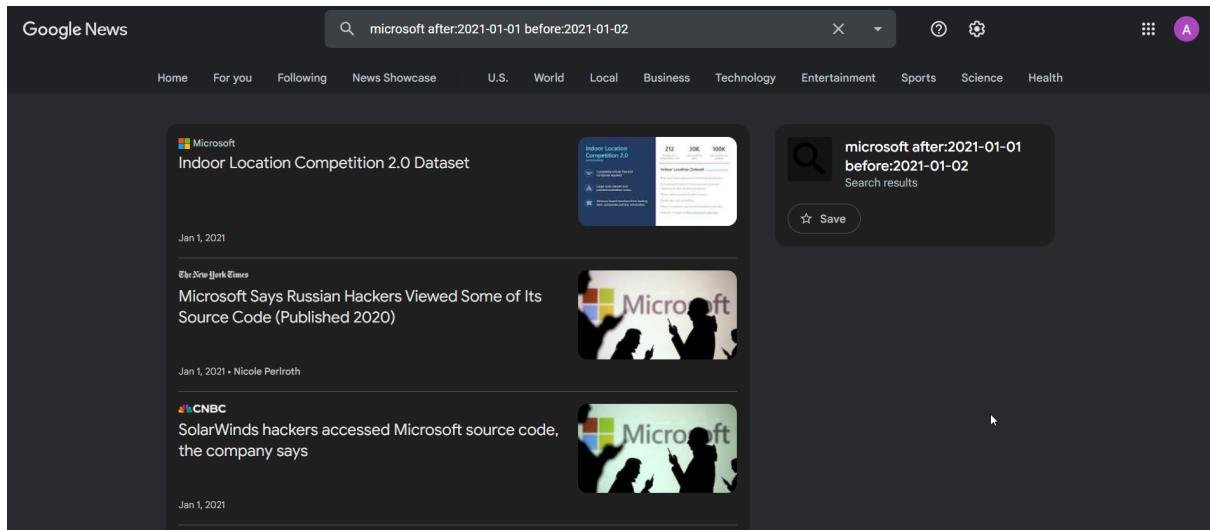


Fig.1: Resultados Google News.

Como se puede observar, el filtro de la búsqueda se aplica de manera correcta, además de poder encontrar varios resultados pertinentes y estos se organizan por relevancia de manera automática.

Otra cuestión importante es que para poder hacer la búsqueda de las noticias en Google News no se podía hacer uso de un HTTP request de tipo GET con la librería estándar de Python, por lo que se recurrió a Selenium con Chrome Driver.

El registro se hace de manera similar a como se hizo el de las APIs, con el cambio que en esta ocasión se tiene que buscar la información de las noticias de manera manual. El código se incluye a continuación.

```
from selenium import webdriver
from selenium.webdriver.chrome.service import Service
from selenium.webdriver.common.by import By
from bs4 import BeautifulSoup
from newspaper import Article
import datetime
import datetime
import pandas as pd
import time

options = webdriver.ChromeOptions()
options.add_argument('--headless')
options.add_argument('--incognito')
options.add_argument('--ignore-certificate-errors')
options.add_argument('user-agent=Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/122.0.0.0 Safari/537.36')

chrome_driver_path = r"C:\Users\aefig\Downloads\chromedriver-win64\chromedriver.exe"

driver = webdriver.Chrome(service=Service(chrome_driver_path), options = options)
```

```

df_SP500 = pd.read_csv('S&P500_List.csv', encoding='utf8')
SP500_symbol = df_SP500['Symbol'].to_list()
SP500_name = df_SP500['Security'].to_list()

daily_register_news_clean = {
    'Symbol': [],
    'Name': [],
    'Source_1': [],
    'Name_1': [],
    'Text_1': [],
    'Source_2': [],
    'Name_2': [],
    'Text_2': [],
    'Source_3': [],
    'Name_3': [],
    'Text_3': [],
    'Source_4': [],
    'Name_4': [],
    'Text_4': [],
    'Source_5': [],
    'Name_5': [],
    'Text_5': []
}

date = datetime.date.fromisoformat('2021-01-05')

while date < datetime.date.fromisoformat('2021-01-16'):
    i_date = str(date)
    e_date = str(date + datetime.timedelta(days=1))

    print(i_date)
    daily_register_news = daily_register_news_clean

    for i in range(len(SP500_symbol)):
        if i % 50 == 0:
            print(f"{int(i/5)}%")

        stock = SP500_symbol[i]
        name = SP500_name[i]

        daily_register_news['Symbol'].append(stock)
        daily_register_news['Name'].append(name)

        url = "https://news.google.com/search?q=" + name + "%20after%3A" + i_date
        + "%20before%3A" + e_date + "&hl=en-US&gl=US&ceid=US%3Aen"

        # time.sleep(0.01)
        driver.get(url)

        soup = BeautifulSoup(driver.page_source, 'html.parser')
        allData = soup.find_all("article", {"class": "IFHyqb DeXSAC"})

        for j in range(5):
            try:
                data = allData[j]
                source = data.find('a').get('href')
                source = f"https://news.google.com{source[1:]}"
                name = data.find('a', {'class': 'JtKRv'}).text
                name = name.replace(", ", "")
            
```

```

        daily_register_news[f'Source_{j+1}'].append(source)
        daily_register_news[f'Name_{j+1}'].append(name)
        article = Article(source)
        try:
            article.download()
            article.parse()
            text = article.text
            text = text.replace(", ", "")
            daily_register_news[f'Text_{j+1}'].append(text)
        except Exception as error:
            text = F"ERROR: {error}"
            daily_register_news[f'Text_{j+1}'].append(text)
    except:
        daily_register_news[f'Source_{j+1}'].append(' ')
        daily_register_news[f'Name_{j+1}'].append(' ')
        daily_register_news[f'Text_{j+1}'].append(' ')

    file = fr"C:\Users\aefig\OneDrive\Escritorio\Tec\07_6to
Trimestre\02_ProyectoIntegrador\NewsRegister\{i_date}.csv"

    df_news = pd.DataFrame(daily_register_news)
    df_news.to_csv(file, index=False)

    date += datetime.timedelta(days=1)

driver.close()

```

Se puede observar que se trata de un código mucho más resumido, ya que Google News considera varias cosas en la búsqueda que Bing News con el API lo hacía de manera distinta.

Además, gracias a la opción de establecer un rango temporal, nos permite poder acceder a noticias más viejas que lo que se podía esperar de Bing News, igualando así la granularidad de lo que se tendrá con los valores de las acciones.

Entendimiento de los Datos

Descripción de los datos

En el proyecto se incluirán los datos de alrededor de 500 activos que forman parte del índice S&P500, estos símbolos están descritos en la siguiente publicación de internet: https://en.wikipedia.org/wiki/List_of_S%26P_500_companies

Utilizando el API de Yahoo! Finance se obtendrán los precios diarios correspondientes a las 500 empresas que forman parte del índice.

```

!pip install yfinance

import yfinance as yf
import pandas as pd
from datetime import datetime, timedelta

```

```

def get_snp500_symbols():

    table=pd.read_html('https://en.wikipedia.org/wiki/List_of_S%26P_500_companies')
    df = table[0]
    symbols = df['Symbol'].tolist()
    return symbols

def download_stock_prices(symbols, start_date, end_date):
    data = {}
    for symbol in symbols:
        try:
            stock = yf.download(symbol, start=start_date, end=end_date)
            if not stock.empty:
                data[symbol] = stock['Close']
        except Exception as e:
            print(f"Error downloading data for {symbol}: {e}")
    return data

if __name__ == "__main__":
    # ultimos tres años
    end_date = datetime.now()
    start_date = end_date - timedelta(days=3*365)

    # S&P500 symbols
    symbols = get_snp500_symbols()

    # descargar stock prices
    stock_data = download_stock_prices(symbols, start_date, end_date)

    # Save data to CSV files
    for symbol, data in stock_data.items():
        data.to_csv(f"{symbol}_prices.csv")

```

El siguiente gráfico describe los retornos diarios de todos los símbolos.

```

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

snp500_index = pd.concat(stock_data.values(), axis=1,
keys=stock_data.keys()).mean(axis=1)

daily_returns = snp500_index.pct_change()

```

```

cumulative_returns = (1 + daily_returns).cumprod()

plt.figure(figsize=(10, 6))
cumulative_returns.plot()
plt.title('Retorno Acumulado del Indice S&P500 ')
plt.xlabel('Fecha')
plt.ylabel('Retorno Acumulado')
plt.grid(True)
plt.show()

total_return = cumulative_returns[-1] - 1
total_trading_days = len(snp500_index)
annualized_return = ((1 + total_return) ** (252 / total_trading_days)) - 1

volatility = np.std(daily_returns)

annual_volatility = volatility * np.sqrt(252)

risk_free_rate = 0.02
daily_rf_rate = (1 + risk_free_rate) ** (1/252) - 1
sharpe_ratio = (np.mean(daily_returns) - daily_rf_rate) / annual_volatility

plt.figure(figsize=(10, 6))
daily_returns.plot(kind='hist', bins=50, alpha=0.6)
plt.title('Histograma de Retorno Diario del Indice S&P500 ')
plt.xlabel('Retorno Diario')
plt.ylabel('Frecuencia')
plt.grid(True)
plt.show()

print(f"Retorno Anualizado: {annualized_return:.4f}")
print(f"Volatilidad Anualizada: {annual_volatility:.4f}")
print(f"Sharpe Ratio: {sharpe_ratio:.4f}")

```

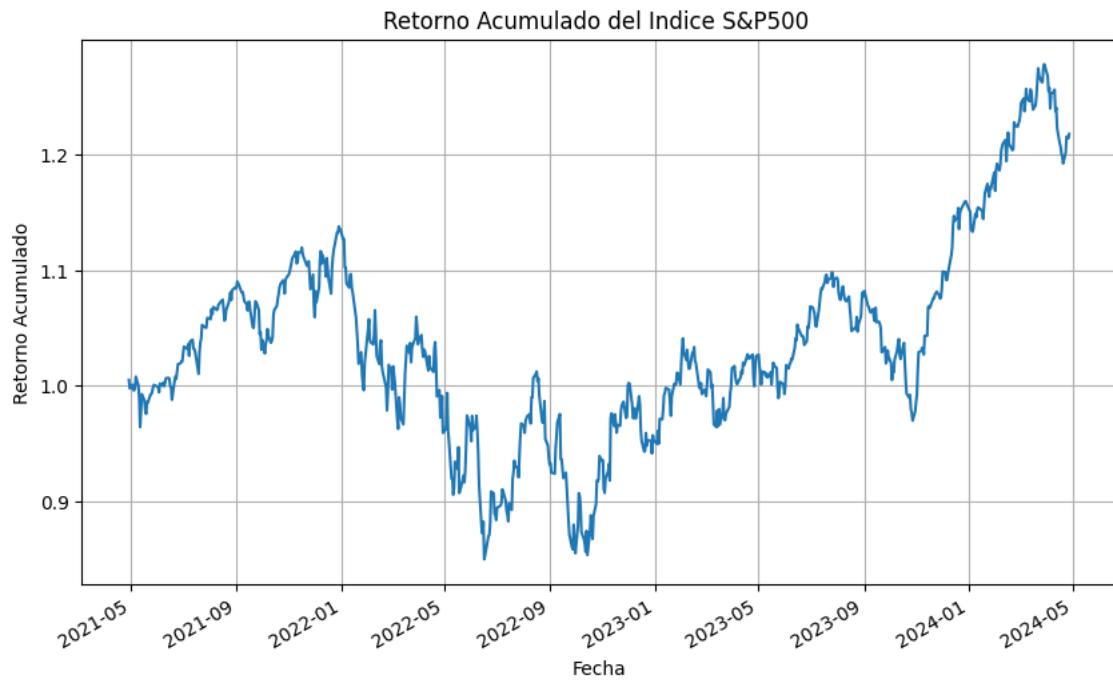


Fig.2: Histograma de los retornos diarios.

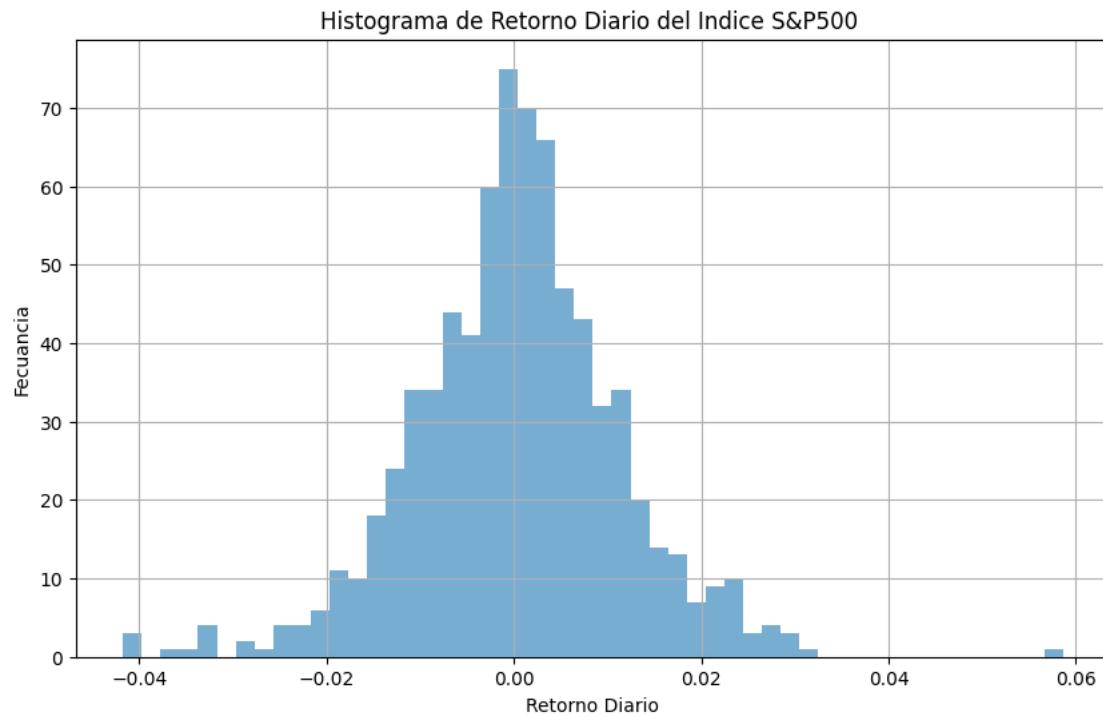


Fig.3: Histórico de retorno de inversión.

El retorno en los últimos tres años de las 500 empresas fue del aproximadamente un 6.8%, con una volatilidad o riesgo anualizado de un 17.5%. El modelo propuesto debe superar este retorno sin aumentar el riesgo del índice.

Identificación de variables

Se pretende tener información diaria (de lunes a viernes) para cada una de las acciones información del valor de la acción a la apertura de la bolsa, el valor más alto, el valor más bajo y el valor al cierre, así como noticias relacionadas con la acción en el último día en caso de existir.

De manera general las columnas que se van a incluir de manera diaria para cada acción incluyen:

- Symbol: Abreviatura con la que se presenta en la bolsa la empresa.
- Name: Nombre comercial con el que se conoce a la empresa.
- Open: Valor a la apertura.
- High: Valor más alto del día.
- Low: Valor más bajo del día.
- Close: Valor al cierre.
- Status_Stock: Estado de actualización de valores en base de datos en el día.
- Source_1,2,3: Fuente de la noticia.
- Name_1,2,3: Título de la noticia
- Description_1,2,3: Descripción de la noticia.
- Status_News: Estado de actualización de noticias en base de datos en el día.
- Retorno diario compuesto: se define como el cambio de precio diario de una acción.
- Riesgo: es la volatilidad de una acción se calcula por medio de la desviación estándar asociados a los retornos diarios.
- Sharpe Ratio: Retorno / Riesgo

Para el registro diario se trabajó en un script de Python en el cual se obtiene de manera independiente un Data Frame de las acciones y otro de las noticias. Posteriormente se realiza una unión de ambas estructuras y se guarda en un CSV para así poder formar el histórico de las acciones a través del tiempo.

En el alcance actual se trabaja para nutrir la base de datos con noticias previas. De lo contrario, se estaría trabajando con un mes de información y se dejaría el script de Python para la actualización diaria de noticias y acciones como legado para nutrir cada vez con más información para el entrenamiento del modelo.

| A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q |
|----------|---------------------------------|-----------|-----------|-----------|-----------|--------------|------------------|--|---------------------------|-----------------------------|---|-----------------------------|----------------------------|-----------------|---------------|-------------|
| Symbol | Name | Open | High | Low | Close | Status Stock | Source_1 | Name_1 | Description_1 | Source_2 | Name_2 | Description_2 | Source_3 | Name_3 | Description_3 | Status News |
| 1 MMM | 3M | 91.589996 | 92 | 90.65 | 91.410003 | OK | YAHOO!Fi | Here's How Weakness in the advertising | MMI Is Bad The democratic | Zacks.com | 3M (MMM) | In the latest tra | | OK | | OK |
| 2 AOS | A. O. Smith | 83.9 | 83.910003 | 80.639995 | 82.819999 | OK | YAHOO!Fi | Decoding A.O. Smith Cor | Zacks.com | A. O. Smith Cor | Zacks.com | Why A.O. S For new and old | | | | OK |
| 4 ABT | Abbott | 106.65000 | 107.45999 | 106.13999 | 106.86800 | OK | Newsweek | Greg Abbott | The Houston CI USA Today | Marla Adas | Marla Adams, The Cincinnati Reds, Andi Left-hander AB | | | | | OK |
| 5 ABBV | AbbVie | 167.66000 | 169.28999 | 165.57000 | 167.28999 | OK | Reuters | Drugmake AbbVie raised | I Bloomberg | AbbVie Inc. | lifts The Motley Better Divi | Founded in 19 | | | | OK |
| 6 ACN | Accenture | 309.19000 | 310.24 | 305.35000 | 309 | OK | YAHOO!Fi | Shareholders want to fir | YAHOO!Fi | Meet the à | We develop our business | TCS, Infosys Earnings seaso | | | | OK |
| 7 ADBE | Adobe Inc. | 468.41000 | 474.5 | 465.77999 | 473.44000 | OK | Business Insider | New Buy R Wells Fargo | an Reuters | Adobe to b Adobe said on | on Fox Business | Adobe's new Ad | | | | OK |
| 8 AMD | Advanced Micro Devices | 149.14999 | 155.13999 | 145.75 | 153.75999 | OK | The Motley | 3 Reasons Advanced Micr | Seeking Alpha | Advanced Micr | The Motley is Advance | Chipmaker Adv | | | | OK |
| 9 AES | AES Corporation | 17.18000 | 17.36000 | 17.049995 | 17.17 | OK | Seeking Alpha | Learn about AE WITV | AES custo | Indianapolis Mi | WXIN-TV | Whistleblow | | | | OK |
| 10 AFL | Aflac | 84.089996 | 84.339996 | 83.22000 | 83.73 | OK | Forbes | President | How a Little Le | Seeking Alpha | Aflac recently | CSR Witr | Aflac U.S. I imagine being | | | OK |
| 11 A | Agilent Technologies | 137.05999 | 137.21000 | 134.11999 | 136.36999 | OK | Benzinga | c Here's How Agilent Techno | YAHOO!Fi | Declining | Agilent Techno | USA Today | S&P 500 (S & P 500 | | | OK |
| 12 APD | Air Products and Chemicals | 234.47999 | 235.97999 | 233.47000 | 235.08000 | OK | YAHOO!Fi | Armed sus The Austin Poli | WSB Atlanta | Man shot it around | 11:10 p | WSB-TV | Suspects Atlanta Police | | | OK |
| 13 ABNB | Airbnb | 161.50999 | 163.72999 | 159.5 | 163.00999 | OK | Gizmodo | 9 Disturbin | Airbnb announ | Business I See inside | The 56-year-old | The Motley | I Want to B Airbnb rentals | | | OK |
| 14 AKAM | Akamai | 101.62000 | 102.16000 | 100.29000 | 101.79000 | OK | techzine | Name S Noname Secur | Zacks.com | Akamai (AI Akamai Technic | Business is RBC Capit | RBC Capital an | | | | OK |
| 15 ALB | Albemarle Corporation | 113.52999 | 115.52999 | 111.05000 | 114.98000 | OK | Barron's | EV Woes C When the EV | b Mena FN | Multi-Billio Melco Progress | Seeking Alpha | My Optimis | Weak lithium pi | | | OK |
| 16 ARE | Alexandria Real Estate Equities | 117.45999 | 118.23999 | 115.46999 | 117.30000 | OK | Seeking Alpha | Alexandria Rea | Seeking Alpha | Alexandria Rea | YAHOO!Fi | Alexandria Alexandria | Rea | | | OK |
| 17 ALGN | Align Technology | 325 | 327.49 | 297.27999 | 310.5 | OK | YAHOO!Fi | Align Tech Align Technolo | YAHOO!Fi | Align Tech Q1 2024 | Earnin | Reuters | Align Tech Align Technolo | | | OK |
| 18 ALLE | Allegion | 126.54000 | 127.66999 | 123.16000 | 124.87000 | OK | Yahoo Fin | Allegion pl | Last week saw | YAHOO!Fi | Align Allegion (A | Q1 revenues | What's in Allegion plc | | | OK |
| 19 LNT | Alliant Energy | 50.26 | 50.529998 | 49.70000 | 50.22999 | OK | The Gazette | Google dat As Google cons | WKOW | More than | Over three hun | Madison.c Sheriff | Ex Law enforcement | | | OK |
| 20 ALL | Allstate | 172.30000 | 173.80000 | 171.28999 | 172.33999 | OK | YAHOO!Fi | Allstate sa Amid the ongoi | Artemis | Allstate to US insurer | Allstate Chicago Tr | Logistics C Post-pandemic | | | | OK |
| 21 GOOGL | Alphabet Inc. A (Class A) | 151.33000 | 156.49000 | 150.86999 | 156 | OK | TheStreet | Analysts ur JPMorgan's Do | Inc | With I Sen On Thursday, | C Forbes | Google | Intr Google launche | | | OK |
| 22 GOOG | Alphabet Inc. A (Class C) | 153.36000 | 158.27999 | 152.76800 | 157.94999 | OK | TheStreet | Analysts ur JPMorgan's Do | Forbes | Google | Google Intr | Google launche | | | | OK |
| 23 MO | Altria | 43.25 | 43.650001 | 42.76 | 43.540000 | OK | The Sun | Mo Salah& | Mo Salah&c;s | YAHOO!Fi | MO House | The Missouri H | USA Today Edge rushe | The Miami Dol | | OK |
| 24 AMZN | Amazon | 169.67999 | 173.91999 | 166.32000 | 173.66999 | OK | TheStreet | Jeff Bezos | Bezos has neve | NBC News | Missing Ut a cat named Gi | Forbes | Amazon Pr | The best new sl | | OK |
| 25 AMCR | Amcor | 9.020004 | 9.100003 | 8.930003 | 8.9499998 | OK | YAHOO!Fi | Amcor unv | The stock bottle | Broadway | The Fox Cit | The Fox Cities F Nasdaq | Will Declin | The estimate hi | | OK |

Fig.4: Información diaria para el análisis de las acciones del S&P500.

Análisis exploratorio de datos

Valores Faltantes:

- Algunos nombres o símbolos de acciones tenían valores faltantes debido a que el API reconoce otro tipo de caracteres. El incidente fue identificado y corregido.

```
symbols = df['Symbol'].str.replace('.',' -').values.tolist()
```

- Por la naturaleza del mercado, no todas las empresas tienen asociadas noticias, por lo tanto no se podría hacer un análisis de sentimiento para todo el mercado

Estadísticas resumidas del Conjunto de Datos

- Retorno diario de inversión de cada portafolio y del mercado
- Riesgo o desviación estándar
- Rendimientos diarios promedio
- Riesgo diario promedio
- Covarianza, varianza y desviación estándar
- Razón de rendimiento sobre riesgo

```
def calculate_portfolio_return(self):
    self.portfolio_return = np.dot(self.avg_returns.mean(),
    self.weights)
    #print("return")
    print(self.portfolio_return)
    #print(" end return")
    return self.portfolio_return

def calculate_sharpe_ratio(self):
    avg_returns = self.portfolio_return
    std_dev = self.calculate_risk_ratio()
```

```

    portfolio_return = self.calculate_portfolio_return()
    risk_free_rate = 0.02
    sharpe_ratio = (portfolio_return - risk_free_rate) /
std_dev.mean()
        return sharpe_ratio

def calculate_risk_ratio(self):
    cov_matrix = self.avg_returns.cov()

    weights = np.array(self.weights)
    portfolio_variance = np.dot(weights.T, np.dot(cov_matrix,
weights))
    portfolio_risk = np.sqrt(portfolio_variance)

    return portfolio_risk

```

Valores Atípicos

Los valores atípicos en los precios son parte de la naturaleza del negocio, se estudiaron por medio de comparar el comportamiento de empresas altamente volátiles con otros grupos, inclusive ya se hicieron algunos intentos de apoyar la toma de decisiones por medio de Inteligencia artificial implementando el algoritmo de k medias para encontrar 10 acciones que generen un mejor retorno con un menor riesgo.

A continuación, se incluyen los resultados:

Los marcados con la X son candidatos a ser estudiados como los activos más óptimos, dado que han generado mayor rentabilidad con riesgo bajo.

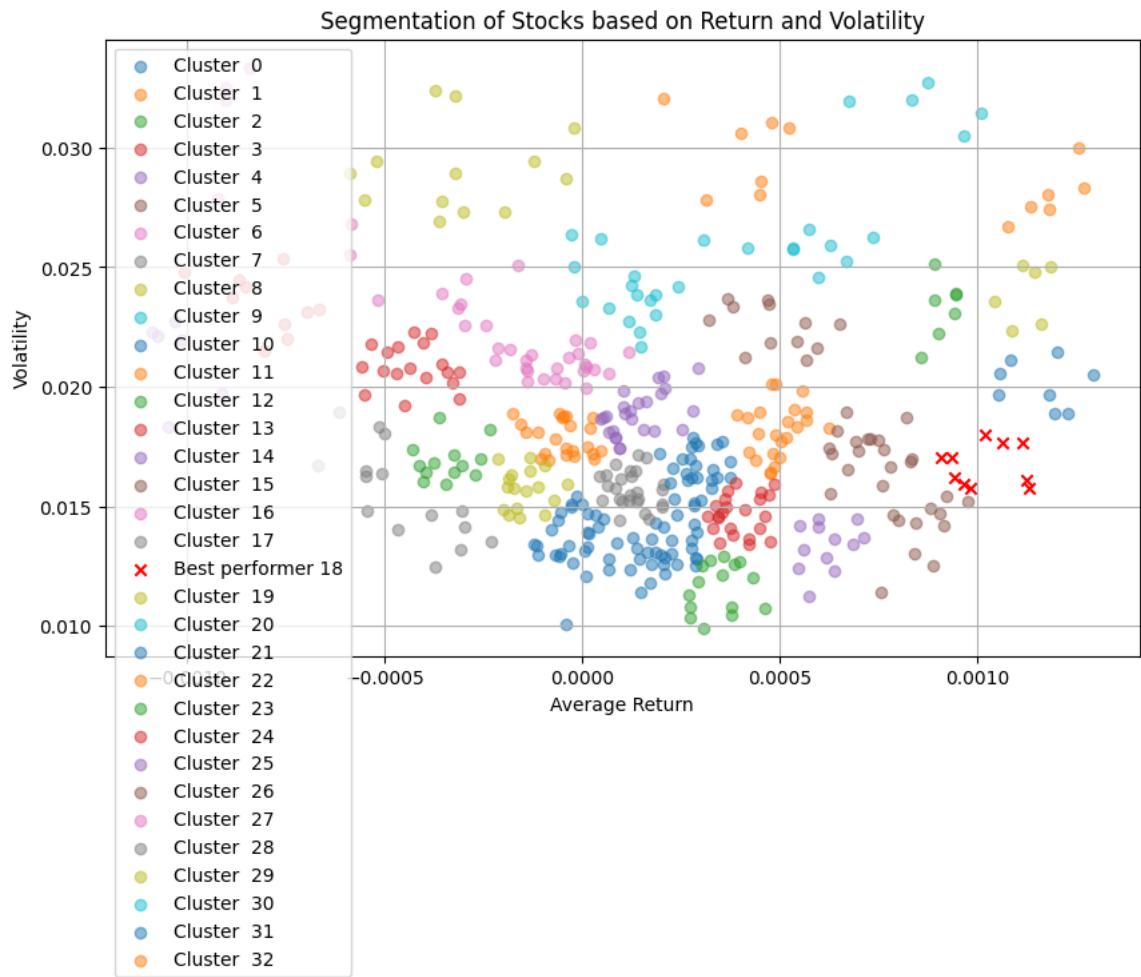


Fig.5: Gráfico de la volatilidad vs el retorno promedio.

```

import pandas as pd
from sklearn.cluster import KMeans
from sklearn.preprocessing import StandardScaler

# Assuming normal_stocks_df contains historical returns for each stock
# Calculate the average return and standard deviation of returns for
each stock
average_returns = normal_stocks_df.mean(axis=0)
volatility = normal_stocks_df.std(axis=0)

# Combine average returns and volatility into one DataFrame
features = pd.concat([average_returns, volatility], axis=1)
features.columns = ['Average Return', 'Volatility']

# Calculate return/risk ratio
features['Return/Risk Ratio'] = features['Average Return'] /
features['Volatility']

# Standardize the data
scaler = StandardScaler()

```

```

scaled_features = scaler.fit_transform(features)

i=11
j=0
while i>10:
    # Perform K-means clustering
    num_clusters = 10+j # You can adjust this parameter
    kmeans = KMeans(n_clusters=num_clusters, random_state=42)
    cluster_labels = kmeans.fit_predict(scaled_features)

    # Add cluster labels to the DataFrame
    features['Cluster'] = cluster_labels

    # Identify the cluster with the highest return/risk ratio
    cluster_with_highest_return_risk_ratio =
    features.groupby('Cluster')['Return/Risk Ratio'].max().idxmax()

    # Select the stocks from that cluster
    selected_stocks = normal_stocks_df.columns[features['Cluster'] ==
cluster_with_highest_return_risk_ratio].tolist()
    i=len(selected_stocks)
    j=j+1

print("Selected Stocks with Highest Return/Risk Ratio:")
print(selected_stocks)

```

```

import matplotlib.pyplot as plt

# Plot the segmentation chart
plt.figure(figsize=(10, 6))
for cluster in range(num_clusters):
    cluster_data = features[features['Cluster'] == cluster]
    if cluster == cluster_with_highest_return_risk_ratio: # Compare
with the index of the series
        plt.scatter(cluster_data['Average Return'],
cluster_data['Volatility'], label=f'Best performer {cluster}', color='red', marker='x')
    else:
        plt.scatter(cluster_data['Average Return'],
cluster_data['Volatility'], label=f'Cluster {cluster}', alpha=0.5)
plt.xlabel('Average Return')
plt.ylabel('Volatility')
plt.title('Segmentation of Stocks based on Return and Volatility')
plt.legend()
plt.grid(True)
plt.show()

```

Otras observaciones con respecto al análisis de datos

Los valores atípicos con los que están asociados a las mejores ganancias o peores pérdidas

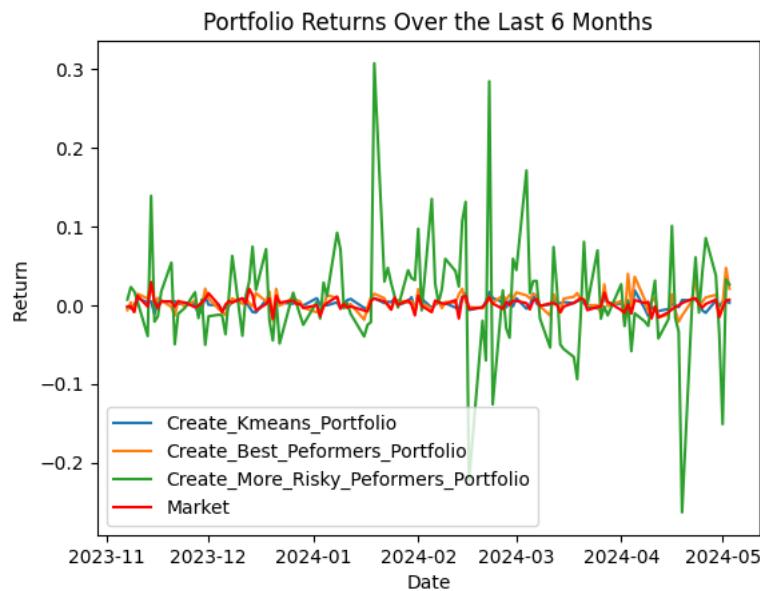


Fig.6: Gráfico del retorno del portafolio en los últimos 6 meses.

Cada línea representa un portafolio de inversión de 10 activos. La línea verde representa el portafolio con mayor riesgo, y la azul un portafolio sugerido por medio de aprendizaje no supervisado.

La siguiente gráfica es muestra la dispersión de los tres portafolios creados con respecto a cada uno de los activos del mercado.

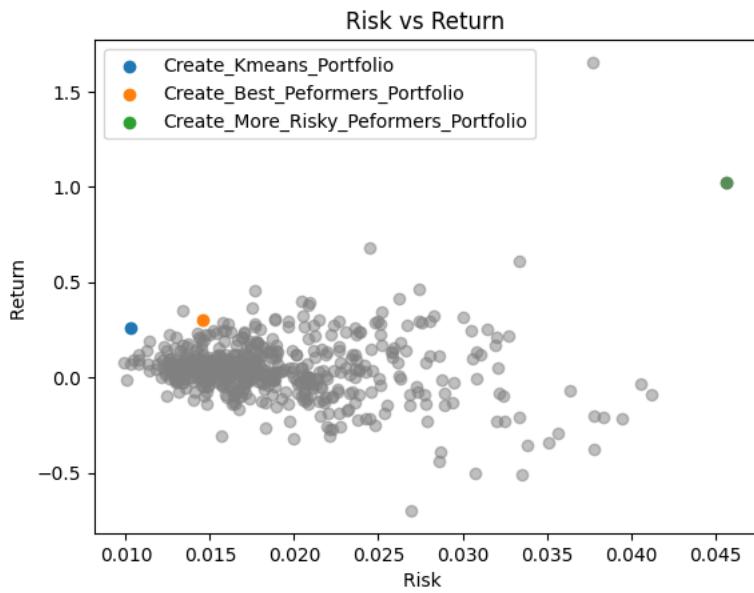


Fig.7: Obtención de agrupamientos con base en el retorno vs el riesgo.

Es muy interesante que el retorno promedio diario del portafolio por medio de aprendizaje no supervisado ofrece un rendimiento diario alto con una volatilidad que tiende a cero.

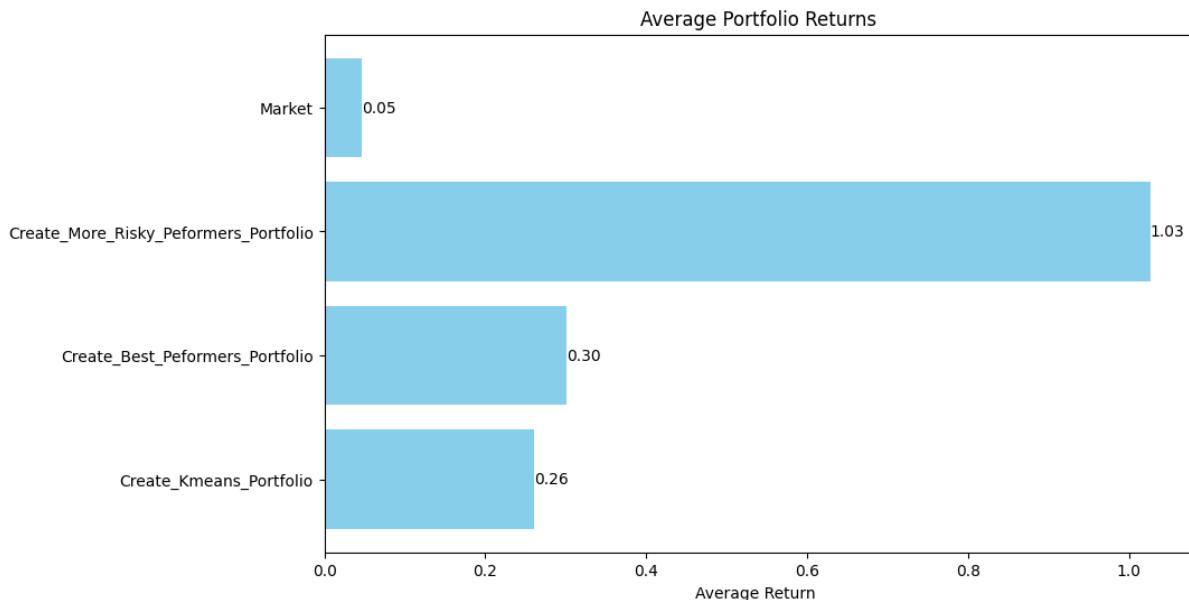


Fig.8: Retorno promedio por los grupo presentado en la gráfica anterior.

Los rendimientos que se presentan en la gráfica anterior son rendimientos anualizados, existe una posibilidad de poder generar un 26% de rendimiento sobre las inversiones.

En las siguiente gráfica se muestran las acciones seleccionadas por medio de analítica descriptiva y aprendizaje no supervisado o segmentación.

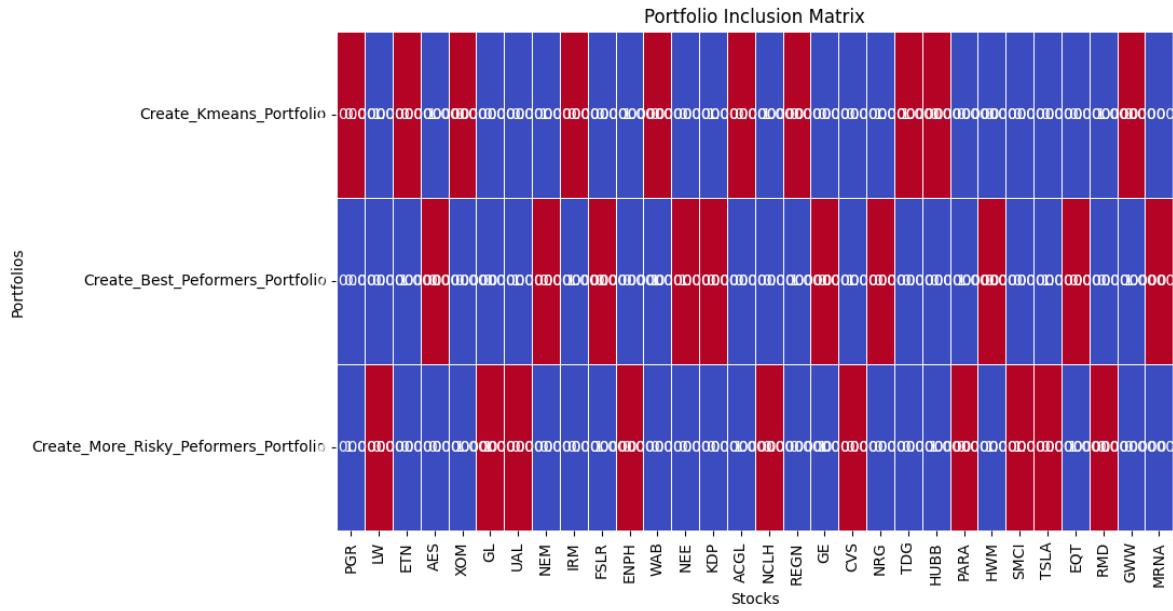


Fig.9: Representación de los portafolios en función de las acciones que incluyen.

El best portfolio es básicamente la selección de los activos más rentable durante los últimos tres años.

El more risky es el conjunto de activos con una mayor desviación estándar.

El portafolio seleccionado por medio de aprendizaje no supervisado es el que se realizó por medio de k medias

Registro de noticias

Dentro del registro de noticias se pudo observar que este era acumulativo en lugar de iniciar nuevamente de manera diaria. Eso es algo que se corregirá y trabajará en la semana.

```
[6]: for i in range(1,5):
    file = fr"C:\Users\aefig\OneDrive\Escritorio\Tec\07_6to Trimestre\02_ProyectoIntegrador\NewsRegister\2021-01-0(i).csv"
    df_news = pd.read_csv(file, encoding='utf8')

    shape = df_news.shape
    print(shape)

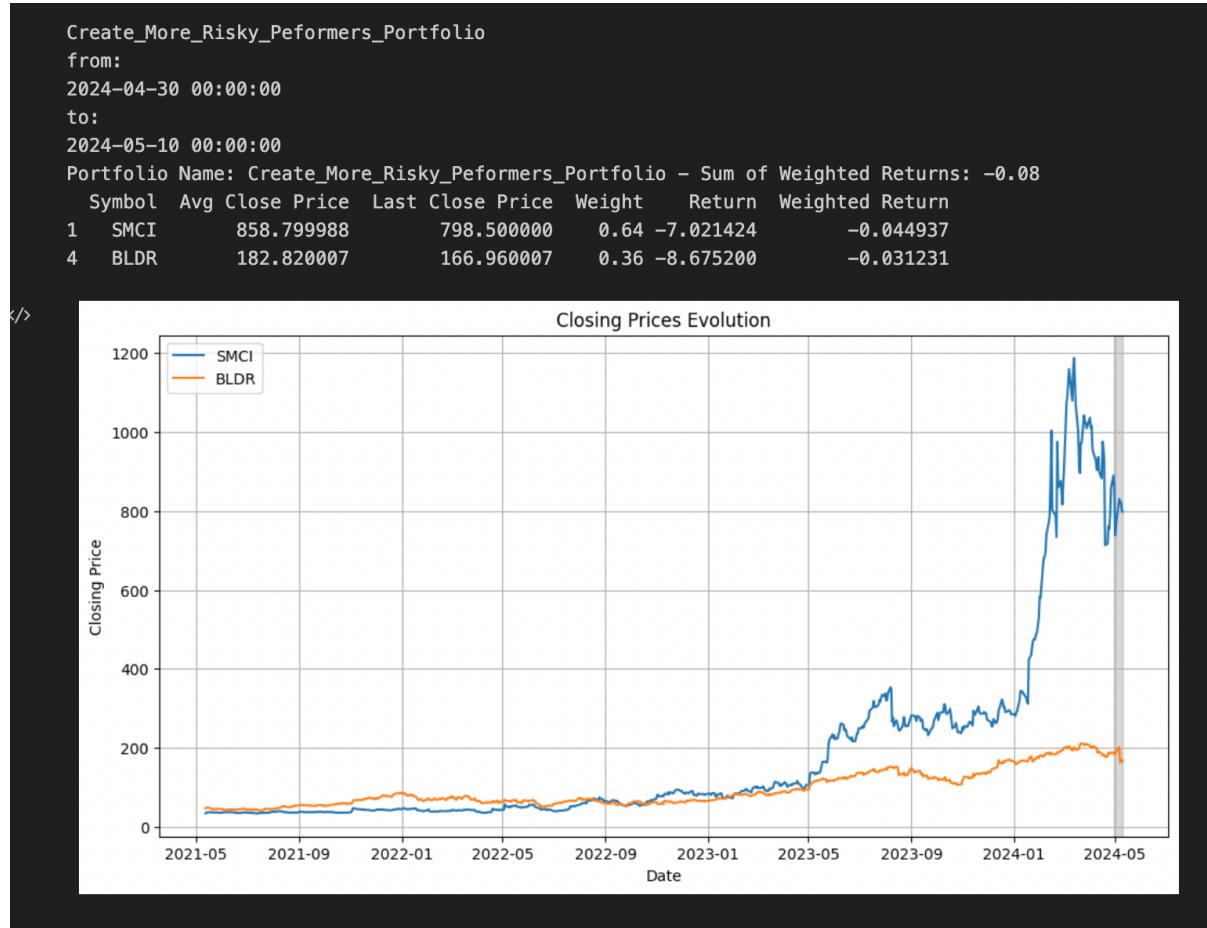
(503, 17)
(1006, 17)
(1509, 17)
(2012, 17)
```

Fig.10: Análisis del tamaño de los registros de noticias.

Sin embargo, se puede proceder con un análisis de las noticias de esta manera y hacer un análisis exploratorio para análisis de sentimiento.

Ingeniería de Características

Para evaluar el desempeño de los modelos y portafolios de inversión se seleccionaron las características que se muestran a continuación:



```
data['Avg Close Price'].append(avg_close_price)
data['Last Close Price'].append(last_close_price)
data['Weight'].append(weight)
data['Return'].append(pct_changes[symbol])
data['Weighted Return'].append(weighted_return)
```

Se utilizan las variables de retorno promedio diario y volatilidad para determinar los pesos más óptimos para el portafolio de inversión.

```
def efficient_frontier(self, num_portfolios=10000):
    returns_data=self.avg_returns
    returns_mean = returns_data.mean()
    returns_cov = returns_data.cov()

    portfolio_returns = []
    portfolio_risks = []

    for _ in range(num_portfolios):
```

```
weights = np.random.random(len(self.symbols))
weights /= np.sum(weights) # Normalize weights to ensure they sum up
to 1

portfolio_return = np.dot(returns_mean, weights)
portfolio_risk = np.sqrt(np.dot(weights.T, np.dot(returns_cov,
weights)))

portfolio_returns.append(portfolio_return)
portfolio_risks.append(portfolio_risk)

additional_return = self.calculate_portfolio_return()
additional_risk = self.calculate_risk_ratio()

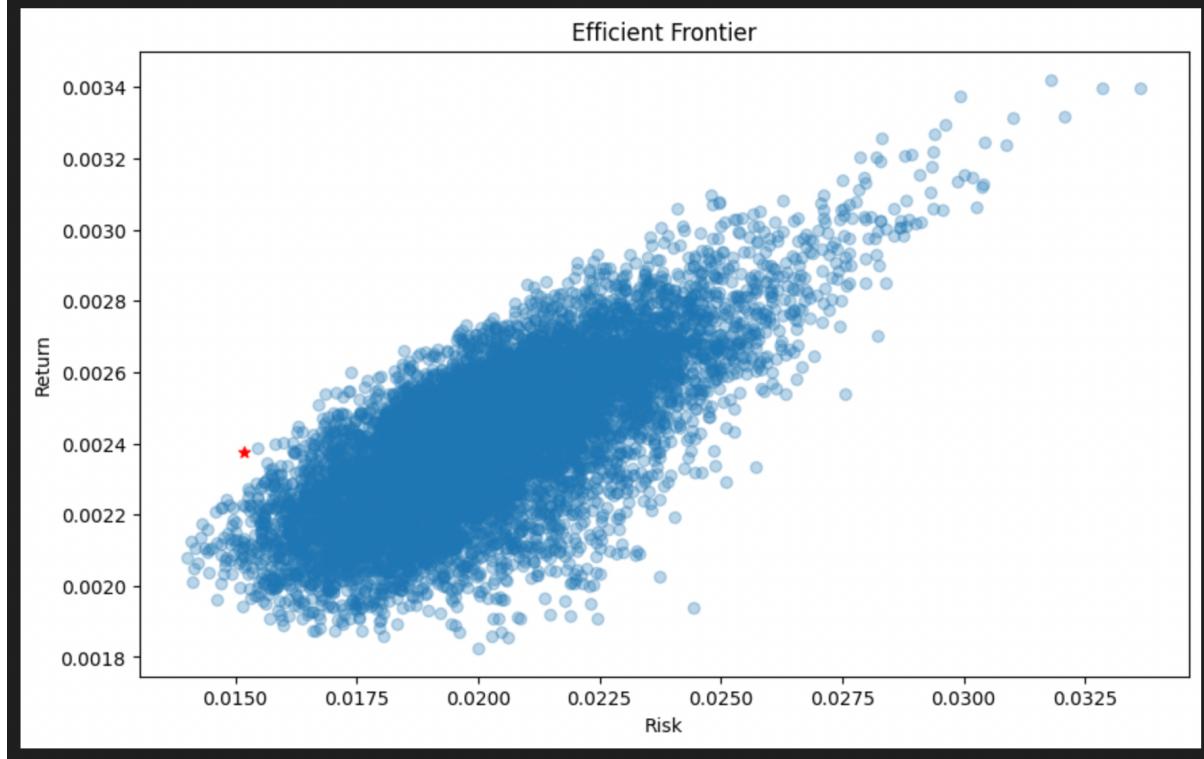
plt.figure(figsize=(10, 6))
plt.scatter(portfolio_risks, portfolio_returns, alpha=0.3)
plt.scatter(additional_risk, additional_return, color='red', marker='*',
label='Additional Point')
plt.title('Efficient Frontier')
plt.xlabel('Risk')
plt.ylabel('Return')
plt.show()
```

Estos son algunos de los resultados

```

Portfolio 1 weighted return: 59.88834865713352
Create_Kmeans_Portfolio
Portfolio Name: Create_Kmeans_Portfolio
Average Return Rate: 0.24%
Sharpe Ratio: -1.16
Weights per Symbol:
ANET: 3.00%
LLY: 44.00%
NVDA: 1.00%
SMCI: 12.00%
VST: 41.00%

```



El portafolio marcado con la estrella roja fue el seleccionado para optimizar el riesgo y el beneficio por medio de la maximización de la razón de Sharpe

```

def calculate_portfolio_return(self):

    self.portfolio_return = np.dot(self.avg_returns.mean(), self.weights)
    #print("return")
    #print(self.portfolio_return)
    #print(" end return")
    return self.portfolio_return

def calculate_sharpe_ratio(self):
    avg_returns = self.portfolio_return
    std_dev = self.calculate_risk_ratio()
    portfolio_return = self.calculate_portfolio_return()
    risk_free_rate = 0.02
    sharpe_ratio = (portfolio_return - risk_free_rate) / std_dev.mean()
    return sharpe_ratio

def calculate_risk_ratio(self):
    cov_matrix = self.avg_returns.cov()

```

```

weights = np.array(self.weights)
portfolio_variance = np.dot(weights.T, np.dot(cov_matrix, weights))
portfolio_risk = np.sqrt(portfolio_variance)

return portfolio_risk

```

Para calcular el riesgo del portafolio y en la selección de pesos para el portafolio óptimo se utilizó una matriz de covarianza para calcular el riesgo del portafolio combinado. Los pesos se maximizaron por medio de la utilización de un algoritmo genético.

```

def assign_weights_markowitz(self, population_size=50, num_generations=100):
    returns_data=self.avg_returns
    num_assets = len(self.symbols)
    creator.create("FitnessMax", base.Fitness, weights=(1.0,))
    creator.create("Individual", list, fitness=creator.FitnessMax)
    def evaluate(individual):
        portfolio_return = np.dot(returns_data.mean(), individual)
        portfolio_risk = np.sqrt(np.dot(individual, np.dot(returns_data.cov(),
individual)))
        sharpe_ratio = portfolio_return / portfolio_risk
        return sharpe_ratio,
    toolbox = base.Toolbox()
    def mate(ind1, ind2):
        ind1, ind2 = tools.cxBlend(ind1, ind2, alpha=0.5)
        ind1 = creator.Individual([max(0, w) for w in ind1])
        ind2 = creator.Individual([max(0, w) for w in ind2])
        return ind1, ind2
    def mutate(individual):
        individual, = tools.mutGaussian(individual, mu=0, sigma=0.2, indpb=0.2)
        individual = creator.Individual([max(0, w) for w in individual])
        return individual,
    toolbox.register("attr_float", random.random)
    toolbox.register("individual", tools.initRepeat, creator.Individual,
toolbox.attr_float, n=num_assets)
    toolbox.register("population", tools.initRepeat, list, toolbox.individual)
    #toolbox.register("mate", tools.cxBlend, alpha=0.5)
    #toolbox.register("mutate", tools.mutGaussian, mu=0, sigma=0.2, indpb=0.2)
    toolbox.register("mate", mate)
    toolbox.register("mutate", mutate)
    toolbox.register("select", tools.selTournament, tournsize=3)
    toolbox.register("evaluate", evaluate)

#def enforce_non_negative(*individuals):
#    return [[max(0, w) for w in individual] for individual in individuals]

```

```

#toolbox.decorate("mate", enforce_non_negative)
#toolbox.decorate("mutate", enforce_non_negative)

population = toolbox.population(n=population_size)
algorithms.eaSimple(population, toolbox, cxpb=0.5, mutpb=0.2,
ngen=num_generations, verbose=False)

best_individual = tools.selBest(population, k=1)[0]
total_weight = sum(best_individual)
self.weights = [round(w / total_weight, 2) for w in best_individual]

```

Para la realización de tareas de segmentación se utilizó el StandardScaler.

```

def Create_Kmeans_Portfolio_Second_Best(market_data):
    data_close=market_data.transpose().dropna().transpose()
    returns = np.log(data_close / data_close.shift(1))
    total_returns = returns.sum()
    current_year = pd.Timestamp.now().year
    df_current_year = returns[returns.index.year == current_year]

    normal_stocks = total_returns.index

    normal_stocks_df = returns[normal_stocks]

    average_returns = normal_stocks_df.mean(axis=0)
    volatility = normal_stocks_df.std(axis=0)

    # Combine average returns and volatility into one DataFrame
    features = pd.concat([average_returns, volatility], axis=1)
    features.columns = ['Average Return', 'Volatility']

    # Calculate return/risk ratio
    features['Return/Risk Ratio'] = features['Average Return'] /
    features['Volatility']

    # Standardize the data
    scaler = StandardScaler()
    scaled_features = scaler.fit_transform(features)

    i=11
    j=0
    while i>10:
    # Perform K-means clustering
        num_clusters = 2+j  #
        kmeans = KMeans(n_clusters=num_clusters, random_state=42)
        cluster_labels = kmeans.fit_predict(scaled_features)

```

```

# Add cluster labels to the DataFrame
    features['Cluster'] = cluster_labels
    #cluster_counts = features['Cluster'].value_counts()
    #small_clusters = cluster_counts[cluster_counts < 10].index
    #features = features[~features['Cluster'].isin(small_clusters)]


# Identify the cluster with the highest return/risk ratio
    #print(features.groupby('Cluster')['Return/Risk
Ratio'].max().nlargest(2).index[1])
    cluster_with_highest_return_risk_ratio =
features.groupby('Cluster')['Return/Risk Ratio'].max().idxmax()

    selected_max_stocks = normal_stocks_df.columns[features['Cluster'] ==
cluster_with_highest_return_risk_ratio].tolist()


    cluster_with_second_highest_ratio = features.groupby('Cluster')['Return/Risk
Ratio'].max().nlargest(2).index[1]

# Select the stocks from that cluster
    selected_stocks = normal_stocks_df.columns[features['Cluster'] ==
cluster_with_second_highest_ratio].tolist()


# Select the stocks from that cluster
    #i=len(selected_stocks)

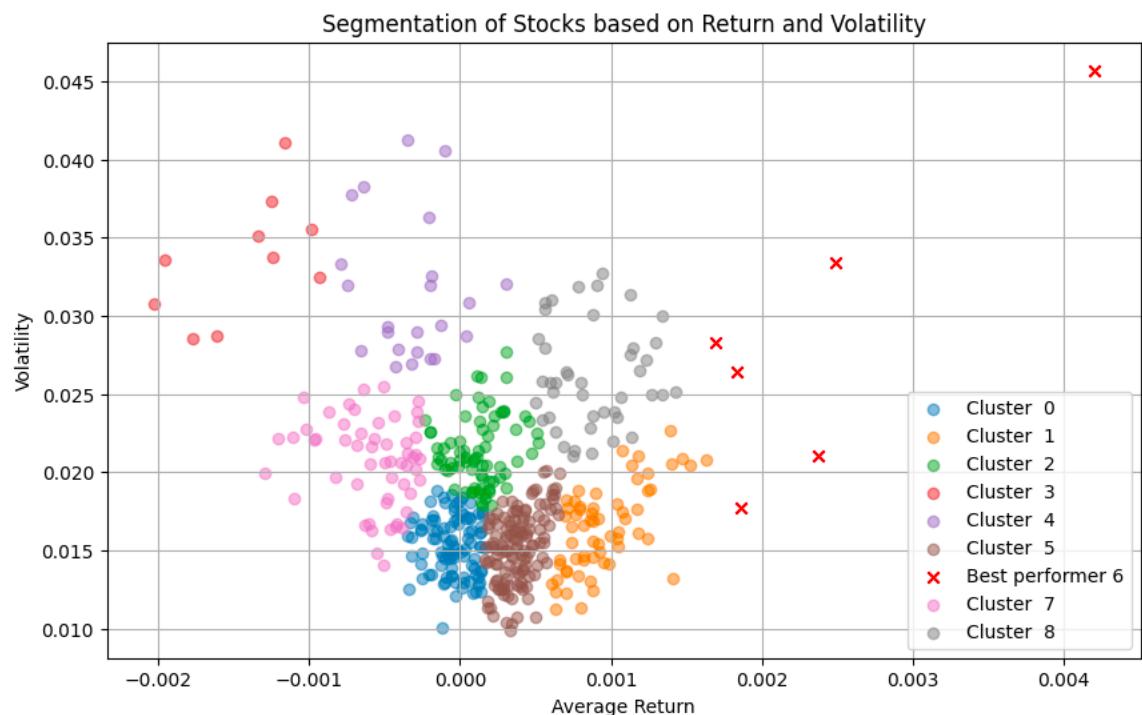
    i=len(selected_max_stocks)
    #if i<10:
    #    j=j-1
    #else:
    j=j+1

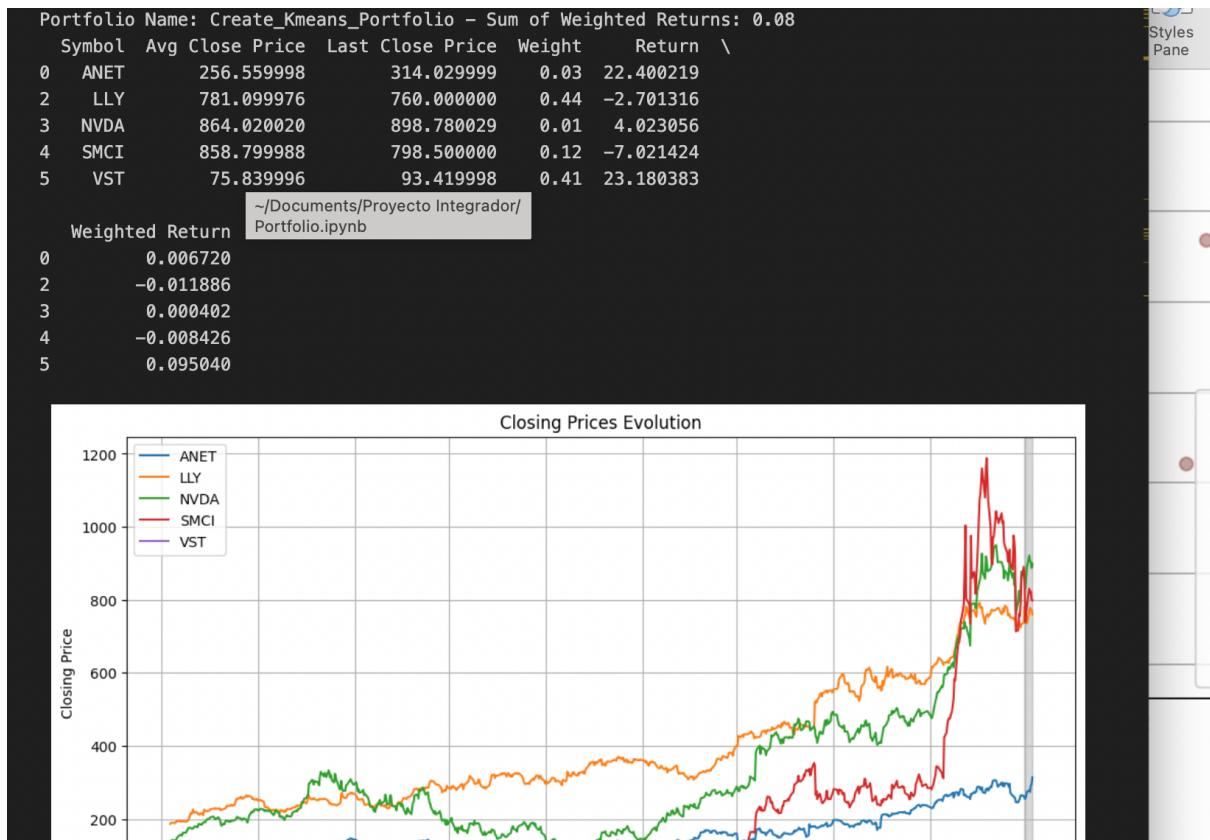
    print("Selected Stocks with Highest Return/Risk Ratio:")
    print (cluster_with_second_highest_ratio)
    retVal = Portfolio(market_data[selected_stocks],selected_stocks)
    plt.figure(figsize=(10, 6))
    for cluster in range(num_clusters):
        cluster_data = features[features['Cluster'] == cluster]
        if cluster == cluster_with_second_highest_ratio: # Compare with the index of
the series
            plt.scatter(cluster_data['Average Return'], cluster_data['Volatility'],
label=f'Second BEST performer {cluster}', color='red', marker='x')
        else:
            plt.scatter(cluster_data['Average Return'], cluster_data['Volatility'],
label=f'Cluster {cluster}', alpha=0.5)
    plt.xlabel('Average Return')
    plt.ylabel('Volatility')
    plt.title('Segmentation of Stocks based on Return and Volatility')
    plt.legend()
    plt.grid(True)
    plt.show()
    return retVal

```

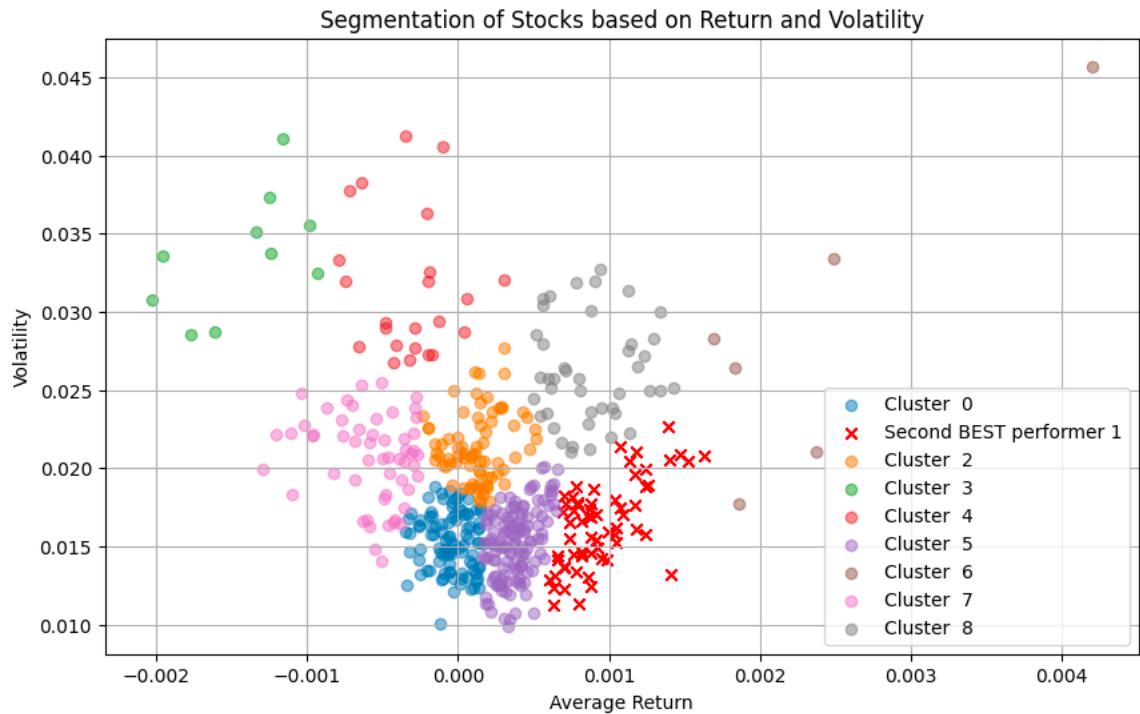
Se utilizaron las variables de retorno, riesgo y razón de retorno sobre riesgo para poder generar una segmentación adecuada. Los valores atípicos juegan un papel importante en este caso de uso, dado que por medio de ellos se pueden obtener mayores beneficios, en tal sentido se estudiarán estos fenómenos. Se seleccionarán los dos mejores segmentos, incluido un segmento de valores atípicos

Segmento de valores atípicos

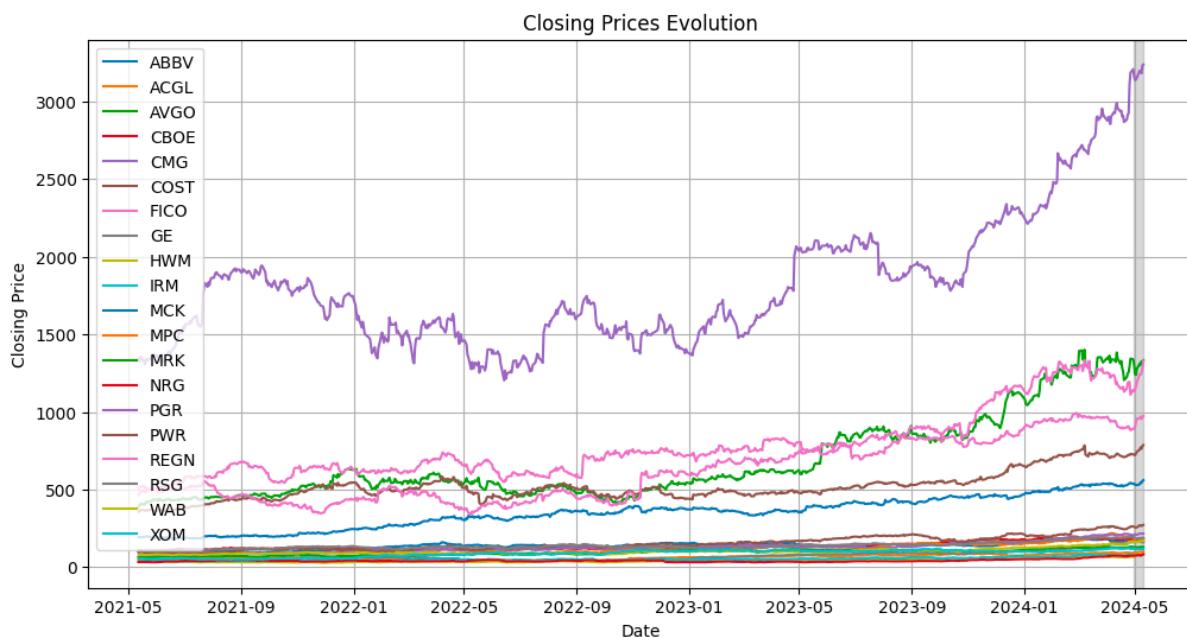




Segmento conservador

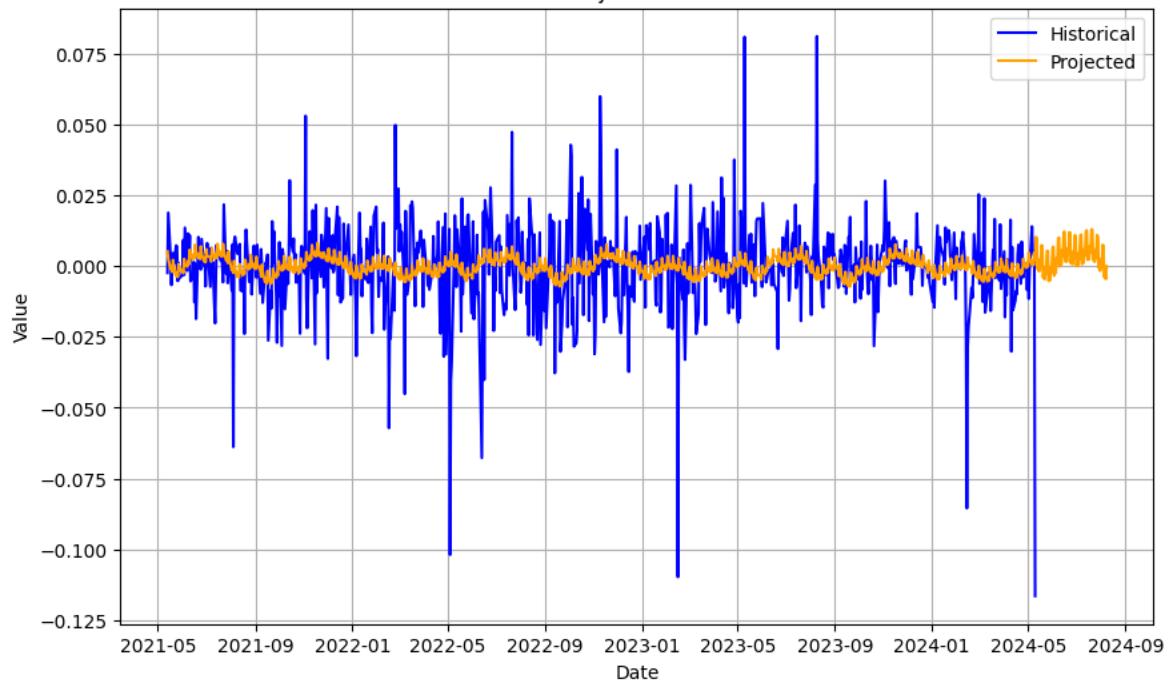


| Portfolio Name: Create_Kmeans_Portfolio_Second_Best - Sum of Weighted Returns: 0.04 | | | | | | |
|---|--------|-----------------|------------------|--------|-----------|---|
| | Symbol | Avg Close Price | Last Close Price | Weight | Return | \ |
| 0 | ABBV | 162.639999 | 160.750000 | 0.01 | -1.162075 | |
| 1 | ACGL | 93.540001 | 100.050003 | 0.04 | 6.959592 | |
| 7 | AVGO | 1300.270020 | 1332.800049 | 0.11 | 2.501790 | |
| 13 | CBOE | 181.149994 | 181.059998 | 0.03 | -0.049681 | |
| 15 | CMG | 3159.600098 | 3239.229980 | 0.03 | 2.520252 | |
| 18 | COST | 722.900024 | 787.190002 | 0.01 | 8.893343 | |
| 23 | FICO | 1133.329956 | 1328.609985 | 0.04 | 17.230642 | |
| 25 | GE | 161.820007 | 163.380005 | 0.01 | 0.964033 | |
| 30 | HWM | 66.750000 | 80.870003 | 0.01 | 21.153562 | |
| 32 | IRM | 77.519997 | 79.769997 | 0.01 | 2.902477 | |
| 35 | MCK | 537.210022 | 559.909973 | 0.32 | 4.225526 | |
| 38 | MPC | 181.720001 | 179.559998 | 0.07 | -1.188644 | |
| 39 | MRK | 129.220001 | 130.059998 | 0.15 | 0.650051 | |
| 42 | NRG | 72.669998 | 83.650002 | 0.03 | 15.109404 | |
| 47 | PGR | 208.250000 | 215.759995 | 0.03 | 3.606240 | |
| 49 | PWR | 258.559998 | 271.480011 | 0.01 | 4.996911 | |
| 50 | REGN | 890.659973 | 973.799988 | 0.01 | 9.334653 | |
| 51 | RSG | 191.699997 | 189.240005 | 0.03 | -1.283251 | |
| ... | | | | | | |
| 50 | | 0.000933 | | | | |
| 51 | | -0.000385 | | | | |
| 57 | | 0.000451 | | | | |
| 61 | | -0.000052 | | | | |

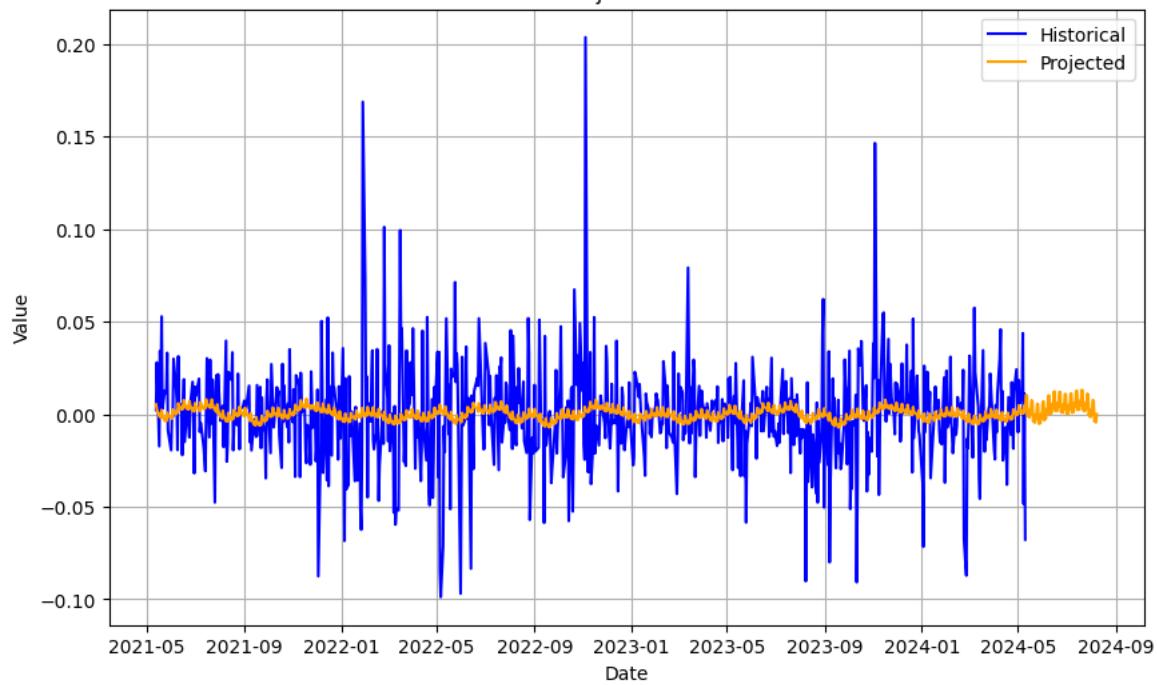


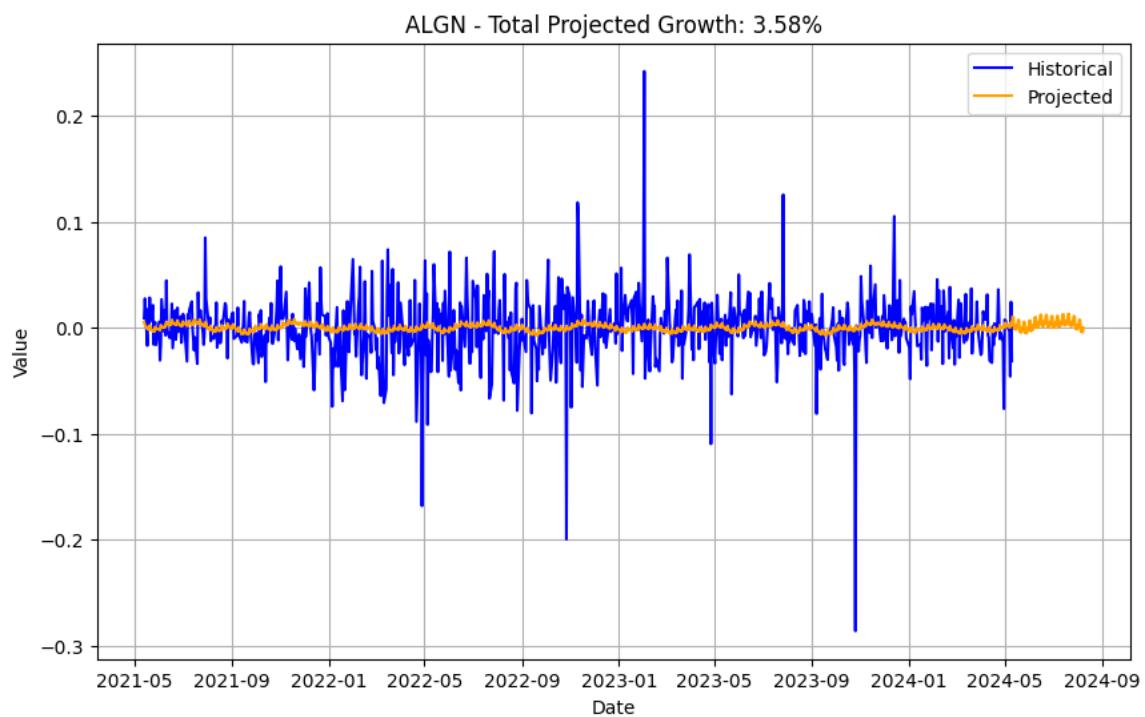
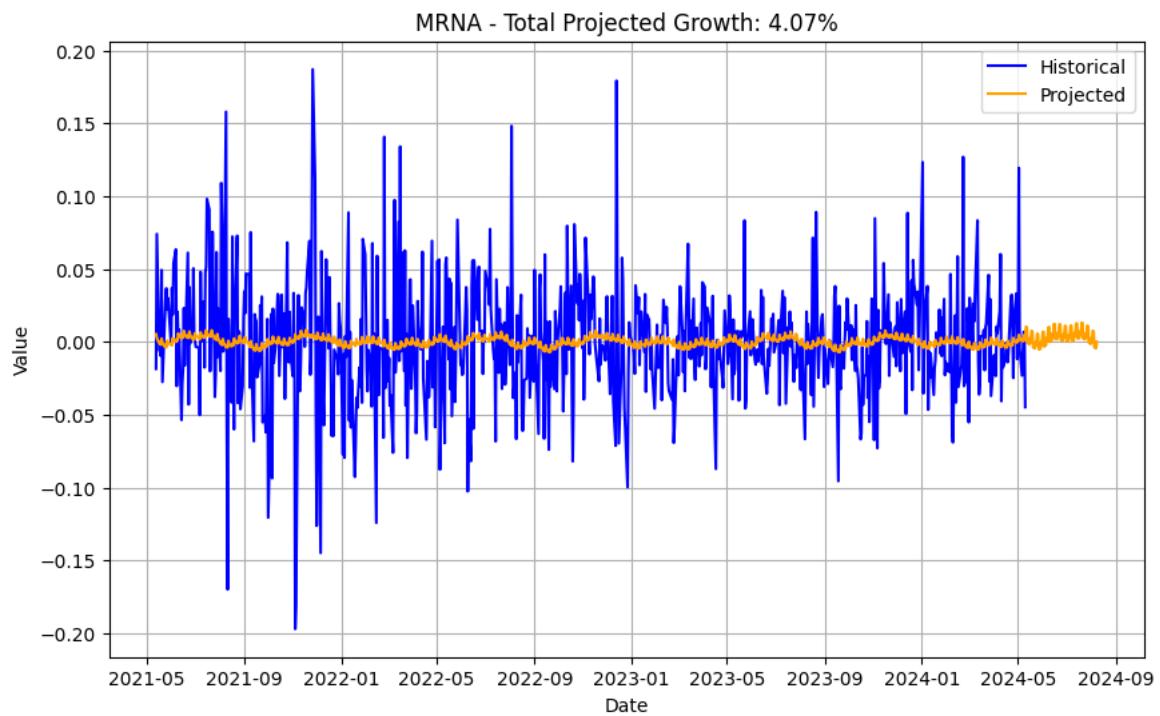
Además de los modelos de correlación entre riesgo y beneficio, se estará probando el modelo Prophet para proyección en series de tiempo. Estos son algunos resultados preliminares de la selección de un portafolio de inversión basado en proyecciones de beneficios:

AKAM - Total Projected Growth: 11.96%



PODD - Total Projected Growth: 6.25%





En entregas futuras se estarán generando modelos de portafolio de inversión con los resultados de los análisis de sentimientos, además de generar portafolios con la mezcla de métodos para probar cuales son los más eficiente, emulando la dinámica de algoritmos genéticos.

Conclusiones: Fase de Preparación de Datos

1. Se concluyó la configuración para la extracción de datos cualitativos, se identificaron datos faltantes, se completaron y se identificaron datos atípicos.
2. Los datos atípicos serán estudiados y evaluados dado que estos pueden resultar en mayores beneficios en la vida real.
3. Las métricas clave para la definición de portafolios óptimos están listas para ser colocadas y refinadas dentro de los modelos.
4. Se incorporó un modelo en series de tiempo ya que los datos se prestan para ese tipo de aplicación.
5. Se estará completando el análisis de sentimientos para poder crear modelos basados en sentimientos provenientes de las noticias asociadas a las empresas que forman parte del S&P 500.

Anexos

Github – Equipo 15

[AEFGa/MNA_ProyectoIntegrador_Equipo15 \(github.com\)](https://github.com/AEFGa/MNA_ProyectoIntegrador_Equipo15)

Script Python

<https://drive.google.com/file/d/1lciy8nmrmK336EYHE3BRIUnAhBcpRhB/view?usp=sharing>

CSV Generado

<https://drive.google.com/file/d/19QOfnopVT4sy-Du4OwbCIlt1TvLQhKHg/view?usp=sharing>

Bibliografía

- Baldridge, R. (2023, 26 de junio) Understanding Modern Portfolio Theory. Forbes. <https://www.forbes.com/advisor/investing/modern-portfolio-theory/>
- Granieri, M. (2023, 13 de septiembre) *Text Mining: Qué es, para qué sirve y principales técnicas.* OBS Business School. <https://www.obsbusiness.school/blog/text-mining-que-es-para-que-sirve-y-principales-tecnicas>
- Tretina, K. (2023, 9 de agosto). *What is the S&P 500? How does it work?* Forbes. <https://www.forbes.com/advisor/investing/what-is-sp-500/>
- Vázquez, I. (2012). Bolsa de Valores “¿Cómo? ¿Por qué? Y ¿Para qué?”. *Tiempo económico*, 7(21), 55-79. <https://tiempoeconomico.azc.uam.mx/wp-content/uploads/2017/07/21te4.pdf>
- Wikipedia (2024). *List of S&P 500 companies.* Wikipedia. https://en.wikipedia.org/wiki/List_of_S%26P_500_companies