

Actualización de la herramienta de visualización de los datos históricos y futuros a corto plazo y mediano plazo del S&P500 y sus principales componentes, que mejore la toma de decisiones oportunas con respecto a los portafolios de inversión del Grupo Stanley

Presentado por:

Gustavo Andrés Parra

Dora Shirley Sánchez

Viviana Vales

Proyecto de grado para optar al título de MAGISTER EN INTELIGENCIA ANALÍTICA DE DATOS PARA LA TOMA DE DECISIONES

Director: Camilo Gómez

UNIVERSIDAD DE LOS ANDES

FACULTAD DE INGENIERÍA INDUSTRIAL

MAESTRÍA EN INTELIGENCIA ANALÍTICA DE DATOS PARA LA TOMA DE

DECISIONES

BOGOTÁ

2023

1. Tratamiento previo de los datos

La base de datos inicial que se construyó para la realización de este proyecto se explicará a continuación:

- Se tuvieron en cuenta el precio de cierre diario ajustado desde el 2012-05-18 hasta el 2023-02-28 de las siguientes acciones que pertenecen al portafolio del grupo Stanley:
 - ✓ SP500
 - ✓ Apple (AAPL)
 - ✓ Microsoft (MSFT)
 - ✓ Amazon (AMZN)
 - ✓ Tesla (TSLA)
 - ✓ Alphabet Class A (GOOGL)
 - ✓ Alphabet Class C (GOOG)
 - ✓ NVIDIA Corporation (NVDA)
 - ✓ Berkshire Hathaway Class B (BRK.B)
 - ✓ Meta (FB), formerly Facebook, Class A (META)
 - ✓ UnitedHealth Group (UNH)

Pero también se tuvieron en cuenta estas dos acciones que están por fuera del portafolio de la empresa:

- ✓ Jhonson y Johnson (JNJ)
- ✓ Procter & Gamble (PG)

Como variables macroeconómicas se añadió a la base de datos:

- ✓ Tasa de desempleo en Estados Unidos (UNRATE)
- ✓ Tasa de interés Banco Central EEUU (FEDFUNDS)
- ✓ Índice de sentimiento del consumidor (UMCSENT)
- ✓ Índice de precios al consumidor (CPIAUCSL)
- ✓ Nominal Broad U.S. Dollar Index (DTWEXBGS)
- ✓ VIX

Las cuatro primeras variables macroeconómicas mencionadas sus datos se encuentran de manera mensual, mientras que el Dollar Index y el VIX de manera diaria. La información de las acciones se extrae de la API de Yahoo Finance, mientras que las macroeconómicas por medio de la API FRED.

- Se calcularon los rendimientos logarítmicos con respecto a los precios de cierre ajustados de las acciones sin estandarizar, por facilidad en el manejo de los datos se multiplicaron por 100.
- Una idea inicial con la cual se trabajó, pero posteriormente descartada fue imputar los días calendario fuera del calendario bursátil, donde en general no hay valores para fines de semana y festivos en Estados Unidos. Imputar estos registros con algún criterio implicaría agregarle al modelo 1040 datos (52 semanas x 10 años x 2 datos por semana), un 38.9% de la información imputada. Por lo cual, se decide trabajar con los datos disponibles.
- Con los rendimientos logarítmicos calculados se realizó suavizamiento de los puntos atípicos por medio de interpolación lineal

El anterior procedimiento fue aplicado para las acciones tanto en el portafolio como fuera de él, el VIX y el dollar index. Sin embargo, la tasa de desempleo, la tasa de interés, el índice de sentimiento del consumidor y el IPC son datos mensuales, y al calcular los rendimientos mensuales logarítmicos y tenerlos en cuenta en el análisis, se hacía necesario imputar para cada día del mes el rendimiento mensual, es decir un mismo dato repetido 30 veces, lo cual generaba

inconvenientes y no mostraba correlación con las demás variables(Ver grafica de correlación en el análisis descriptivo), por lo tanto las variables mensuales no incluidas.

Con todo el proceso descrito anteriormente se diseñó una base de datos con 2.667 filas y 75 columnas para el proceso de experimentación de modelos.

2. Análisis descriptivo de los datos

Para realizar el análisis se tuvo en cuenta el precio de cierre diario ajustado de las acciones del portafolio y dos adicionales (JNJ y PG) desde el 2012-05-18 hasta el 2023-02-28.



El índice Standard & Poor's 500, es uno de los índices bursátiles más importantes de Estados Unidos. Se considera el índice más representativo de la situación real del mercado. Al observar la anterior gráfica, se identifica que en general la tendencia es creciente excepto en el 2020, debido a la pandemia del COVID-19. Con respecto a las acciones dentro y fuera del portafolio:



A continuación se muestra el color que representa cada una de las acciones y el valor de la misma para el último día de nuestro análisis:

Feb, 28, 2023:

AAPL.Adjusted: 147.41

AMZN.Adjusted: 94.23

MSFT.Adjusted: 249.42

TSLA.Adjusted: 205.71

GOOG.Adjusted: 90.3

GOOGL.Adjusted: 90.0

NVDA.Adjusted: 232.12

BRK.B.Adjusted: 305.1

META.Adjusted: 174.94

UNH.Adjusted: 474.25

JNJ.Adjusted: 153.26

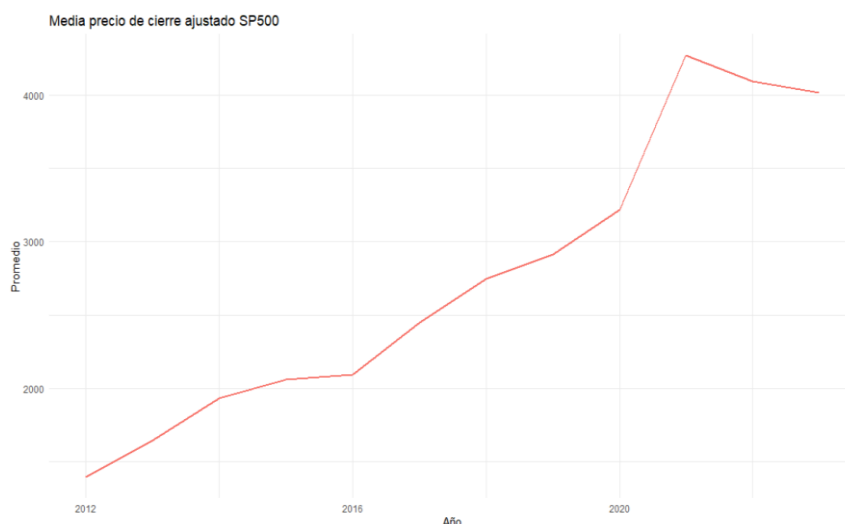
PG.Adjusted: 136.7

Realizando un análisis descriptivo de la información:

	mean <dbl>	sd <dbl>	median <dbl>	trimmed <dbl>	mad <dbl>	min <dbl>	max <dbl>
CSPC.Adjusted	2684.38123	911.762548	2496.65991	2619.42120	817.149686	1278.040039	4796.5601
AAPL.Adjusted	59.33625	49.443288	37.26645	52.47807	25.255499	12.046193	180.6839
AMZN.Adjusted	70.59102	53.416974	50.45650	65.60258	53.308367	10.411000	186.5705
MSFT.Adjusted	113.60767	91.445061	71.30583	102.27288	61.403224	21.689644	339.0756
TSLA.Adjusted	70.40390	100.401707	18.01467	49.90722	8.099938	1.740000	409.9700
GOOG.Adjusted	57.39797	35.622027	48.82850	52.66969	31.223556	13.924059	150.7090
GOOGL.Adjusted	57.58061	35.124550	49.69200	53.02601	31.703918	13.990240	149.8385
NVDA.Adjusted	62.76533	74.192549	37.73744	49.04662	48.976484	2.610562	333.3508
BRK.B.Adjusted	185.18283	65.286760	178.53999	181.04335	61.750281	78.830002	359.5700
META.Adjusted	148.15280	86.404439	143.27000	141.48331	85.086424	17.730000	382.1800
UNH.Adjusted	212.31591	144.220786	183.06285	196.01436	142.958986	43.228561	551.4797
JNJ.Adjusted	110.70730	35.455922	112.40154	110.20529	43.910330	45.895130	181.1088
PG.Adjusted	88.37054	31.760390	73.68686	85.63539	21.534875	42.930496	159.2098

Se observa que en general las acciones del portafolio con mayor promedio el su precio de cierre son UNH, BRK, META y MSTF, y las de menor valor son GOOGL, GOOG y AAPL. Sin embargo, al observar la gráfica se observa que hasta septiembre del año del 2021 las acciones más representativas presentan una tendencia creciente, exceptuando el pico decreciente del mes de marzo, pero a partir de esta fecha empieza a decrecer el precio de cierre de las acciones excepto por UNH donde su comportamiento en el periodo de análisis siempre al alza.

Realizando un análisis más detallado del SP500 con respecto a su precio promedio de cierre por año y mes:



Año <dbl>	MediaSP500 <dbl>
2012	1393.009
2013	1643.799
2014	1931.376
2015	2061.068
2016	2094.651
2017	2449.076
2018	2746.214
2019	2913.356
2020	3217.856
2021	4273.386
2022	4098.515
2023	4018.645

En general y como se mencionó anteriormente presenta una tendencia creciente, aunque desde el año 2014 al 2016 el comportamiento fue muy similar.



Mes <dbl>	MediaSP500 <dbl>
1	2766.058
2	2794.134
3	2659.475
4	2671.193
5	2618.511
6	2588.491
7	2625.866
8	2680.163
9	2672.731
10	2643.805
11	2737.304
12	2766.455

Analizando el comportamiento mensual, el precio promedio de cierre del SP500 oscila entre 2500 y 2800, siendo el mes de junio con el valor más bajo. Sin embargo, la diferencia con los meses de a principio y final de año no es significativa.

Realizando un análisis más detallado entre la relación del SP500 y las variables VIX y dólar index que son diaria, tenemos:

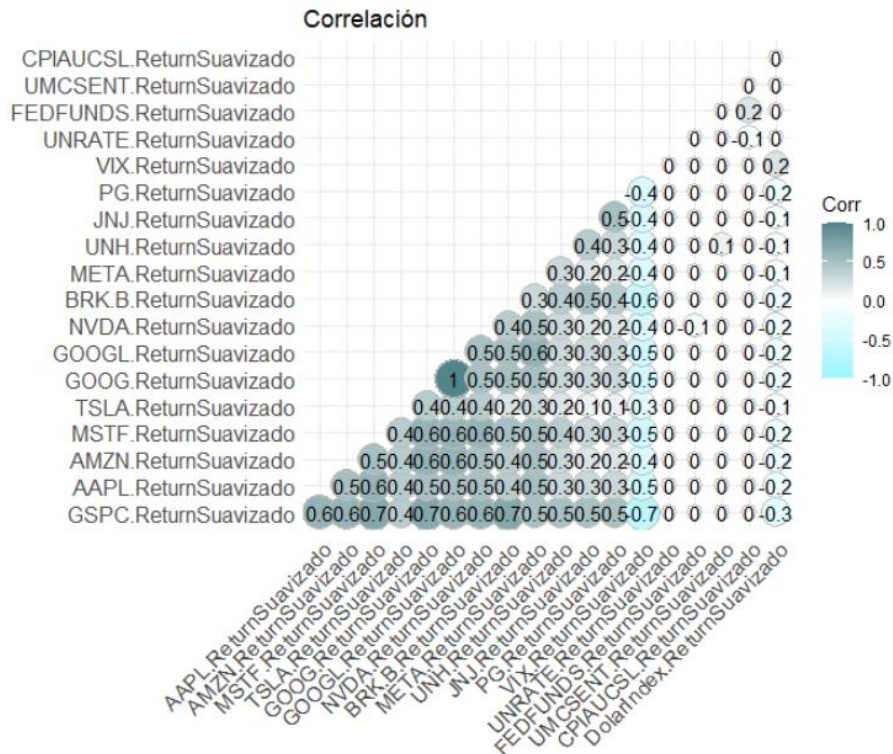
	GSPC.Return Suavizado	VIX.Return Suavizado	DolarIndex.Return Suavizado
count	2667.000000	2667.000000	2667.000000
mean	0.062439	-0.344104	0.008501
std	0.880420	7.081882	0.298783
min	-3.002265	-26.622753	-1.188374
25%	-0.360074	-4.487971	-0.166130
50%	0.063567	-0.749383	-0.001720
75%	0.544763	3.449533	0.181680
max	3.101521	26.760500	1.222749

GSPC: Las variaciones del S&P500 oscilan entre el -3% y el 3.1%, con una media de 0.06% y una mediana cercana a la media. El análisis nos muestra que a partir del primer cuartil tenemos variaciones positivas, lo cual nos indica que las variaciones han sido positivas en un 75% del tiempo. Tan solo en un 25% del tiempo aproximadamente han sido negativas.

VIX: Este indicador mide el nivel de volatilidad esperada del mercado en un futuro cercano. Un alto nivel puede indicar incertidumbre y riesgo. El comportamiento de sus rendimientos nos muestra una media de -0.3 y una mediana de -0.7 lo cual presume que en general los inversionistas no han tenido a lo largo del tiempo una alta preocupación por el riesgo en un poco más del 50% del tiempo. Sin embargo, vemos que el último cuartil nos muestra que un 25% del tiempo (lo que equivale a unos 2.5 años aproximadamente) esta incertidumbre creció a grandes niveles llegando al 26.7

DOLARINDEX: En relación con el valor del dolar, vemos que en el 50% del tiempo éste se ha depreciado en porcentajes inferiores al 0.004% llegando a mínimos del -1.18% y que en el 50% restante ha adquirido fuerza vs las demás monedas contempladas en este índice en donde podemos observar que ha logrado tener comportamientos máximos de 1.22%. Sin embargo, podemos observar que estos rendimientos son pequeños no superando en ningún caso el 1.5%

Al realizar el análisis de correlación



En general se observa una alta correlación entre las variables excepto para las variables macroeconómicas diarias, es decir para la tasa de interés, el IPC, el índice del consumidor y la tasa de desempleo, razón por la cual y como se mencionó anteriormente no fueron tenidas en cuenta en el modelo final.

3. Selección de modelos y pruebas de verificación de los supuestos requeridos para usar los modelos elegidos

- **Modelos vectoriales autorregresivos:** Estos modelos pueden considerarse una extensión de los modelos autorregresivos AR(p).

Se usa un modelo VAR cuando:

- **Cuando las series temporales que se van a modelizar son estacionarias:** Como se va a trabajar con las variaciones logarítmicas inicialmente de las series: S&P 500, índice VIX, tasa efectiva de los fondos federales y el índice del dólar, la prueba de Dickey Fuller afirma que las series son estacionarias como se muestra a continuación:

```
Augmented Dickey-Fuller Test
data: BaseEntrenamiento$GSPC.Return
Dickey-Fuller = -13.996, Lag order = 13, p-value = 0.01
alternative hypothesis: stationary
```

```
Augmented Dickey-Fuller Test
data: BaseEntrenamiento$VIX.Return
Dickey-Fuller = -16.265, Lag order = 13, p-value = 0.01
alternative hypothesis: stationary
```

```
Augmented Dickey-Fuller Test
data: BaseEntrenamiento$DTWEXBGS.Return
Dickey-Fuller = -13.432, Lag order = 13, p-value = 0.01
alternative hypothesis: stationary
```

```
Augmented Dickey-Fuller Test
data: BaseEntrenamiento$DFF.Return
Dickey-Fuller = -11.753, Lag order = 13, p-value = 0.01
alternative hypothesis: stationary
```

- **Contraste de causalidad en el sentido de Granger:** Esta prueba sirve para determinar si los resultados de una variable y sus rezagos sirven para predecir a otra variable, y así poder saber si este modelo VAR es el adecuado a implementar. A continuación los resultados:

Las pruebas de hipótesis son:

H0: La serie de tiempo X no causa en el sentido de Granger la serie de tiempo Y

HA: La serie de tiempo X causa en el sentido de Granger la serie de tiempo Y

- ✓ **Conclusión 1:** Las variaciones logarítmicas de índice del VIX causan en el sentido de Granger las variaciones logarítmicas del SP500 con un rezago.

```
Granger causality test
Model 1: BaseEntrenamiento$GSPC.Return ~ Lags(BaseEntrenamiento$GSPC.Return, 1:1) + Lags(BaseEntrenamiento$VIX.Return, 1:1)
Model 2: BaseEntrenamiento$GSPC.Return ~ Lags(BaseEntrenamiento$GSPC.Return, 1:1)
Res. Df  Df      F      Pr(>F)
1      2678
2      2679 -1 25.036 5.992e-07 ***
---
signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

- ✓ **Conclusión 2:** Las variaciones logarítmicas del índice del dólar causan en el sentido de Granger las variaciones logarítmicas del SP500 con dos rezagos

```
Granger causality test
Model 1: BaseEntrenamiento$GSPC.Return ~ Lags(BaseEntrenamiento$GSPC.Return, 1:2) + Lags(BaseEntrenamiento$DTWEXBGS.Return, 1:2)
Model 2: BaseEntrenamiento$GSPC.Return ~ Lags(BaseEntrenamiento$GSPC.Return, 1:2)
Res. Df  Df      F      Pr(>F)
1      2675
2      2677 -2 6.0024 0.002506 **
---
signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

- ✓ **Conclusión 3:** Las variaciones logarítmicas de la tasa de interés causan en el sentido de Granger las variaciones logarítmicas del SP500 con quince rezagos

```
Granger causality test
Model 1: BaseEntrenamiento$GSPC.Return ~ Lags(BaseEntrenamiento$GSPC.Return, 1:15) + Lags(BaseEntrenamiento$DFF.Return, 1:15)
Model 2: BaseEntrenamiento$GSPC.Return ~ Lags(BaseEntrenamiento$GSPC.Return, 1:15)
Res. Df  Df      F      Pr(>F)
1      2636
2      2651 -15 1.8185 0.02712 *
---
signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

- ✓ **Conclusión 4:** Las variaciones logarítmicas del SP500 causan en el sentido de Granger las variaciones logarítmicas del VIX con un rezago


```
Granger causality test
Model 1: BaseEntrenamiento$VIX.Return ~ Lags(BaseEntrenamiento$VIX.Return, 1:1) + Lags(BaseEntrenamiento$GSPC.Return, 1:1)
Model 2: BaseEntrenamiento$VIX.Return ~ Lags(BaseEntrenamiento$VIX.Return, 1:1)
Res.Df Df F Pr(>F)
1 2678
2 2679 -1 12.062 0.0005228 ***
---
Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

- ✓ **Conclusión 5:** Las variaciones logarítmicas del índice del dólar NO causan en el sentido de Granger las variaciones logarítmicas del VIX. Se intento hasta con 20 rezagos

```
Granger causality test
Model 1: BaseEntrenamiento$VIX.Return ~ Lags(BaseEntrenamiento$VIX.Return, 1:20) + Lags(BaseEntrenamiento$DTWEXBGS.Return, 1:20)
Model 2: BaseEntrenamiento$VIX.Return ~ Lags(BaseEntrenamiento$VIX.Return, 1:20)
Res.Df Df F Pr(>F)
1 2621
2 2641 -20 0.9392 0.5361
```

- ✓ **Conclusión 6:** Las variaciones logarítmicas de la tasa de interés NO causan en el sentido de Granger las variaciones logarítmicas del índice VIX. Se intento hasta con 20 rezagos.

```
Granger causality test
Model 1: BaseEntrenamiento$VIX.Return ~ Lags(BaseEntrenamiento$VIX.Return, 1:20) + Lags(BaseEntrenamiento$OFF.Return, 1:20)
Model 2: BaseEntrenamiento$VIX.Return ~ Lags(BaseEntrenamiento$VIX.Return, 1:20)
Res.Df Df F Pr(>F)
1 2621
2 2641 -20 1.1508 0.2891
```

- ✓ **Conclusión 7:** Las variaciones logarítmicas del SP500 causan en el sentido de Granger las variaciones logarítmicas del índice del dólar con un rezago.

```
Granger causality test
Model 1: BaseEntrenamiento$DTWEXBGS.Return ~ Lags(BaseEntrenamiento$DTWEXBGS.Return, 1:1) + Lags(BaseEntrenamiento$GSPC.Return, 1:1)
Model 2: BaseEntrenamiento$DTWEXBGS.Return ~ Lags(BaseEntrenamiento$DTWEXBGS.Return, 1:1)
Res.Df Df F Pr(>F)
1 2678
2 2679 -1 67.174 3.808e-16 ***
---
Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

- ✓ **Conclusión 8:** Las variaciones logarítmicas del índice VIX causan en el sentido de Granger las variaciones logarítmicas del indice del dólar con un rezago

```
Granger causality test
Model 1: BaseEntrenamiento$DTWEXBGS.Return ~ Lags(BaseEntrenamiento$DTWEXBGS.Return, 1:1) + Lags(BaseEntrenamiento$VIX.Return, 1:1)
Model 2: BaseEntrenamiento$DTWEXBGS.Return ~ Lags(BaseEntrenamiento$DTWEXBGS.Return, 1:1)
Res.Df Df F Pr(>F)
1 2678
2 2679 -1 21.576 3.565e-06 ***
---
Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

- ✓ **Conclusión 9:** Las variaciones logarítmicas de la tasa de interés causan en el sentido de Granger las variaciones logarítmicas del indice del dólar con un rezago

```
Granger causality test
Model 1: BaseEntrenamiento$DTWEXBGS.Return ~ Lags(BaseEntrenamiento$DTWEXBGS.Return, 1:1) + Lags(BaseEntrenamiento$OFF.Return, 1:1)
Model 2: BaseEntrenamiento$DTWEXBGS.Return ~ Lags(BaseEntrenamiento$DTWEXBGS.Return, 1:1)
Res.Df Df F Pr(>F)
1 2678
2 2679 -1 6.0141 0.01426 *
---
Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

- ✓ **Conclusión 10:** Las variaciones logarítmicas del SP500 causan en el sentido de Granger las variaciones logarítmicas de la tasa de interés con dos rezagos

```
Granger causality test
Model 1: BaseEntrenamiento$OFF.Return ~ Lags(BaseEntrenamiento$OFF.Return, 1:2) + Lags(BaseEntrenamiento$GSPC.Return, 1:2)
Model 2: BaseEntrenamiento$OFF.Return ~ Lags(BaseEntrenamiento$OFF.Return, 1:2)
Res.Df Df F Pr(>F)
1 2675
2 2677 -2 3.2806 0.03776 *
---
Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

- ✓ **Conclusión 11:** Las variaciones logarítmicas del indice VIX causan en el sentido de Granger las variaciones logarítmicas de la tasa de interés hasta 17 rezagos

```

Granger causality test
Model 1: BaseEntrenamiento$DFF.Return ~ Lags(BaseEntrenamiento$DFF.Return, 1:17) + Lags(BaseEntrenamiento$VIX.Return, 1:17)
Model 2: BaseEntrenamiento$DFF.Return ~ Lags(BaseEntrenamiento$DFF.Return, 1:17)
Res. DF   DF      F    Pr(>F)
1    2630
2    2647 -17  1.7055 0.03529 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

```

- ✓ **Conclusión 12:** Las variaciones logarítmicas del índice del dólar causan en el sentido de Granger las variaciones logarítmicas de la tasa de interés con dos rezagos

```

Granger causality test
Model 1: BaseEntrenamiento$DFF.Return ~ Lags(BaseEntrenamiento$DFF.Return, 1:2) + Lags(BaseEntrenamiento$OTWEXBG5.Return, 1:2)
Model 2: BaseEntrenamiento$DFF.Return ~ Lags(BaseEntrenamiento$DFF.Return, 1:2)
Res. DF   DF      F    Pr(>F)
1    2675
2    2677 -2  4.2576 0.01425 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

```

En general se concluye que los resultados de cada variable y sus rezagos pueden predecir de manera bidireccional de las demás variables. Por lo demostrado anteriormente se concluye que el modelo VAR es adecuado para nuestro proyecto.

- **Modelos de Ensamble y Redes Neuronales:** Se realizó la identificación de features más relevantes, aplicando la función tsfeatures en RStudio a la serie de tiempo del S&P 500, y obtuvimos los siguientes resultados:

frequency <dbl>	nperiods <dbl>	seasonal_period <dbl>		trend <dbl>	spike <dbl>	linearity <dbl>	curvature <dbl>	e_acf1 <dbl>	e_acf10 <dbl>	
365	1	365		0.004775456	1.285495e-07	0.06862472	1.496762	0.3406603	0.1490159	
	seasonal_strength <dbl>	peak <dbl>	trough <dbl>	entropy <dbl>	x_acf1 <dbl>	x_acf10 <dbl>	diff1_acf1 <dbl>	diff1_acf10 <dbl>	diff2_acf1 <dbl>	
	0.1794018	1	313	0.9837202	0.3416181	0.1467163	-0.3696743	0.1497599	-0.630538	
1 row 10-18 of 20 columns										
	peak <dbl>	trough <dbl>	entropy <dbl>	x_acf1 <dbl>	x_acf10 <dbl>	diff1_acf1 <dbl>	diff1_acf10 <dbl>	diff2_acf1 <dbl>	diff2_acf10 <dbl>	seas_acf1 <dbl>
	1	313	0.9837202	0.3416181	0.1467163	-0.3696743	0.1497599	-0.630538	0.4311978	-0.008426393

Estos resultados nos muestran que tenemos una serie con baja linealidad y alta entropía, lo cual hace que tenga características que la hacen difícil de pronosticar, y nos lleva a deducir que modelos lineales no se ajustarán muy bien a los datos.

Por tanto, emplearemos modelos tipo Data Based como Redes Neuronales “LSTM” y Modelos de Ensamble como Random Forest que son propios para series con alta entropía.

4. Implementación de modelos para predicciones del S&P500

- **Modelo # 1 – Red Neuronal –LSTM**

Las redes neuronales artificiales es un método que procesa datos de manera similar en que lo hace el ser humano. De manera resumida, una red neuronal funciona de la siguiente manera: Existe una capa de entrada con un conjunto de neuronas que acepta los datos, es decir los predictores, luego las neuronas que se encuentran en la capa oculta procesan la información de la anterior capa, y generalmente la última capa consiste en una neurona que consolida todo el procesamiento de las capas internas para estimar el valor de la variable de respuesta.

La LSTM (Long-Short Term Memory) son un tipo de red neuronal artificial, exactamente una extensión de las redes neuronales recurrentes (RNN) donde puede aprender dependencias a largo plazo entre unidades de tiempo de datos secuenciales, es decir las LSTM permiten a las RNN recordar sus entradas durante un largo período.

Entre las ventajas de este modelo se encuentran: Presentan capacidad para manejar secuencia de datos de longitud variable, pueden recordar información a largo plazo y actualmente existen diversas bibliotecas de software que permiten una fácil implementación de estos modelos

Para la implementación de este modelo, el propósito es predecir el rendimiento logarítmico del S&P500 teniendo en cuenta algunas acciones del portafolio, y algunos indicadores macroeconómicos de la misma periodicidad diaria que el S&P500, los cuales son el índice VIX y el dólar Index

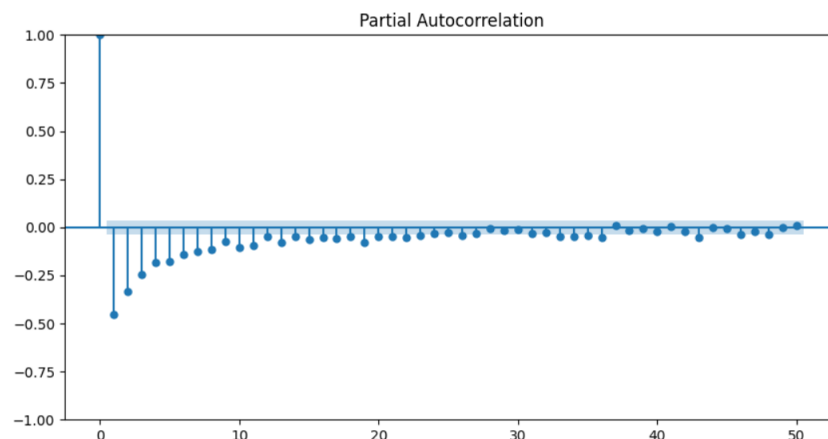
4..1. Calibración y entrenamiento del modelo

Para el proceso de calibración y el entrenamiento del modelo se dividió la base de datos en tres partes. Una primera base servirá para en el entrenamiento de la red neuronal, compuesta por los datos de los rendimientos logarítmicos de 2767 observaciones; un set de datos de validación que se utilizará para validar los pesos y la estructura de la red neuronal, compuesta por los siguientes 20 datos de la serie, y un set de datos final de test, que no se utilizarán en la creación de estructura ni de pesos de la red, que servirán para hacer un testing de la red sobre datos no visualizados anteriormente

a. Selección de variables

En el proceso de selección de variables y su transformación se siguieron los siguientes pasos:

- Se realiza la función de autocorrelación parcial (PACF) de los rendimientos del S&P500 en donde se observa que existen varios rezagos significativos negativos. Para este experimento utilizamos los primeros 5 rezagos como variables predictoras autoregresivas



- Adicionalmente basado en el análisis de correlación presentado en el análisis descriptivo, para este experimento utilizando redes neuronales LSTM se seleccionan las dos variables macroeconómicas VIX y Dólar Index, y en el caso de las variables del portafolio las dos acciones con mayor índice de correlación, en este caso BRK.B y MSFT
- Dado que se seleccionan los anteriores 5 rezagos de los rendimientos del S&P 500, se realiza también los 5 rezagos de las anteriores variables mencionadas, y se agregan a la base de datos
- Las redes neuronales, debido al uso de las funciones de activación, necesitan que los datos estén escalados entre 0 y 1, y poder experimentar con diferentes funciones de mejor manera. Para el caso de este experimento, el proceso se realiza en dos pasos: la primer set de datos de entrenamiento es utilizado para ajustar un escalador MinMaxScaler(), y con este escalador se transforma el set de datos de validación. El segundo paso es, luego de encontrada la estructura final de la red, se utilizará la suma de los set de datos de entrenamiento y validación para ajustar otro escalador, con el fin de transformar los datos finales de test.

- Por medio de la función *LocalOutlierFactor* en python, se realiza suavizamiento de los datos atípicos previamente escalados
- Los datos resultantes se cambian a tipos de datos de numpy reestructurados de la manera necesaria para alimentar la red
- La información resultante se ingresa a una red neuronal de tipo LSTM donde se optimizaron sus hiperparámetros por medio del paquete Optuna, y se describirá a continuación

b. Parametrización

Como anteriormente se mencionó se crea una red neuronal de tipo LSTM donde se optimizaron sus hiperparámetros por medio del paquete optuna. Este paquete realiza, de acuerdo con los parámetros establecidos, una búsqueda en el espacio definido, con el fin de minimizar o maximizar una función objetivo. Para este experimento, se hizo uso del siguiente espacio de variables, con sus posibles valores:

- Número de capas de la red: De 1 a 4
- Número de neuronas por capa: Desde la cantidad de variables del set de entrenamiento hasta 400
- Función de activación por capa: relu, linear, swish y sigmoid
- Porcentaje de Dropout por capa: la función de Dropout es una función de regularización utilizada en la estructura de redes neuronales donde aleatoriamente se ignoran algunas neuronas en la fase de entrenamiento haciendo que su peso sea 0. Normalmente se utiliza un rango entre 0 y 0.5

Con respecto al método *ReduceLROnPlateau* en python, el cual reduce la tasa de aprendizaje cuando la métrica objetivo ha dejado de mejorar, se utilizaron los siguientes rangos para calibración:

- Factor: En un rango entre 0 y 0.5
- Patience: En un rango entre 5 a 20
- Para este experimento, la última capa de la red es una capa de condensación de una neurona, en la cual se calibrará la función de activación final de dicha neurona, la cual puede ser relu, linear, swish y sigmoid

Finalmente, al compilar el modelo se calibra el optimizador entre adagrad, adam, sgd y RMSprop

La función definida para ejecutar el modelo de optimización con el paquete optuna buscará el modelo que minimice el valor de la métrica Val_Loss del entrenamiento de la red. Se permiten hacer 30 ejecuciones con un tiempo máximo de 1800 segundos, como se muestra a continuación

```
def objective(trial):
    keras.backend.clear_session()
    n_layers = trial.suggest_int('n_layers', 1, 4)
    model = keras.Sequential()
    for i in range(n_layers):
        num_hidden = trial.suggest_int(f'n_units_{i}', data_train.shape[1]-1, 400, log=True)
        model.add(keras.layers.LSTM(num_hidden, input_shape=(1, data_train.shape[1]-1), return_sequences=True,
                                   activation=trial.suggest_categorical(f'activation_{i}', ['relu', 'linear', 'swish', 'sigmoid'])))
        model.add(keras.layers.Dropout(rate = trial.suggest_float(f'dropout_{i}', 0.0, 0.5)))
    model.add(keras.layers.Dense(1, activation=trial.suggest_categorical(f'finalact', ['relu', 'linear', 'swish', 'sigmoid'])))
    val_ds = (xval_n, yval_n)
    reduce_lr = ReduceLROnPlateau(monitor='val_loss', factor=trial.suggest_float('LRfactor', 0.0, 0.5), patience=trial.suggest_int('patience', 5, 20))
    model.compile(loss='mse', optimizer=trial.suggest_categorical(f'optimizer', ['Adagrad', 'adam', 'sgd', 'RMSprop']))
    run_history = model.fit(xtrain_n, ytrain_n, validation_data=val_ds, epochs=50, callbacks=[reduce_lr], verbose=0)
    return min(run_history.history['val_loss'])

study = optuna.create_study(direction="minimize")
study.optimize(objective, n_trials=30, timeout=1800)
print("Number of finished trials: {}".format(len(study.trials)))
print("Best trial:")
trial = study.best_trial
print("  Value: {}".format(trial.value))
```

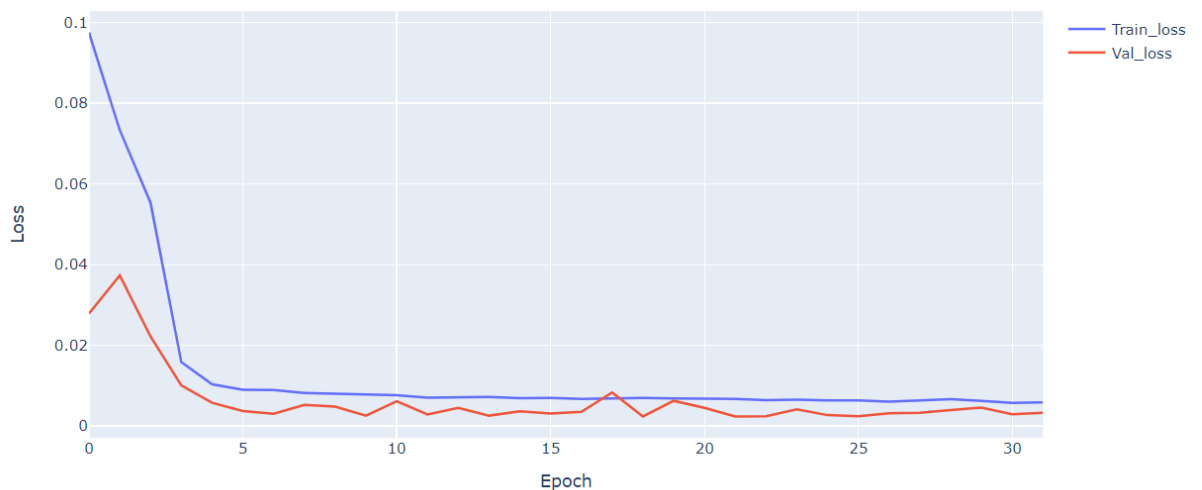
Aplicando el paquete optuna los resultados son los siguientes:

```
Params:
  n_layers: 3
  n_units_l0: 197
  activation0: linear
  dropout0: 0.14939677163716047
  n_units_l1: 193
  activation1: swish
  dropout1: 0.40035016781108546
  n_units_l2: 376
  activation2: sigmoid
  dropout2: 0.29387234491389896
  finalact: linear
  LRfactor: 0.16202265241592126
  LRpatience: 10
  optimizer: adam
```

Se tienen 3 capas donde la primera tiene 197 neuronas con función de activación lineal y porcentaje de dropout de 15%, la segunda 193 con función de activación swish y dropout de 40% y la tercera 376 con función de activación sigmoide y dropout de 29%, con una función de activación de la última capa lineal. En factor por el cual se reducirá la tasa de aprendizaje es 0.16 y el número de épocas sin mejora después de las cuales la tasa de aprendizaje se reducirá es 10. El optimizador seleccionado es Adam

Podemos ver cómo evoluciona las funciones de pérdida del conjunto de entramiento y del conjunto de validación a través de las épocas, y como el modelo se estabiliza en épocas menores a las definidas gracias a la definición del EarlyStopping

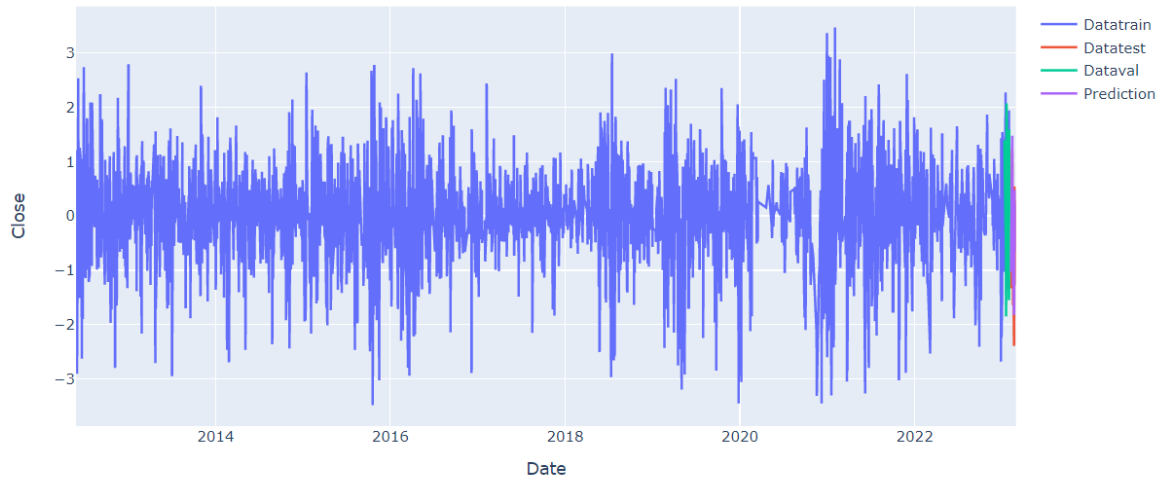
Optuna Loss best model History



Luego de tener ya la estructura de la red definida, se reentrena la red con el conjunto de datos de entrenamiento y de validación agregados, para predecir el conjunto de test.

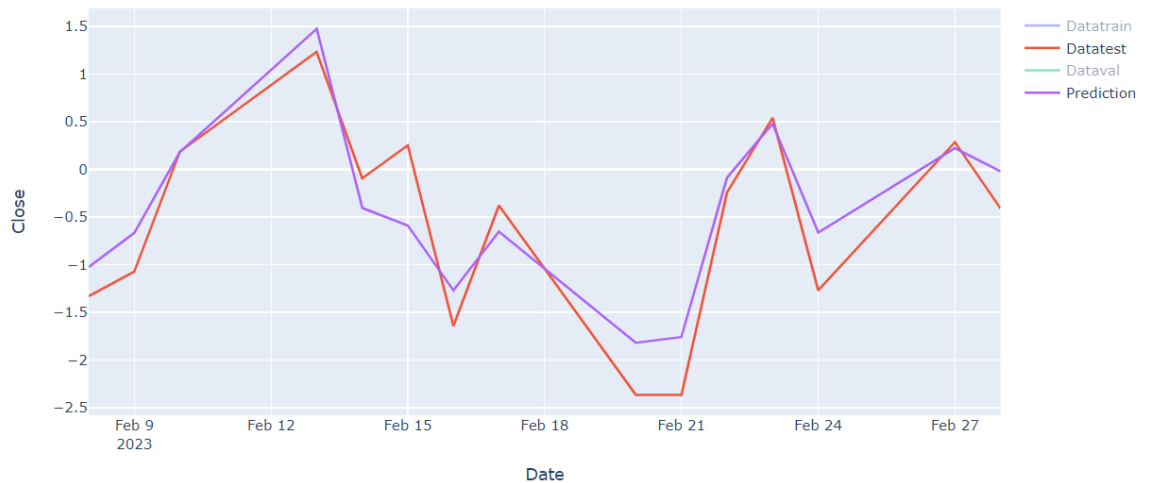
A continuación se muestra gráficamente el conjunto de entrenamiento, validación y de prueba tomando el modelo con la mejor optimización de hiperparametros:

S&P 500 Forecast



Haciendo revisión en el conjunto de test y de predicción del modelo vemos los resultados

S&P 500 Forecast



c. Definición de métricas adecuadas de desempeño

Evaluando el modelo en el conjunto de prueba se obtienen las siguientes métricas:

Métrica	Valor
RMSE	0.4126
MAE	0.3460
R^2	0.8381

- Modelo # 2 – Modelos vectoriales autorregresivos**

Estos modelos pueden considerarse una extensión de los modelos autorregresivos AR(p) y se utiliza cuando se quiere caracterizar las interacciones simultaneas entre un grupo de variables, por lo tanto, no existe una variable dependiente y un conjunto de variables independientes que intentan explicarla, si no que existe un sistema de ecuaciones constituido por un bloque de rezagos de cada una de las variables del modelo que presentan interacción entre sí.

En general un modelo VAR de orden n , se puede expresar de la siguiente manera:

$$y_t = A_0 + \sum_{s=1}^n A_s y_{t-s} + GW_t + u_t$$

Donde Y_t es un vector columna, n es el orden del modelo VAR o el número de retrasos de cada variable de la ecuación, U_t es un vector de innovación y W_t es un vector de variable exógenas, que en nuestro problema no se tendrán en cuenta.

La ventaja más relevante para este modelo es que no necesita especificar cuáles variables con endógenas o exógenas.

Para este modelo se usó los retornos logarítmicos del S&P500, VIX, índice del dólar y la tasa efectiva de los fondos federales, esta tasa presenta información diaria

4.1. Calibración y entrenamiento del modelo

Para el proceso de calibración y el entrenamiento del modelo se dividió la base de datos en dos partes. En donde el conjunto de prueba consta de los últimos 20 datos de la base.

a. Selección de variables

Se tomaron los retornos logarítmicos del S&P500, VIX, índice del dólar y la tasa efectiva de los fondos federales y se realizaron pruebas de causalidad de Granger para determinar si cada variable y sus rezagos pueden predecir de manera unidireccional o bidireccional las demás variables, lo cual se demostró en el literal 3.

b. Parametrización

Este modelo se implementó en R en donde la función *VARselect* determina el número de rezagos que se deben tener en cuenta en el modelo siguiendo los criterios de información AIC, SC, Hannan Quinn y FPE, en donde se obtienen los siguientes resultados:

```
$selection
AIC(n)  HQ(n)  SC(n) FPE(n)
    9      2    2     9
```

Por lo tanto el proceso de calibración para este modelo consiste en:

- Obtener el número de rezagos para cada una de las variables, según los criterios de información puede ser 2 o 9
- Incluir o no la constante en las ecuaciones de cada una de las variables

Finalmente el modelo con menor RMSE es aquel con 2 rezagos y teniendo en cuenta la constante para las 4 ecuaciones. Los resultados son los siguientes:

✓ Ecuación para el S&P500:

```

Estimation results for equation GSPC.Return:
=====
GSPC.Return = GSPC.Return.11 + VIX.Return.11 + DTWEXBGS.Return.11 + DFF.Return.11 + GSPC.Return.12 + VIX.Return.12 + DTWEXBGS.Return.12 + DFF.Return.12 + const

      Estimate Std. Error t value Pr(>|t|)
GSPC.Return.11 -0.2301736 0.0292568 -7.867 5.24e-15 ***
VIX.Return.11 -0.0199145 0.0039025 -5.103 3.58e-07 ***
DTWEXBGS.Return.11 0.0086255 0.0718752 0.120 0.904487
DFF.Return.11 -0.0024928 0.0022354 -1.115 0.264886
GSPC.Return.12 0.0531917 0.0296519 1.794 0.072949 .
VIX.Return.12 0.0004061 0.0039308 0.103 0.917721
DTWEXBGS.Return.12 -0.2398077 0.0699885 -3.426 0.000621 ***
DFF.Return.12 -0.0002983 0.0022369 -0.133 0.893926
const 0.0521122 0.0212307 2.452 0.014261 *
---
Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 1.086 on 2637 degrees of freedom
Multiple R-squared: 0.03757, Adjusted R-squared: 0.03465
F-statistic: 12.87 on 8 and 2637 DF, p-value: < 2.2e-16

```

✓ Ecuación para el VIX:

```

Estimation results for equation VIX.Return:
=====
VIX.Return = GSPC.Return.11 + VIX.Return.11 + DTWEXBGS.Return.11 + DFF.Return.11 + GSPC.Return.12 + VIX.Return.12 + DTWEXBGS.Return.12 + DFF.Return.12 + const

      Estimate Std. Error t value Pr(>|t|)
GSPC.Return.11 0.779550 0.213666 3.648 0.000269 ***
VIX.Return.11 -0.004797 0.028501 -0.168 0.866346
DTWEXBGS.Return.11 0.315342 0.524913 0.601 0.548058
DFF.Return.11 0.008940 0.016325 0.548 0.583987
GSPC.Return.12 0.230066 0.216552 1.062 0.288149
VIX.Return.12 -0.010814 0.028707 -0.377 0.706427
DTWEXBGS.Return.12 0.167509 0.511134 0.328 0.743149
DFF.Return.12 0.010036 0.016336 0.614 0.539066
const -0.054410 0.155196 -0.351 0.725928
---
Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 7.931 on 2637 degrees of freedom
Multiple R-squared: 0.01248, Adjusted R-squared: 0.009481
F-statistic: 4.165 on 8 and 2637 DF, p-value: 5.797e-05

```

✓ Ecuación para el índice del dólar:

```

Estimation results for equation DTWEXBGS.Return:
=====
DTWEXBGS.Return = GSPC.Return.11 + VIX.Return.11 + DTWEXBGS.Return.11 + DFF.Return.11 + GSPC.Return.12 + VIX.Return.12 + DTWEXBGS.Return.12 + DFF.Return.12 + const

      Estimate Std. Error t value Pr(>|t|)
GSPC.Return.11 -0.0596555 0.0083617 -7.134 1.25e-12 ***
VIX.Return.11 -0.0029226 0.0011134 -1.813 0.0699 .
DTWEXBGS.Return.11 -0.0083292 0.0205422 -0.405 0.6852
DFF.Return.11 -0.0015600 0.0006389 -2.442 0.0147 *
GSPC.Return.12 -0.0108027 0.0084746 -1.275 0.2025
VIX.Return.12 -0.0015075 0.0011234 -1.342 0.1797
DTWEXBGS.Return.12 0.0192728 0.0200030 0.963 0.3354
DFF.Return.12 -0.0010766 0.0006393 -1.684 0.0923 .
const 0.0124751 0.0060735 2.054 0.0401 *
---
Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.3104 on 2637 degrees of freedom
Multiple R-squared: 0.03259, Adjusted R-squared: 0.02966
F-statistic: 11.11 on 8 and 2637 DF, p-value: 1.51e-15

```

✓ Ecuación para la tasa:

```

Estimation results for equation DFF.Return:
=====
DFF.Return = GSPC.Return.11 + VIX.Return.11 + DTWEXBGS.Return.11 + DFF.Return.11 + GSPC.Return.12 + VIX.Return.12 + DTWEXBGS.Return.12 + DFF.Return.12 + const

      Estimate Std. Error t value Pr(>|t|)
GSPC.Return.11 -0.22734 0.23282 -0.900 0.3682
VIX.Return.11 -0.03547 0.03372 -1.052 0.2930
DTWEXBGS.Return.11 -0.65133 0.62111 -1.049 0.2944
DFF.Return.11 -0.34660 0.01932 -17.943 < 2e-16 ***
GSPC.Return.12 0.64668 0.25624 2.524 0.0117 *
VIX.Return.12 0.06742 0.03397 1.985 0.0473 *
DTWEXBGS.Return.12 -1.15390 0.60480 -1.908 0.0565 .
DFF.Return.12 -0.11593 0.01933 -5.997 2.28e-09 ***
const 0.18127 0.18364 0.987 0.3237
---
Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 9.384 on 2637 degrees of freedom
Multiple R-squared: 0.1145, Adjusted R-squared: 0.1119
F-statistic: 42.64 on 8 and 2637 DF, p-value: < 2.2e-16

```

c. Definición de métricas adecuadas de desempeño

Evaluando el modelo en el conjunto de prueba se obtienen las siguientes métricas:

Métrica	Valor
RMSE	0.9316
MAE	0.7508
R ²	-0.0135

• Modelo # 3 – Forecaster Autoregresor de Skforercaster con Random Forest

El *Forecaster Autoregresor* funciona como una herramienta bastante útil para realizar predicciones de series de tiempo usando modelos de regresión que emplean valores anteriores de la serie

temporal como datos de entrada. Se encuentra en la biblioteca de Python y es una clase que hace parte de “scikit-learn”. Funciona con cualquier regresor de la biblioteca de “scikit-learn” (pipelines, CatBoost, LightGBM, XGBoost, entre otros).

Los valores anteriores que utiliza este modelo son conocidos como “lags”, “rezagos” o “retrasos” y son las variables de entrada de los modelos. Adicionalmente, permite incluir variables exógenas como predictores e incluir predictores personalizados como (media móvil, varianza móvil, entre otros).

Tiene la ventaja de permitir la optimización de hiperparámetros a través de búsqueda de cuadrícula y además de ofrecer la posibilidad de personalizar sus métricas para validar el modelo. Adicionalmente, la funcionalidad de obtener intervalos de predicción y conocer la importancia de los predictores del modelo.

Dado que cuando se trabaja por predicciones, generalmente se quiere predecir no solo un siguiente momento de la serie o *step*, si no varios, existen estrategias que permiten generar predicciones de múltiples. Para esto, es importante tener en cuenta que para predecir el momento t_n se requiere conocer el valor de t_{n-1} , es decir, siempre se hace uso del valor del día anterior para predecir el día siguiente. A este proceso se le conoce como *recursive forecasting* y puede generarse fácilmente a través de las clases `ForecasterAutoreg` y `ForecasterAutoregCustom` de `skforecast`.

En cuanto al regresor utilizado, *Random Forest Regressor*, es un regresor de bosque aleatorio que ajusta un número determinado de árboles de decisión en varias submuestras del conjunto de datos utilizando el promedio para controlar su capacidad de predicción y evitar el sobreajuste. Recordemos que los árboles de decisión comienzan con la raíz del árbol y hacen divisiones hasta que se alcanza el nodo hoja y se obtiene un resultado. Este método se basa en la construcción de una gran colección de árboles que no están correlacionados entre sí y que luego promedia para obtener una mejor calidad de la predicción. Además, son muy útiles para estructuras de datos complejas dado que logran a través de las particiones leer mucho mejor sus comportamientos vs modelos tradicionales de regresión lineal.

Random Forest proporciona ventajas al ser uno de los algoritmos de aprendizaje que corre eficientemente para grandes cantidades de datos con alto volumen de variables, permite entender cuáles son las variables más importantes dentro del modelo y ha demostrado altos niveles de precisión en el uso de series temporales. Sin embargo, es importante tener en cuenta que reduce interpretabilidad y tiene el riesgo de caer en sobreajuste, por lo que se implementarán estrategias de partición de los datos para evitarlo.

Para nuestro caso de estudio de predicciones del S&P500, combinar estos dos modelos es muy útil dado que todo lo explicado anteriormente responde a las necesidades de la herramienta que debemos generar. Haremos pruebas con las dos estrategias, incluyendo además predictores exógenos que nos permitan alimentar el modelo con mayor cantidad de información.

4.1. Calibración y entrenamiento del modelo

a. Selección de variables

Se configura la frecuencia de las series de tiempo con la función `asfreq()` con el parámetro “B” que indica información de Business Days. Al incluir este parámetro el dataframe resultante contiene un total de 2812 datos pues se rellenan los días de negocios sin información con el parámetro “bfill” que indica imputar la información del día siguiente.

Se construyen los conjuntos de entrenamiento y test bajo el siguiente criterio:

- Los últimos 20 datos se dejarán para las pruebas finales
- Con los 2792 datos restantes se construye un conjunto de train con 2772 datos y un conjunto de test con 20 datos para evaluar la calidad de las predicciones.

Se estandarizan los datos con *StandardScaler()* de la librería sklearn. Y se procede a iniciar con la parametrización de modelos, inicialmente incluyendo todas las variables con las cuales se cuenta.

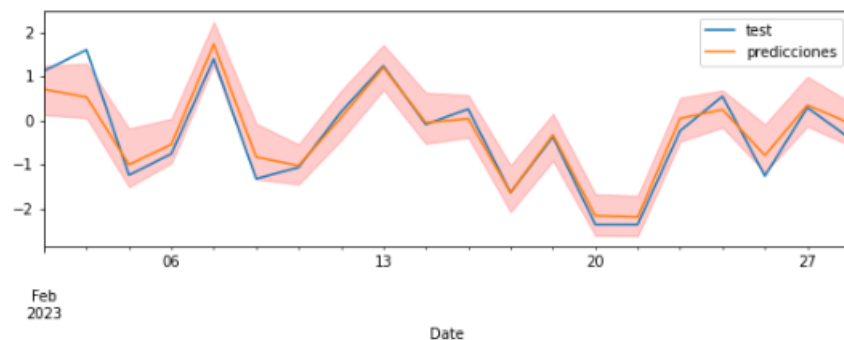
b. Parametrización

- Experimento # 1: Se implementa un modelo *Forecaster Autoregresor* inicialmente definido para 14 lags, incluyendo el conjunto de train, las variables exógenas seleccionados y con un *RandomForestRegressor* como regresor. Se optimizan los siguientes hiperparámetros con *grid_search_forecaster* lags = [5, 20], n_estimators = [100, 500] y max_depth = [3, 5, 10]. Como resultado de la optimización obtenemos los siguientes hiperparámetros configurando el parámetro return_best = True para que se guarde el mejor modelo para las predicciones:

```
`Forecaster` refitted using the best-found lags and parameters, and the whole data set:  
Lags: [1 2 3 4 5]  
Parameters: {'max_depth': 10, 'n_estimators': 500}  
Backtesting metric: 0.3250850094881204
```

Se realizan las predicciones y se encuentra una precisión con mse de 0.13 y un R2 de 0.90. Luego, se corre el modelo con el 100% del conjunto de datos de train y se obtienen resultados similares con un mse de 0.12 y un R2 de 0.90.

```
: grafico_intervalos(df_testall.iloc[:,0],pred_1fin)
```



Dado que el modelo requiere de la entrada de variables exógenas, se analiza la importancia de los predictores encontrando que los datos de las acciones de Microsoft y BRK.B y del VIX son las 3 variables exógenas que mayor contribuyen a la predicción del S&P500.

	feature	importance			
0	lag_1	0.008050	10	GOOGL.ReturnSuavizado	0.044740
1	lag_2	0.005388	11	NVDA.ReturnSuavizado	0.026820
2	lag_3	0.005674	12	BRK.B.ReturnSuavizado	0.328805
3	lag_4	0.006963	13	META.ReturnSuavizado	0.008780
4	lag_5	0.006396	14	UNH.ReturnSuavizado	0.016280
5	AAPL.ReturnSuavizado	0.043067	15	JNJ.ReturnSuavizado	0.013042
6	AMZN.ReturnSuavizado	0.023042	16	PG.ReturnSuavizado	0.014296
7	MSTF.ReturnSuavizado	0.176339	17	VIX.ReturnSuavizado	0.204769
8	TSLA.ReturnSuavizado	0.009582	18	DolarIndex.ReturnSuavizado	0.010659
9	GOOG.ReturnSuavizado	0.047308			

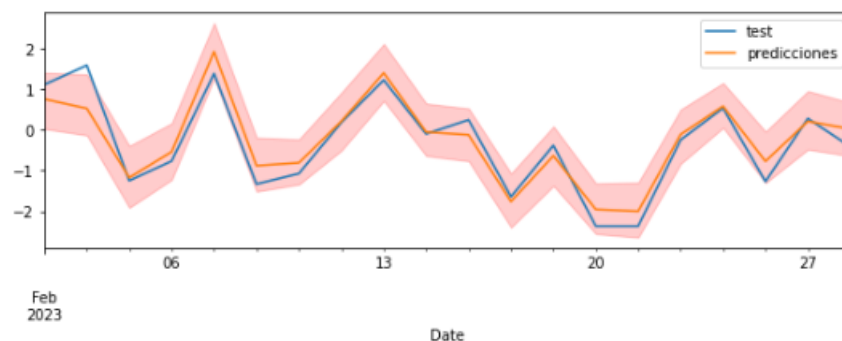
El mapa de correlación nos muestra que al incluir los precios de las acciones de empresas como Microsoft (MSFT) y Berkshire Hathaway (BRK.B) en el modelo para predecir el S&P500, podríamos obtener una mejor precisión pues dado que la correlación indica una relación lineal entre dos variables, los precios de las acciones de MSFT, BRK.B y el S&P500. Como vemos, encontramos una correlación positiva alrededor del 0.7, lo que puede interpretarse como que los precios de las acciones se mueven en la misma dirección que el S&P500.

De otro lado, al revisar el portafolio de acciones, observamos que el precio promedio de estas dos acciones está dentro de las acciones con los mayores precios de las seleccionadas, lo cual, vinculándolo a la forma como se calcula el S&P500 hace que estas acciones tengan un alta importancia en el cálculo del resultado final del S&P. Si bien, hay otras acciones con mayores precios no tienen una correlación tan fuerte con el índice, razón por la cual no se incluirán.

- Experimento # 2: Con lo anterior, y teniendo en cuenta que el modelo requiere contar con los datos futuros de las variables exógenas, lo cual implica que debemos construir modelos predictivos para estos predictores, se procede a probar qué sucede con el desempeño si se corre el modelo únicamente con estas 3 variables y haciendo uso de los hiperparámetros obtenidos (5 lags, max_depth 10 y n_estimators 500).

```
: grafico_intervalos(df_test_filtado.iloc[:,0],pred_filtado)
desempeño(df_test_filtado,pred_filtado)
```

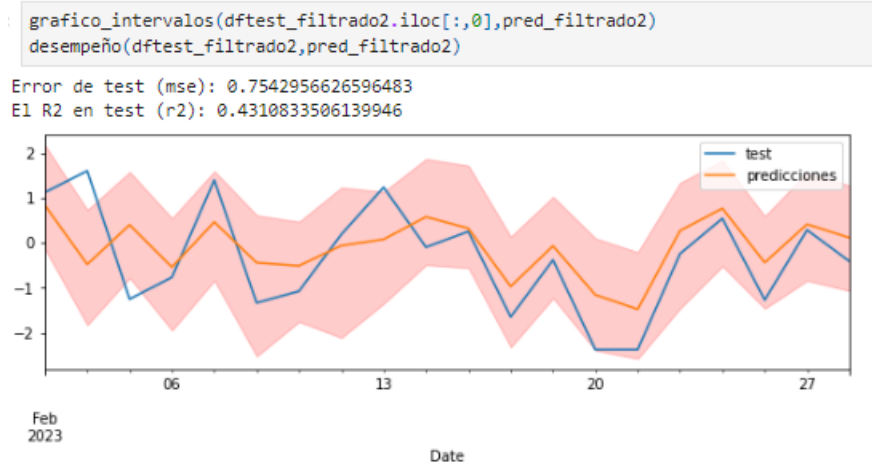
Error de test (mse): 0.14504318029190116
El R2 en test (r2): 0.8906032684093637



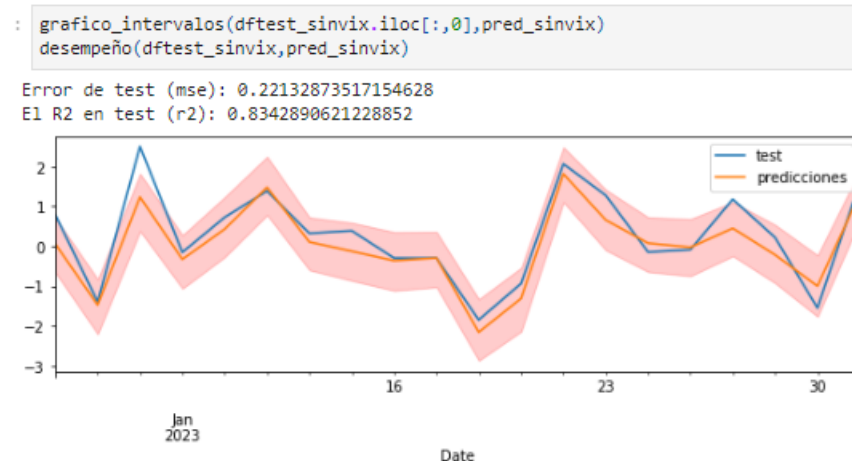
Los resultados nos muestran que el desempeño es significativamente similar y que no hay una reducción considerable si omitimos la información de las demás acciones. Ahora bien, el VIX es un indicador que mide la volatilidad del índice S&P500 para un periodo de los 30 días siguientes, se generaliza como un indicador de “miedo” en el sentido en que es una medida de sentimiento del mercado (en particular, la preocupación). Este indicador tiene una correlación negativa con el S&P500 lo cual indica que si el VIX sube el S&P baja y viceversa. Dentro de este contexto, este puede ser un indicador complejo para predecir ya que podría implicar casi

lo mismo que predecir el S&P500 (objetivo de este proyecto). La importancia de los predictores nos muestra que este contribuye con un 0.2 a la predicción del S&P500 y es el segundo de los 3 predictores más importantes. Procederemos a examinar qué sucede con el modelo si incluimos solamente el VIX y qué sucede si no se tiene en cuenta.

- Experimento # 3: Entrenamos el modelo con los mismos hiperparámetros ya conocidos y con el VIX como única variable exógena. Los resultados nos muestran que solo con el VIX el desempeño del modelo se reduce significativamente, casi a la mitad aumentando el mse a 0.75 y reduciendo el R2 a 0.43.



- Experimento # 4: Entrenamos ahora el modelo omitiendo los datos del VIX y dejando como variables exógenas las acciones de MSTF y BRK.B. Los resultados nos muestran que aunque el desempeño sí disminuye no lo hace de manera considerable como sí sucede cuando se incluye únicamente el VIX. Y el incremento del mse no es significativo.



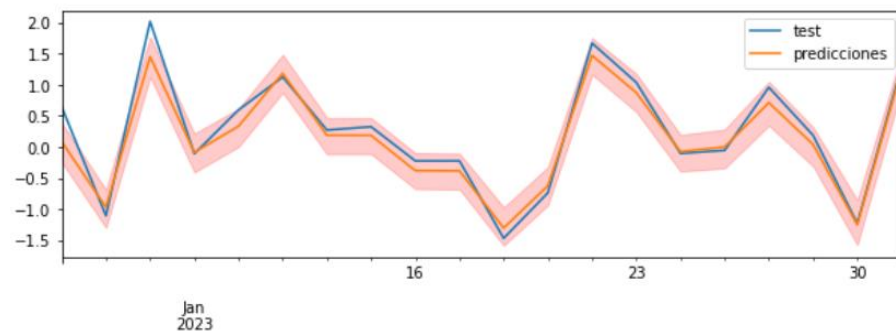
Por lo anterior, y entendiendo la complejidad de la predicción del VIX, trabajaremos con el modelo que incluye como variables exógenas las dos acciones mencionadas. Esto implica que debemos entrenar modelos para predecir sus comportamientos futuros. Es importante destacar que el desempeño del modelo puede verse impactado por el desempeño de los modelos para predecir las variables exógenas.

Experimento # 5: Como recomendación del equipo docente, se sugirió incluir más variables incluso adicionales a las que conforman el portafolio del grupo Stanley para validar cómo estas

pueden aportar al desempeño del modelo. Al realizar el experimento incluyendo como predictores exógenos las dos principales acciones de cada uno de los 9 sectores que componen el S&P500 encontramos los siguientes resultados:

```
desempeño(df_test.iloc[:, -1], pred_1)
```

```
EL MSE del modelo en test es de: 0.0502049410477914
EL RMSE del modelo en test es de: 0.2240645912405425
EL R2 del modelo en test es de: 0.9413740591313827
EL MAE del modelo en test es de: 0.16772053200778625
```



Al revisar la importancia de los predictores, encontramos que MSFT y el BRK.B continúan siendo los predictores más importantes dentro del modelo. Las demás variables agregadas no tienen el mismo nivel de relevancia para la predicción, y aunque si logramos un mejor desempeño, esta mejora podría perderse significativamente con los errores que generen los modelos de todas las acciones que se requiere predecir para agregar como variables exógenas al modelo. Es por esta razón que se toma la decisión de seleccionar el modelo del Experimento # 4 como modelo seleccionado para las predicciones del S&P500 en el proyecto.

	feature	importance		feature	importance
18	BRK-B	0.350264	8	T	0.010384
24	MSFT	0.211808	9	CF	0.010162
21	GOOGL	0.132120	15	PG	0.009097
20	GOOG	0.047184	10	PPL	0.008198
25	AAPL	0.036613	6	OXY	0.007573
13	WFC	0.033768	22	TSLA	0.005544
14	JNJ	0.024942	17	META	0.004521
12	GE	0.018636	3	lag_4	0.003608
7	AMT	0.016652	5	VZ	0.003420
11	XOM	0.015804	0	lag_1	0.003333
19	NVDA	0.014850	2	lag_3	0.003206
16	UNH	0.012248	1	lag_2	0.003082
23	AMZN	0.011072	4	lag_5	0.001911

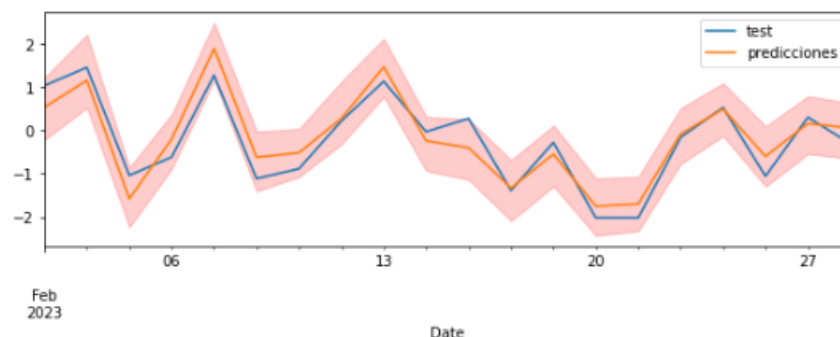
Definido el modelo final, procedemos a correr el modelo con el 100% de los datos entrenamiento y, teniendo en cuenta que escalamos los datos, los devolvemos a sus dimensiones originales para obtener las predicciones de los retornos, obteniendo los siguientes resultados finales:

```

: # Calculamos las métricas
# =====
grafico_intervalos(return_real.iloc[:,0],pred_final_1)
desempeño(return_real,pred_final_1)

```

Error de test (mse): 0.14076582767600526
El R2 en test (r2): 0.8625737678146981



c. Definición de métricas adecuadas de desempeño

Evaluando el modelo en el conjunto de prueba para el modelo que finalmente se decidió implementar se obtienen las siguientes métricas:

Métrica	Valor
RMSE	0.375
MAE	0.326
R²	0.862

5. Análisis de resultados

Métrica	LSTM	VAR	Forecaster Autoregresor
RMSE	0.4126	0.9316	0.375
MAE	0.3460	0.7508	0.326
R²	0.8381	-0.0135	0.862

Los resultados finales nos muestran que el modelo con mejor desempeño es el SKForecaster con Random Forest, que aunque incluye una complejidad adicional al proyecto que supone tener que predecir los precios de dos de las acciones del portafolio para ingresar como variables exógenas al modelo, el desempeño que se obtiene frente a los demás modelos es significativamente superior, por lo cual se hará este paso adicional dentro del proyecto y se seleccionará el modelo del Experimento # 4 con el SKForecaster con Random Forest como modelo para la construcción de la herramienta.

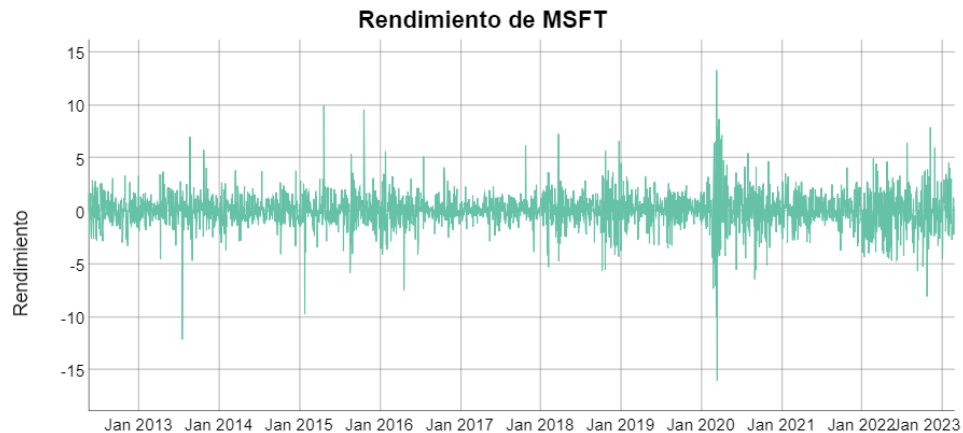
6. Modelos para predicción de variables exógenas para el modelo - Acciones MSFT y BRK

Para usar el modelo Forecaster Autoregresor de Skforercaster con Random Forest es necesario predecir de manera univariada los rendimientos logarítmicos del MSFT y BRK

5.1 Predicción de la acción de MSFT

Para realizar la predicción de los rendimientos logarítmicos se usa el modelo GARCH, el cual encuentra la volatilidad promedio a medio plazo mediante una autorregresión que depende de la suma de perturbaciones rezagadas y de la suma de varianzas rezagadas. El procedimiento realizado y los resultados obtenidos son los siguientes:

- a. La información se descargó directamente de Yahoo Finance y se calculó los rendimientos logarítmicos. Gráficamente se ve de la siguiente manera:



- b. Posteriormente se toma el 20% de la información como conjunto de prueba y el resto, como la base de entrenamiento. Aproximadamente son 500 datos para el test.
- c. Se halla el modelo *ARMA* de la media por medio de la función *auto.arima()* de R. El cual arroja un *AR(1)* con media diferente de cero.

```
Series: BaseEntrenamiento
ARIMA(1,0,0) with non-zero mean

Coefficients:
      ar1      mean
    -0.1625  0.1050
s.e.    0.0210  0.0298

sigma^2 = 2.64:  log likelihood = -4188.55
AIC=8383.1   AICc=8383.11   BIC=8400.19

Training set error measures:
              ME      RMSE      MAE  MPE  MAPE      MASE
Training set 3.06991e-05 1.62408 1.088033 NaN  Inf  0.656637
              ACF1
Training set 9.991774e-05
```

- d. Posteriormente se calculan los residuales al cuadrado del modelo *AR(1)* y se realiza la prueba *ArchTest* para comprobar los efectos *Arch*

```
ARCH LM-test; Null hypothesis: no ARCH effects

data: BaseEntrenamiento
Chi-squared = 423.47, df = 2, p-value < 2.2e-16
```

Como el valor *p* es cercano a cero, se rechaza la hipótesis nula y se concluye que existen efectos *Arch* en nuestra base de datos

- e. Se calcula el modelo usando finalmente *AR(1) + GARCH(1,1)*. Se probó hasta con dos rezagos para el modelo *GARCH* pero el anteriormente mencionado fue el que dio mejores resultados


```

*-----*
*          GARCH Model Fit          *
*-----*

Conditional Variance Dynamics
-----
GARCH Model      : SGARCH(1,1)
Mean Model       : ARFIMA(1,0,0)
Distribution      : norm

Optimal Parameters
-----
      Estimate  Std. Error  t value  Pr(>|t|)
mu      0.132366   0.026584   4.9792  0.000001
ar1     -0.047978   0.025236  -1.9012  0.057277
omega    0.288063   0.043617   6.6044  0.000000
alpha1   0.213378   0.029890   7.1387  0.000000
beta1    0.691620   0.032540  21.2547  0.000000

Robust Standard Errors:
      Estimate  Std. Error  t value  Pr(>|t|)
mu      0.132366   0.024002   5.5149  0.000000
ar1     -0.047978   0.031629  -1.5169  0.129300
omega    0.288063   0.090258   3.1916  0.001415
alpha1   0.213378   0.053070   4.0207  0.000058
beta1    0.691620   0.056998  12.1341  0.000000

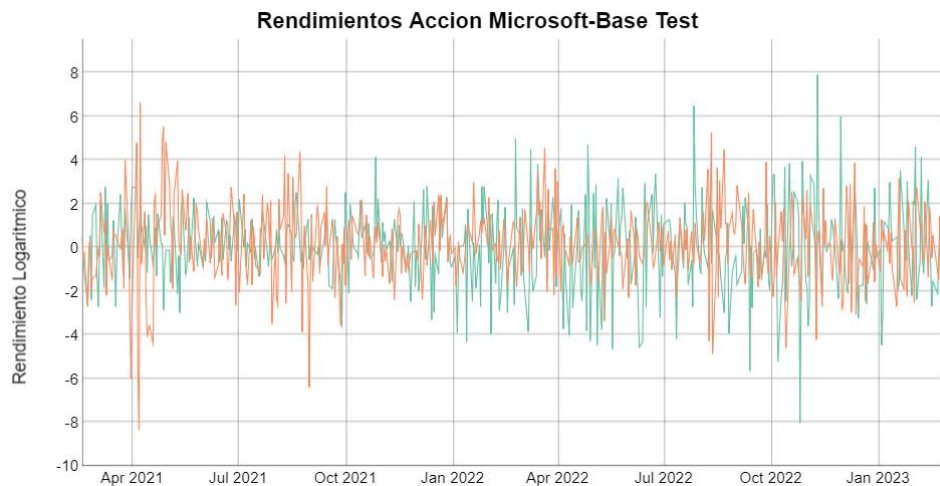
LogLikelihood : -3953.059

Information Criteria
-----
Akaike      3.5982
Bayes       3.6112
Shibata     3.5982
Hannan-Quinn 3.6030

```

Se observa que los valores para alpha1 y beta1 que significan respectivamente el valor del coeficiente al cuadrado y el coeficiente de la varianza al cuadrado son significativos.

- f. Finalmente se realiza la predicción de 500 valores y se realiza el analisis con la base de prueba. Gráficamente:



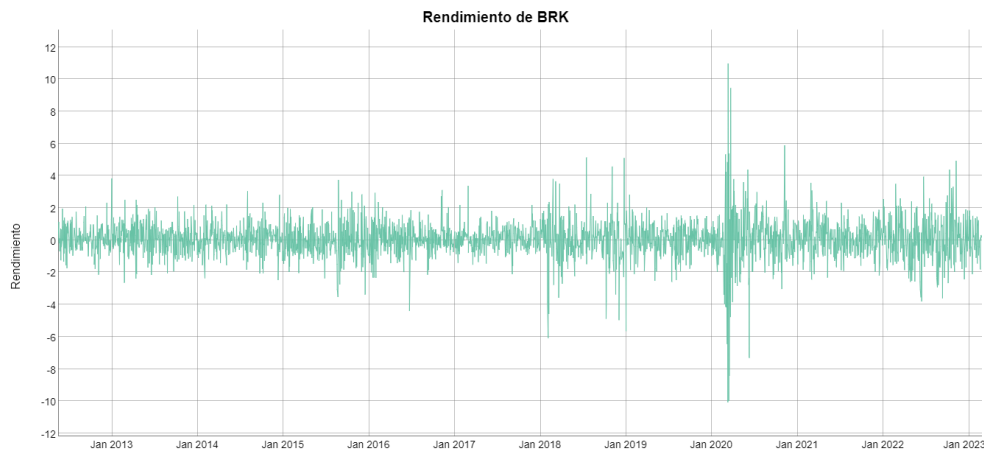
Donde la verde son los datos reales y la línea roja los pronosticados. Se observa que antes de octubre del 2021 la variabilidad del pronóstico es mucho mayor a los datos reales, pero posterior a esta fecha cambia el comportamiento. Calculando las métricas de RMSE, MAE, AIC, BIC y R^2 se obtiene:

Métrica	Valor
RMSE	2.5655

MAE	2.0297
AIC	3.5982
BAYES	3.6112

5.2 Predicción de la acción de BRK

- a) Para predecir los retornos logarítmicos porcentuales de la acción BRK, se procede de la misma manera que con la serie de Microsoft, es decir se va a utilizar un modelo GARCH. Gráficamente la serie se ve de la siguiente manera:



- b) Se toma el 20% de la información como conjunto de prueba y el resto, como la base de entrenamiento. Aproximadamente son 500 datos para el test.
- c) Se halla el modelo *ARMA* de la media por medio de la función *auto.arima()* de R. El cual arroja un *AR* (2) con media diferente de cero.

```
Series: BaseEntrenamiento
ARIMA(2,0,0) with non-zero mean

Coefficients:
      ar1      ar2      mean
    -0.1517  0.0536  0.0515
s.e.    0.0212  0.0212  0.0228

sigma^2 = 1.383; log likelihood = -3495.44
AIC=6998.87  AICC=6998.89  BIC=7021.68

Training set error measures:
              ME      RMSE      MAE  MPE  MAPE      MASE      ACF1
Training set -3.132561e-05  1.175002  0.7890054  NaN   Inf  0.6621919 -0.0001920435
```

- d) Se calculan los residuales al cuadrado del *AR*(2) y se realiza la prueba *ArchTest* para comprobar los efectos del *Arch*

```
ARCH LM-test; Null hypothesis: no ARCH effects

data: BaseEntrenamiento
Chi-squared = 698.95, df = 2, p-value < 2.2e-16
```

Como el valor *p* es cercano a cero, se rechaza la hipótesis nula y se concluye que existen efectos *Arch* para la serie

- e) Se calcula el modelo usando finalmente *AR* (2) +*GARCH* (1,1). Sin embargo, al visualizar el segundo parámetro del *AR* este no es significativo

```

*-----*
*          GARCH Model Fit          *
*-----*

Conditional Variance Dynamics
-----
GARCH Model      : sGARCH(1,1)
Mean Model       : ARFIMA(2,0,0)
Distribution      : norm

Optimal Parameters
-----

```

	Estimate	Std. Error	t value	Pr(> t)
mu	0.065922	0.018549	3.55387	0.000380
ar1	-0.073179	0.023700	-3.08773	0.002017
ar2	0.015437	0.023321	0.66194	0.508009
omega	0.083307	0.016058	5.18804	0.000000
alpha1	0.133128	0.018723	7.11043	0.000000
beta1	0.797860	0.026840	29.72617	0.000000

Por lo tanto, el modelo final es $AR(1) + GARCH(1,1)$

```

*-----*
*          GARCH Model Fit          *
*-----*

Conditional Variance Dynamics
-----
GARCH Model      : sGARCH(1,1)
Mean Model       : ARFIMA(1,0,0)
Distribution      : norm

Optimal Parameters
-----

```

	Estimate	Std. Error	t value	Pr(> t)
mu	0.065823	0.018288	3.5992	0.000319
ar1	-0.074542	0.023681	-3.1478	0.001645
omega	0.083190	0.016063	5.1790	0.000000
alpha1	0.133037	0.018714	7.1089	0.000000
beta1	0.798034	0.026854	29.7174	0.000000

Robust Standard Errors:

	Estimate	Std. Error	t value	Pr(> t)
mu	0.065823	0.020498	3.2112	0.001322
ar1	-0.074542	0.025334	-2.9423	0.003257
omega	0.083190	0.030849	2.6967	0.007003
alpha1	0.133037	0.036172	3.6779	0.000235
beta1	0.798034	0.053185	15.0050	0.000000

LogLikelihood : -3150.835

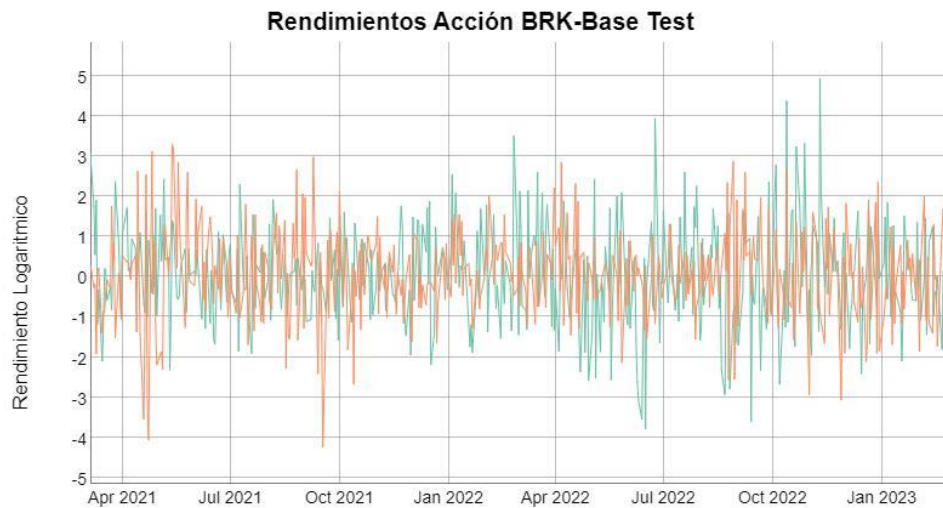
Information Criteria

```

-----
Akaike          2.8534
Bayes           2.8663
Shibata         2.8534
Hannan-Quinn    2.8581

```

- f) Finalmente se realiza la predicción de 500 valores y se realiza el análisis con la base de prueba. Gráficamente:



Calculando las métricas se obtiene:

Métrica	Valor
RMSE	1.6160
MAE	1.2725
AIC	2.8534
BAYES	2.8663

7. Implementación de regresión lineal para las predicciones de los Betas de las acciones

Para calcular los Betas se hace referencia al modelo de valoración de activos (CAPM) el cual permite estimar la rentabilidad esperada en función del riesgo sistemático. En nuestro proyecto se toma la ecuación del modelo CAPM la cual es:

$$r_t = \alpha + \beta r_{m,t} + e_t, \quad t = 1, 2 \dots T.$$

$r_{m,t}$: rendimiento o índice del mercado

r_t : rendimiento del activo en cuestión.

Y por medio de una regresión lineal simple se calcula el valor de α y β teniendo en cuenta como variable independiente los rendimientos logarítmicos del S&P500 y la variable dependiente va variando por el resto de los rendimientos logarítmicos de las demás acciones contempladas en este trabajo. El valor de α se refiere respectivamente a la valoración errónea del activo relativo al mercado, libre de riesgo, mientras que β indica la sensibilidad del riesgo del mercado. Calculando estos valores obtenemos:

	Accion	Alpha	Valor-p Alpha	Beta	Valor-p Beta
0	AAPL	0.032	0.202	1.170	0.0
1	AMZN	0.033	0.284	1.136	0.0
2	BRK-B	0.013	0.315	0.887	0.0
3	GOOG	0.020	0.382	1.123	0.0
4	GOOGL	0.019	0.389	1.127	0.0
5	JNJ	0.019	0.247	0.598	0.0
6	META	0.011	0.799	1.220	0.0
7	MSFT	0.037	0.067	1.196	0.0
8	NVDA	0.097	0.017	1.622	0.0
9	PG	0.016	0.366	0.589	0.0
10	TSLA	0.114	0.065	1.432	0.0
11	UNH	0.048	0.041	0.907	0.0

Con respecto al valor p se observa que todos los valores de β son significativos, pero esto no ocurre con los valores de α . Además, se puede observar que en general las acciones del portafolio se consideran riesgosas, excepto por JNJ que presenta un valor de beta bajo.

8. Recomendaciones sobre el modelo final seleccionado

Para los tres modelos principales, es decir para el Random Forest y los dos modelos GARCH se sugiere realizar la calibración de los parámetros de manera mensual, esto debido a la volatilidad y los cambios que ocurren en el mercado. Para el caso de los modelos autorregresivos con heterocedasticidad condicional que corresponden a la predicción de los rendimientos logarítmicos porcentuales de las acciones MSFT y BRK-B, se sugiere cada mes identificar los valores de los parámetros del modelo correspondiente a la media (ARMA) y a la varianza (GARCH) por medio del software R. Al encontrar los nuevos parámetros se deben modificar los archivos MSFTModel y BRKModel.

9. Prototipo Pronóstico del S&P 500 y del portafolio de acciones del grupo Stanley

• Infraestructura necesaria

Para la implementación, ejecución y despliegue del prototipo presentado al grupo Stanley es necesario contar con la siguiente infraestructura de hardware y software:

- ✓ Servidor virtual de Azure con las siguientes características:
 - ✓ Procesador Intel(R) Xeon(R) Platinum 8171M CPU @ 2.60GHz 2.10 GHz
 - ✓ RAM instalada 32.0 GB
 - ✓ Tipo de Sistema basado en 64 – bit
 - ✓ Windows 10 Enterprise 22H2, versión 19045.2965
- ✓ Licencia de PowerBI Pro para un usuario
- ✓ Python versión 3.9
- ✓ R Studio 4.3
- ✓ Anaconda Navigator



• Creación ambiente virtual en Anaconda

Para el desarrollo de los modelos necesarios y para ejecutar el código generado, es necesario crear un ambiente virtual en Anaconda. Las librerías necesarias y las versiones de cada una se muestran a continuación:





Package	Version		
-----	-----		
absl-py	1.4.0	numpy	1.23.5
alembic	1.10.4	oauthlib	3.2.2
anyio	3.6.2	opt-einsum	3.3.0
appdirs	1.4.4	optuna	3.1.1
argon2-cffi	21.3.0	packaging	23.1
argon2-cffi-bindings	21.2.0	pandas	1.5.3
arrow	1.2.3	pandas-datareader	0.10.0
asttokens	2.2.1	pandocfilters	1.5.0
astunparse	1.6.3	parso	0.8.3
attrs	23.1.0	patsy	0.5.3
backcall	0.2.0	pickleshare	0.7.5
beautifulsoup4	4.12.2	Pillow	9.5.0
bleach	6.0.0	pip	23.0.1
cachetools	5.3.0	platformdirs	3.5.1
certifi	2023.5.7	prometheus-client	0.16.0
cffi	1.15.1	prompt-toolkit	3.0.38
charset-normalizer	3.1.0	protobuf	4.23.0
cmaes	0.9.1	pure-eval	0.2.2
colorama	0.4.6	pyasn1	0.5.0
colorlog	6.7.0	pyasn1-modules	0.3.0
contourpy	1.0.7	pycparser	2.21
cryptography	40.0.2	Pygments	2.15.1
cycler	0.11.0	pyparsing	3.0.9
decorator	5.1.1	pyrsistent	0.19.3
defusedxml	0.7.1	python-dateutil	2.8.2
executing	1.2.0	python-json-logger	2.0.7
fastjsonschema	2.16.3	pytz	2023.3
flatbuffers	23.5.9	pytz-deprecation-shim	0.1.0.post0
fonttools	4.39.4	pywin32	306
fqdn	1.5.1	pywinpty	2.0.10
frozendict	2.3.8	PyYAML	6.0
gast	0.4.0	pyzmq	25.0.2
google-auth	2.18.0	requests	2.30.0
google-auth-oauthlib	1.0.0	requests-oauthlib	1.3.1
google-pasta	0.2.0	rfc3339-validator	0.1.4
greenlet	2.0.2	rfc3986-validator	0.1.1
grpcio	1.54.2	rpy2	3.5.5
h5py	3.8.0	rsa	4.9
html5lib	1.1	scikit-learn	1.2.2
idna	3.4	scipy	1.10.1
importlib-metadata	6.6.0	Send2Trash	1.8.2
importlib-resources	5.12.0	setuptools	66.0.0
ipykernel	5.5.6	six	1.16.0
ipython	7.34.0	skforecast	0.7.0
ipython-genutils	0.2.0	sniffio	1.3.0
isoduration	20.11.0	soupsieve	2.4.1
jax	0.4.10	SQLAlchemy	2.0.13
jedi	0.18.2	stack-data	0.6.2
Jinja2	3.1.2	statsmodels	0.14.0
joblib	1.2.0	tensorboard	2.12.3
jsonpointer	2.3	tensorboard-data-server	0.7.0
jsonschema	4.17.3	tensorflow	2.12.0
		tensorflow-estimator	2.12.0

jupyter_client	8.2.0	tensorflow-intel	2.12.0
jupyter_core	5.3.0	tensorflow-io-gcs-filesystem	0.31.0
jupyter-events	0.6.3	termcolor	2.3.0
jupyter_server	2.5.0	terminado	0.17.1
jupyter_server_terminals	0.4.4	threadpoolctl	3.1.0
jupyterlab-pygments	0.2.2	tinycss2	1.2.1
keras	2.12.0	tornado	6.3.1
kiwisolver	1.4.4	tqdm	4.64.1
libclang	16.0.0	traitlets	5.9.0
lxml	4.9.2	typing_extensions	4.5.0
Mako	1.2.4	tzdata	2023.3
Markdown	3.4.3	tzlocal	4.3
MarkupSafe	2.1.2	uri-template	1.2.0
matplotlib	3.7.1	urllib3	1.26.15
matplotlib-inline	0.1.6	wcwidth	0.2.6
mistune	2.0.5	webcolors	1.13
mkl-service	2.4.0	webencodings	0.5.1
ml-dtypes	0.1.0	websocket-client	1.5.1
multitasking	0.0.11	Werkzeug	2.3.4
nbclassic	1.0.0	wheel	0.38.4
nbclient	0.7.4	wrapt	1.14.1
nbconvert	7.4.0	yfinance	0.2.18
nbformat	5.8.0	zipp	3.15.0
nest-asyncio	1.5.6		
notebook	6.5.4		
notebook_shim	0.2.3		









- **Descripción general del proceso de implementación del prototipo**
- **Modelos necesarios para generar el pronóstico:** Para la generación del pronóstico de los rendimientos logarítmicos del S&P 500 y de las acciones del grupo Stanley se tienen 3 modelos principales:
 - ✓ Un modelo de Random Forest creado en el lenguaje Python, acompañado del escalador utilizado. Estos archivos se encuentran en la carpeta SP500 de la ubicación principal

SP500 > SP500	
Nombre	Estado
SP500Model	
SP500Scaler.gz	

- ✓ Un modelo GARCH creado en el lenguaje R que pronostica el rendimiento logarítmico de la acción de MSFT, necesario para el pronóstico del S&P 500, ubicado en la carpeta con el nombre MSFTStock

SP500 > MSFTStock	
Nombre	Estado
.ipynb_checkpoints	
delay	
MSFTModel	
MSFTModel.R	

- ✓ Un modelo GARCH creado en el lenguaje R que pronostica el rendimiento logarítmico de la acción de BRB-K, necesario para el pronóstico del S&P 500, ubicado en la carpeta con el nombre BRKStock

SP500 > BRKStock	
Nombre	Estado
 bk	
 BRBKModel	
 BRBKModel.R	
 delay	

- **Descripción del código principal de generación del pronóstico:** A continuación, se describe el código donde se utilizan los modelos anteriormente presentados, y la transformación de la data hasta generar los archivos planos utilizados en el Power BI.

```
from pandas_datareader import data as pdr
import yfinance as yfin
yfin.pdr_override()
import warnings
warnings.filterwarnings('ignore')
import numpy as np
import pandas as pd
import joblib
import pickle
import matplotlib.pyplot as plt
from pandas.tseries.offsets import BDay
from datetime import datetime, timedelta, date
import tensorflow as tf
import keras

## Librerías necesarias

from sklearn.linear_model import LinearRegression
import statsmodels.api as sm
from pathlib import Path
import sys
home=str(Path.home()) / 'OneDrive - Florexp@SP500'
sys.path.append(r'C:\Users\aparra\Anaconda3')
sys.path.append(r'C:\Users\aparra\Anaconda3\scripts')
sys.path.append(r'C:\Users\aparra\Anaconda3\Library\bin')
|
steps=7
delay=0
```

Para los modelos GARCH implementados en R, es necesario que el valor en *Delay* pase a un archivo de tipo csv, para posteriormente ejecutar estos modelos en Python.

La función `getMSFT()` ejecuta los modelos GARCH para MSFT y BRK, retorna un DataFrame que posteriormente se leerá en Python. Estos DataFrame pronostican los retornos logarítmicos según el step.

Se importan las librerías necesarias, incluidas las que permiten leer datos desde Yahoo Finance, y aquellas que permiten ejecutar los modelos necesarios, los cuales fueron mencionados anteriormente. Al tener un ambiente virtual dedicado al proyecto, se incluyen las rutas principales de Anaconda al modelo, para tener disponibles los archivos necesarios. *Steps* indica cuantos datos se desean pronosticar, mientras que *Delay* indica cuantos datos de días pasados se desea simular con el modelo.

```
delay_table=pd.read_csv(home+"MSFTStock\delay.csv")
delay_table.iat[0,0]=delay
delay_table.to_csv(home+"MSFTStock\delay.csv",index=False)

delay_table=pd.read_csv(home+"BRKStock\delay.csv")
delay_table.iat[0,0]=delay
delay_table.to_csv(home+"BRKStock\delay.csv",index=False)

import os
os.environ["R_HOME"] = r"C:\Program Files\R\R-4.3.0" # change as needed
import rpy2.robj as robj
from rpy2.robj import pandas2ri
import rpy2.robj as ro
pandas2ri.activate()

def getMSFT():
    # Cargar el archivo "garch.R" en R
    robj.r['source'](home+"MSFTStock\MSFTModel.R")

    # Acceder a la función "mi_funcion" de R en Python
    mi_funcion = robj.r['MSFT']
    MSFT=mi_funcion()
    MSFT = ro.conversion.rpy2py(MSFT)

    robj.r['source'](home+"BRKStock\BRBKModel.R")

    # Acceder a la función "mi_funcion" de R en Python
    mi_funcion = robj.r['BRBK']
    BRBK=mi_funcion()
    BRBK = ro.conversion.rpy2py(BRBK)

    return MSFT,BRBK
pred_msft,pred_brk=getMSFT()
```

El siguiente código muestra la carga de los archivos necesarios para ejecutar el Random Forest, teniendo en cuenta las predicciones de las acciones del MSFT y BRK-B usando el escalador y

obteniendo como resultado la predicción de los retornos logarítmicos porcentuales de los siguientes días de acuerdo con la variable Step.

```
with open(home+'\\SP500\\SP500Scaler.gz', 'rb') as f:
    scaler = joblib.load(f)

with open(home+'\\SP500\\SP500Model', 'rb') as f:
    modelo = pickle.load(f)

columnas_scaler = ['GSPC.ReturnSuavizado', 'AAPL.ReturnSuavizado', 'AMZN.ReturnSuavizado',
'MSTF.ReturnSuavizado', 'TSLA.ReturnSuavizado', 'GOOG.ReturnSuavizado',
'GOOGL.ReturnSuavizado', 'NVDA.ReturnSuavizado',
'BRK.B.ReturnSuavizado', 'META.ReturnSuavizado', 'UNH.ReturnSuavizado',
'JNJ.ReturnSuavizado', 'PG.ReturnSuavizado', 'VIX.ReturnSuavizado',
'DolarIndex.ReturnSuavizado']

df2 = pd.DataFrame(0, index=df.index, columns=columnas_scaler)
df2['BRK.B.ReturnSuavizado'] = df['BRK.B']
df2['MSTF.ReturnSuavizado'] = df['MSTF']
df2 = pd.DataFrame(scaler.transform(df2), columns=columnas_scaler, index=df.index)
exog = df2[['MSTF.ReturnSuavizado', 'BRK.B.ReturnSuavizado']]

cur_date=date.today()- BDay(delay)
print(cur_date)
initial_date_SP=cur_date - BDay(20)

initial_date_SP=initial_date_SP.strftime('%Y-%m-%d')
print(initial_date_SP)

data_GSPC = pd.DataFrame(pdr.get_data_yahoo("^GSPC", initial_date_SP, cur_date)['Adj Close'])
log_returns=np.log(data_GSPC).diff()*100
log_returns=log_returns[1:]
log_returns.columns = ['GSPC']

yest_day=cur_date - BDay(1)
initial_date=cur_date - BDay(5)
final_date=cur_date + BDay(steps-1)
last_window_calendar=pd.bdate_range(start=initial_date,end=yest_day)
next_days_calendar=pd.bdate_range(start=cur_date,end=final_date)
if next_days_calendar[0]!=cur_date:
    final_date=final_date+BDay(1)
    next_days_calendar=pd.bdate_range(start=cur_date,end=final_date)
last_window=log_returns.tail(5)
print(last_window)
print(last_window_calendar)
last_window.index = last_window_calendar
exog.index =next_days_calendar
df.index =next_days_calendar
last_window=last_window.iloc[:,0]
df['ForecastDate']=cur_date.strftime('%Y-%m-%d')
df.to_csv(home+"\\ResultFiles\\Back_Test_Stocks.csv", mode='a', header=False, index=True)
print("ok antes de modelo")
pred = pd.DataFrame(modelo.predict_interval(exog=exog,steps=steps,last_window=last_window))
```

El resultado obtenido en el paso anterior se vuelve a transformar a su escala original utilizando el *inverse_transform* del escalador, arrojando como resultado el pronóstico de los retornos logarítmicos, los límites superiores e inferiores y las predicciones 20 días atrás a la fecha actual, el cual se agrega a un archivo csv.

```
pred = pred.reset_index().iloc[:,1:]

pred_final = pd.DataFrame(0, index=pred.index, columns=columnas_scaler)
pred_final['GSPC.ReturnSuavizado'] = pred['pred']
#Matriz para intervalo inferior
predlow_final = pd.DataFrame(0, index=pred.index, columns=columnas_scaler)
predlow_final['GSPC.ReturnSuavizado'] = pred['lower_bound']
#Matriz para intervalo superior
predup_final = pd.DataFrame(0, index=pred.index, columns=columnas_scaler)
predup_final['GSPC.ReturnSuavizado'] = pred['upper_bound']

#Transformamos los datos a la representación original.
pred_return = pd.DataFrame(scaler.inverse_transform(pred_final, copy=None))[0]
predlow_return = pd.DataFrame(scaler.inverse_transform(predlow_final, copy=None))[0]
predup_return = pd.DataFrame(scaler.inverse_transform(predup_final, copy=None))[0]

#Construimos un dataframe con las predicciones en la representación original
pred = pd.concat([pred_return,predlow_return ,predup_return],axis=1)
pred.columns = ["pred","lower_bound","upper_bound"]
pred.index=pd.bdate_range(start=cur_date,end=final_date)
pred['ForecastDate']=cur_date.strftime('%Y-%m-%d')

pred.to_csv(home+"\\ResultFiles\\Back_Test.csv", mode='a', header=False, index=True)
```

Posteriormente se hallan los valores Beta que relacionan cada una de las acciones con el indicador S&P 500 en cuanto a sus retornos logarítmicos porcentuales. Estas betas ayudarán a pronosticar los retornos esperados por cada acción dentro y fuera del portafolio. Finalmente se

guarda estos resultados, más los precios de cierre ajustado de las acciones y los valores pronosticados en la ventana definida.

```
###Valores de alpha, valor p alpha , beta, valor p beta

Accion=[]
Alpha=[]
ValorPAlpha=[]
Beta=[]
ValorPBeta=[]

for i in range(0,(len(log_returns_acciones.columns)-1)):
    X=log_returns_acciones.iloc[:,12]
    X= sm.add_constant(X, prepend=True)
    modelo = sm.OLS(endog=log_returns_acciones.iloc[:,i], exog=X)
    modelo = modelo.fit()

    Accion.append(log_returns_acciones.columns[i])
    Alpha.append(round(modelo.params[0],3))
    Beta.append(round(modelo.params[1],3))
    ValorPAlpha.append(round(modelo.pvalues[0],3))
    ValorPBeta.append(round(modelo.pvalues[1],3))

result = list(zip(Accion,Alpha, ValorPAlpha, Beta,ValorPBeta))
betas = pd.DataFrame(result)
betas = betas.rename(columns={0:'Accion',1:'Alpha',2:'Valor-p Alpha',3:'Beta',4:'Valor-p Beta' })

pred['stock']='S&P500'

New_Stocks=[]
for i in range(betas.shape[0]):
    for j in range(pred.shape[0]):
        new_line=[]
        beta_temp=betas.Beta.iloc[i]
        stock_temp=betas.Accion.iloc[i]
        new_line.append(pred.index[j])
        new_line.append(pred.pred[j]*betas.Beta[i])
        new_line.append(pred.lower_bound[j]*betas.Beta[i])
        new_line.append(pred.upper_bound[j]*betas.Beta[i])
        new_line.append(stock_temp)
        New_Stocks.append(new_line)

New_Stocks=pd.DataFrame(New_Stocks, columns=['Date','pred','lower_bound','upper_bound','Stock'])
New_Stocks.set_index('Date',inplace=True)

log_returns_acciones=log_returns_acciones[log_returns_acciones.index>=min(log_returns.index)]

log_returns_acciones.to_csv(home+r"\ResultFiles\historico_acciones.csv",index=True)
New_Stocks.to_csv(home+r"\ResultFiles\futuro_acciones.csv",index=True)
betas.to_csv(home+r"\ResultFiles\betas.csv",index=True)
acciones.to_csv(home+r"\ResultFiles\precio_acciones.csv",index=True)
```

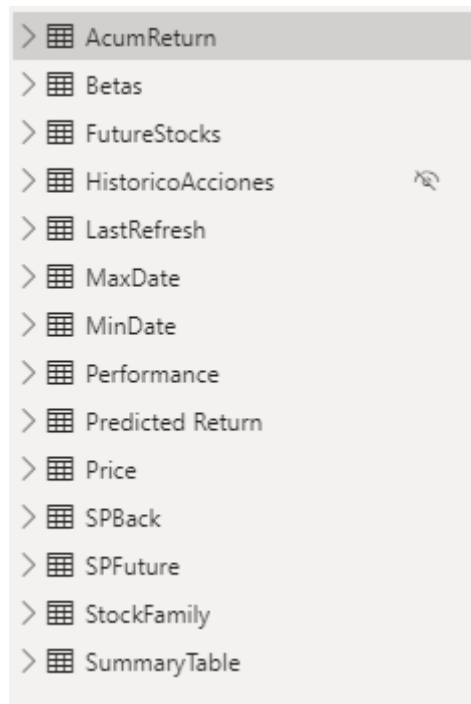
El resultado final son archivos planos csv, que serán consumidos por el Power BI.

- **Actualización programada de los nuevos pronósticos:** Para que la mesa de dinero cuente con información actualizada al inicio del día, se crea una tarea programada en Windows que correrá automáticamente todos los días a las 4:30 am PST. La tarea toma alrededor de 5 minutos en terminar, y se programa a través de un archivo .bat que ejecuta la instancia de Python dentro del ambiente virtual creado e invoca el código descrito anteriormente

Name	Status	Triggers	Next Run Time	Last Run Time	Last Run Result	Author	Created
SP500	Ready	At 4:30 AM every day	5/25/2023 4:30:00 AM	5/24/2023 4:30:01 AM	The operation completed successfully. (0x0)		5/2/2022 10:11:39 PM

- **Creación del modelo de Power BI:** Para la visualización de la información por parte de la mesa de dinero, o de los clientes del grupo Stanley, se genera un reporte en Power BI. Este reporte tiene como componentes las tablas que desagregan la información proveniente

de los archivos planos csv con el propósito de una correcta visualización de los datos. Una copia del modelo de Power BI se encuentra ubicado en el GitHub del proyecto.



El prototipo visualmente cuenta con los siguientes elementos:

- ✓ Gráfico principal donde se evidencia el histórico de los retornos logarítmicos del S&P 500, al igual que las predicciones a un día hechas en el pasado, con sus respectivos límites de confianza.
- ✓ Fecha y hora de la última actualización del reporte con UTC Colombia, para que tanto el analista como el cliente del grupo tenga conocimiento de que la información se encuentra actualizada.
- ✓ Como métrica de desempeño se muestra, de los últimos 20 días, cual es el porcentaje de días donde el histórico y la predicción coinciden en la dirección del retorno, es decir que ambos son positivos y negativos.
- ✓ Se observa el rendimiento logarítmico del S&P 500 para el siguiente día predicho, es decir el día actual.
- ✓ Se visualiza el rendimiento logarítmico acumulado del S&P 500 para el horizonte de los siguientes 7 días.
- ✓ Existe un selector donde se pueden observar las acciones dentro y fuera del portafolio. Este selector afecta únicamente la gráfica inferior del reporte.
- ✓ La tabla de resumen muestra para cada acción el precio de cierre ajustado real para el último día, el retorno logarítmico del siguiente día pronosticado, el retorno logarítmico acumulado de los siguientes 7 días, el beta de cada acción respecto al S&P 500, y una sugerencia de acuerdo con el perfil del cliente que se esté analizando, ya sea conservador o riesgoso. Esta sugerencia se hace basado en el valor de beta y en el retorno logarítmico acumulado de los siguientes 7 días, que se resume en la siguiente tabla:

	Valor de Ret.		
Valor de Beta	Acum	Conservador	Arriesgado

Beta > 1.8	>= 2%	Mantener	Comprar
	>0%	Mantener	Mantener
	<0%	Mantener	Vender
	<= -2%	Vender	Vender
Beta >1 & < 1.8	>= 2%	Compra	Compra
	>0%	Mantener	Compra
	<0%	Mantiene	Vender
	<= -2%	Vender	Vender
Beta < 1	>= 2%	Compra	Compra
	>0%	Compra	Compra
	<0%	Vender	Mantiene
	<= -2%	Vender	Vender

- **Ingreso a la información por parte del usuario final:** para que la información pueda ser consultada de forma segura ya sea por los analistas de la mesa de dinero o por los clientes del grupo, se crea una app web en la cual será necesario un usuario y contraseña para ingresar, y será la única forma de acceder al reporte de Power BI. Esta app web esta creada en un servidor de pruebas con un dominio gratuito. El lenguaje utilizado es PHP para funcionalidades del login y conexión con base de datos MySQL donde estarán alojados los usuarios, HTML para estructura del reporte y de la app, y CSS para estilos. Este servidor actualmente se utiliza para pruebas con la finalidad de migrar a un servidor propio.

10. Cumplimiento tabla de requerimientos

Aspecto	Requerimiento	Prueba Prevista	Métrica de evaluación	Se cumplió	¿Por qué?
Negocio					
R1	Impacto en el margen bruto del área de mesa de dinero	Valoración del margen bruto actual en comparación del margen bruto generado 1 año después de la implementación de la herramienta	Pasar de un margen bruto de -20% a 15%	Espera	En caso de que los analistas de la Mesa del Dinero tomen las sugerencias dadas por la herramienta, se tomaran mejores decisiones que mejorarán el margen bruto y la rentabilidad de la empresa a un año
R2	Impacto en la rentabilidad operativa del área de mesa de Dinero	Valoración de la rentabilidad actual en comparación a la rentabilidad generada 1 año después de la implementación de la herramienta	Pasar de una rentabilidad de -17% a 7%		

R3	Impacto en el índice de satisfacción de los clientes finales del área de mesa de Dinero	Comparación entre el índice CSAT actual en comparación con el índice CSAT 1 año después de la implementación de la herramienta	Pasar de un índice CSAT del 55% a un índice del 90%		La herramienta permitirá que la respuesta de los analistas de la Mesa de Dinero hacia los clientes sobre las inversiones que deben realizarse sea oportuna, confiable y segura. Además la herramienta cuenta con un diseño amigable, lo cual mejorará el índice de satisfacción de los clientes.
Desempeño					
R3	Estabilidad y disponibilidad del servicio de la herramienta	Prueba de consultas múltiples al servidor donde estará alojada la herramienta, entendiendo el nivel de carga y requisitos de infraestructura	Nivel de servicio mayor al 95%	Cumple	La herramienta se compone de un servidor gratis y una licencia paga, donde se realizaron pruebas durante las últimas tres semanas mostrando que la velocidad de respuesta del servidor web, la rapidez de la carga del reporte y la actualización de la información se realiza de manera satisfactoria
R4	Desarrollo de modelo predictivo que permita encontrar los valores de los Betas por acción comprada con el S&P 500	Disponibilidad de datos para entramiento, testing y validación de la información	RSME con un intervalo de confianza frente a los valores encontrados en la literatura para el mismo problema entre +- 25%	Cumple	El valor p de los betas para cada una de las acciones que se relaciona con el SP500 fue significativo

R5	Desarrollo de modelo predictivo que permita encontrar los rendimientos logarítmicos del S&P 500 a futuro en el corto y mediano plazo	Disponibilidad de datos para entramiento, testing y validación de la información	RSME con un intervalo de confianza frente a los valores encontrados en la literatura para el mismo problema entre +- 25%	Cumple	El modelo principal Forecaster Autoregresor de Skforeraster con Random Forest, logró un R2 del 85% y RMSE del 0.07 para las pruebas realizadas en el conjunto de prueba
Funcional					
R6	Visualización de la herramienta amigable con el usuario final	Pruebas piloto con los analistas de la mesa de dinero	Nivel de satisfacción mayor al 90% sobre el total de pruebas piloto	Cumple	Después de tener la retroalimentación sobre la visualización de la herramienta, se realizaron las modificaciones pertinentes para lograr mostrar la información de manera más amigable
R7	Visualización de la serie del S&P 500 con valores históricos, y proyecciones de corto y mediano plazo	Desarrollo del mockup inicial, investigación de la posibilidad en el software definido PowerBI	Que se encuentre en la herramienta final	Cumple	Se logró desarrollar el reporte de análisis con la metodología propuesta
R8	Análisis de sensibilidad del S&P 500 frente al cambio en las variables macroeconómicas	Modificación de las variables macroeconómicas en la herramienta, visualización del ajuste en los datos del S&P 500 y de los valores futuros de las acciones	Que se encuentre en la herramienta final	No cumple	Finalmente no se tuvo en cuenta las variables macroeconómicas, ya que no mejoraban el desempeño de los modelos
R9	Sugerencia a los usuarios analistas de la mesa de Dinero sobre que acciones tomar en el corto y mediano plazo	Resumen de las decisiones a tomar frente a las acciones y el portafolio actual de la mesa de dinero	Que sea posible ver en la herramienta que acciones deben venderse o comprarse de acuerdo a los reflejado en los precios futuros, y con qué velocidad	Cumple	De acuerdo con el perfil del cliente se realizan sugerencias de compra, venta, mantener u omitir las acciones

R10	Actualización diaria de la herramienta donde se visualice los valores históricos y la respectiva proyección	Realización de pruebas piloto en donde se visualice en la herramienta la actualización diaria de la información, mostrando al inicio de cada día el nuevo valor de la proyección y el valor histórico del día anterior	Que se encuentre en la herramienta final	Cumple	Se desarrolló la infraestructura necesaria para actualizar la información de manera diaria, confiable y estable
-----	---	--	--	--------	---

11. Bibliografía

- Giraldo Picón, E. L. Pronóstico de volatilidades a los rendimientos de activos financieros de renta variable en Colombia a través de modelos ARCH y GARCH.
- García-Medina, A., & Aguayo-Moreno, E. (2023). LSTM–GARCH Hybrid Model for the Prediction of Volatility in Cryptocurrency Portfolios. *Computational Economics*, 1-32.
- Manthani, S. (n.d.). *Predicting SP500 index price.ipynb at master · sathishmanthani/predict-sp500-index*.
- Rodrigo, J. A., & Ortiz, J. E. (n.d.). *Welcome to skforecast*. Skforecast.org. Retrieved May 26, 2023, from <https://skforecast.org/0.8.0/index.html>
- Filho, M. (2023, March 7). Multi-step time series forecasting in python. Forecastegy.com. <https://forecastegy.com/posts/multi-step-time-series-forecasting-in-python/>