
SKILLING
CENTER
TECMILENIO

SKILLING CENTER TECMILENIO

Aplicaciones Financieras

Modelos multifactoriales

Introducción

Como inversionistas en el mercado de valores, se tiene una gran variedad de acciones para seleccionar como parte del portafolio de inversión. A simple vista, todas parecen ofrecer atractivos rendimientos, sin embargo, la diferencia de estos rendimientos está en el nivel de riesgo que tiene implícito cada una. En este tema conocerás la forma de calcular este riesgo y, por ende, el rendimiento esperado de la inversión.

Explicación

Modelo de fijación de precios de activos de capital (Capital Assets Pricing Model)

- Desarrollado por William Sharpe (Ming, 2021).
- Permite estimar la rentabilidad esperada en función del riesgo de mercado o sistemático.
- Se basa en la relación de la variación del precio de un activo contra otro, el más común es el índice del mercado, como *benchmark*.
- Cuando la variación de ambos activos es similar, se interpreta como una sincronía entre la empresa y el mercado, lo cual significa menor riesgo. Por el contrario, cuando la variación del activo es diferente a la del mercado, se interpreta como mayor riesgo.

Explicación

- El modelo básico CAPM contempla los factores de la tasa de interés libre de riesgo y los excedentes de rendimientos sobre el mercado.
- La tasa libre de riesgo se refiere a una tasa de interés asegurada sin riesgo al vencimiento del instrumento financiero, por ejemplo, los Bonos o Certificados de la Tesorería (Cetes).
- El propósito es determinar el rendimiento esperado de un activo.
- Se representa de la siguiente forma:

$$re = r_f + \beta(r_m - r_f)$$

En donde *re* es el rendimiento esperado, r_f es la tasa libre de riesgo, r_m es el rendimiento del mercado y β es el nivel de riesgo que incrementa o disminuye el gap de rendimiento entre el mercado y la tasa libre de riesgo. Mayores valores de β significa que se debe incrementar la prima de riesgo para determinar el rendimiento esperado.

- El cálculo se realiza dividiendo la covarianza de los rendimientos diarios del activo financiero y los rendimientos diarios del mercado, entre la varianza de los rendimientos del mercado, sobre un periodo de tiempo especificado (Kenton, 2022).

$$\beta = Cov(r_s, r_m)/Var(r_m)$$

En donde r_s son los rendimientos diarios del activo financiero y r_m son los rendimientos diarios del mercado.

El factor β se usa para ayudar a los inversionistas a entender cuando las fluctuaciones de un activo financiero se mueven en dirección relativa al resto del mercado (Kenton, 2022).

Explicación

Para el ejemplo se toma la acción de la empresa Moderna Inc. (MRNA), empresa del sector salud especializada en Biotecnología, junto con el índice Nasdaq de Estados Unidos (^IXIC), en donde cotiza sus acciones, para un periodo del 1 de enero del 2019 al 28 de febrero del 2023:

```
# importacion de las librerias
import pandas as pd
import yfinance as yf
import statsmodels.api as sm

# especificar la accion, el indice Nasdaq y el periodo de analisis
RISKY_ASSET = 'MRNA'
MARKET_BENCHMARK = '^IXIC'
START_DATE = '2019-01-01'
END_DATE = '2023-02-28’

# descarga de valores historicos de Yahoo Finance
df = yf.download([RISKY_ASSET, MARKET_BENCHMARK],
                 start=START_DATE,
                 end=END_DATE,
                 progress=False)

# cambiando los valores diarios a mensuales y calculando los rendimientos
# como el porcentaje de cambio mensual
X = df['Adj Close'].rename(columns={RISKY_ASSET: 'asset',
                                    MARKET_BENCHMARK: 'market'}) \
                   .resample('M') \
                   .last() \
                   .pct_change() \
                   .dropna()
```

Explicación

Continuación del código Python para determinar el modelo CAPM básico:

```
# calculando el valor de beta
covariance = X.cov().iloc[0,1]
benchmark_variance = X.market.var()
beta = covariance / benchmark_variance

# determinar los parametros del modelo CAPM
y = X.pop('asset')
X = sm.add_constant(X)
capm_model = sm.OLS(y, X).fit()
print(capm_model.summary())
```

Date	MRNA	^IXIC
2/21/2023	160.089996	11492.2998
2/22/2023	158.169998	11507.0703
2/23/2023	147.570007	11590.4004
2/24/2023	139.259995	11394.9404
2/27/2023	138.270004	11466.9805

Últimos registros descargados de precios históricos.

Date	asset	market
10/31/2022	0.27129	0.039008
11/30/2022	0.170159	0.04367
12/31/2022	0.02109	-0.087332
1/31/2023	-0.01982	0.106824
2/28/2023	-0.214643	-0.010149

Últimos registros de porcentajes de cambio mensual.

Explicación

El resumen del modelo para determinar el valor de β se muestra en el siguiente cuadro. β=1.7103 está indicado como la variable “market”, lo que significa que la acción de Moderna es 71 % más volátil que el mercado (índice Nasdaq).

- **# calculo del rendimiento esperado, con una**
  # supuesta tasa libre de riesgo del 3.0%
- **rm = X['market'].mean() * 12 * 100**
- **rf = 3.0**
- **re = rf + (beta * (rm - rf))**
- **print("valor de beta : ", beta)**
- **print("rendimiento del mercado = ", rm)**
- **print("rendimiento esperado MRNA = ",re)**

valor de beta: 1.7103229091117569
rendimiento del mercado = 13.62053615188069
rendimiento esperado MRNA = 21.164546287611167

OLS Regression Results

Dep. Variable:	asset	R-squared:	0.152
Model:	OLS	Adj. R-squared:	0.134
Method:	Least Squares	F-statistic:	8.407
Date:	Fri, 24 Mar 2023	Prob (F-statistic):	0.00567
Time:	23:57:17	Log-Likelihood:	-3.1638
No. Observations:	49	AIC:	10.33
Df Residuals:	47	BIC:	14.11
Df Model:	1		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	0.0561	0.038	1.467	0.149	-0.021	0.133
market	1.7103	0.590	2.899	0.006	0.524	2.897

Omnibus:	19.233	Durbin-Watson:	2.338
Prob(Omnibus):	0.000	Jarque-Bera (JB):	32.369
Skew:	1.173	Prob(JB):	9.36e-08
Kurtosis:	6.217	Cond. No.	15.7

Esta pantalla se obtuvo directamente del software que se está explicando en la computadora, para fines educativos.

Por lo tanto, el rendimiento esperado de una inversión en la acción de Moderna, con base al periodo de análisis y tomando como referencia el índice Nasdaq, es de 21.16% anual.

Explicación

Modelo CAPM de tres factores

- El modelo de tres factores amplía el modelo original con dos factores adicionales.
- Introduce el factor diferenciación del tamaño de empresas (SMB, *small minus big*), denominado *book equity* (BE) y la diferenciación de precios contables con los precios en el mercado (HML, *high minus low*), denominado como *book to market* (BM) (Fama y French, 1992).
- El modelo CAPM de tres factores se representa en la siguiente fórmula:

$$re = r_f + \beta_1(r_m - r_f) + \beta_2(SMB) + \beta_3(HML) + \epsilon$$

En donde β_1 es el factor de correlación de los rendimientos del mercado, β_2 es el factor de correlación las diferencias de tamaño y β_3 es el factor de correlación de las diferencias del *book to market*.

- Los portafolios de acciones del mercado son seleccionados continuamente por Fama-French de una muestra de acciones en el mercado, para luego hacer la separación de tamaños y precios, y los comparten en la página web de Kenneth French, que más adelante se utilizará en código Python.

Explicación

Desarrollo del código Python para determinar el modelo CAPM de tres factores:

```
# importar librerias requeridas
import pandas as pd
import yfinance as yf
import statsmodels.api as sm
import getFamaFrenchFactors as gff

# seleccionar el ticker de la empresa Wells Fargo Bank (WFC) y descargar de Yahoo Finance
ticker = 'wfc'
start = '2020-01-01'
end = '2023-01-31'
stock_data = yf.download(ticker, start, end)

# obtener los datos benchmark de Fama-French
ff3_monthly = gff.famaFrench3Factor(frequency='m')
ff3_monthly.rename(columns={"date_ff_factors": 'Date'}, inplace=True)
ff3_monthly.set_index('Date', inplace=True)

# calculo de rendimientos mensuales de la accion y juntarlo con el dataframe de Fama-French
stock_returns = stock_data['Close'].resample('M').last().pct_change().dropna()
stock_returns.name = "Month_Rtn"
ff_data = ff3_monthly.merge(stock_returns,on='Date')
# calculo de los coeficientes beta1, beta2 y beta3
X = ff_data[['Mkt-RF', 'SMB', 'HML']]
y = ff_data['Month_Rtn'] - ff_data['RF']
X = sm.add_constant(X)
ff_model = sm.OLS(y, X).fit()
print(ff_model.summary())
```

Explicación

El resultado de los valores de las tres betas se indica con las variables “mkt”= 1.1069, “smb”= 0.2274 y “hml”= 1.0261.

Date	Open	High	Low	Close	Adj Close	Volume
1/24/2023	44.91	45.200001	44.369999	44.450001	42.800453	13697200
1/25/2023	44.049999	45.419998	44.029999	45.34	43.657413	17285300
1/26/2023	45.529999	45.900002	45.16	45.810001	44.109978	16724600
1/27/2023	45.810001	46.369999	45.810001	46.119999	44.40847	18161800
1/30/2023	45.91	46.650002	45.810001	46.290001	44.572163	19427400

Tabla parcial de los precios históricos.

Date	Open	High	Low	Close	Adj Close
9/30/2022	-0.0935	-0.0081	0.0005	0.0019	-0.079844
10/31/2022	0.0783	0.0006	0.0801	0.0023	0.143461
11/30/2022	0.046	-0.0352	0.0138	0.0029	0.042618
12/31/2022	-0.0641	-0.0069	0.0137	0.0033	-0.138895
1/31/2023	0.0665	0.0501	-0.0401	0.0035	0.121095

Tabla con los parámetros para los 3 factores.

Explicación

Por último, se calcula el rendimiento esperado:

- **# calculo del rendimiento esperado de la accion**
- **intercept, b1, b2, b3 = ff_model.params**
- **rf = ff_data['RF'].mean()**
- **market_premium = ff3_monthly['Mkt-RF'].mean()**
- **size_premium = ff3_monthly['SMB'].mean()**
- **value_premium = ff3_monthly['HML'].mean()**
- **expected_monthly_return = rf + b1 * market_premium + b2 * size_premium + b3 * value_premium**
- **expected_yearly_return = expected_monthly_return * 12 * 100**
- **print("Rendimiento anual esperado : " + str(expected_yearly_return), "%")**

Rendimiento anual esperado: 15.078960006471318 %

resultados del modelo CAPM de tres factores para la accion : WFC

OLS Regression Results

Dep. Variable:	excess_rtn	R-squared:	0.673
Model:	OLS	Adj. R-squared:	0.642
Method:	Least Squares	F-statistic:	21.94
Date:	Sun, 26 Mar 2023	Prob (F-statistic):	6.60e-08
Time:	13:40:45	Log-Likelihood:	48.464
No. Observations:	36	AIC:	-88.93
Df Residuals:	32	BIC:	-82.59
Df Model:	3		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	-0.0086	0.011	-0.759	0.453	-0.032	0.014
mkt	1.1069	0.185	5.979	0.000	0.730	1.484
smb	0.2274	0.390	0.583	0.564	-0.567	1.022
hml	1.0261	0.217	4.726	0.000	0.584	1.468

Omnibus:	0.577	Durbin-Watson:	2.108
Prob(Omnibus):	0.749	Jarque-Bera (JB):	0.658
Skew:	0.089	Prob(JB):	0.720
Kurtosis:	2.362	Cond. No.	35.6

Esta pantalla se obtuvo directamente del software que se está explicando en la computadora, para fines educativos.

Explicación

Modelo CAPM con cuatro factores

- El modelo Fama-French ha tenido diversas modificaciones por diferentes autores.
- El cuarto factor adicionado por Mark Carhart se denomina como el *factor momentum* y se refiere al ritmo o velocidad de los cambios del precio del activo.
- El *factor momentum* (WML, *winners minus losers*) introduce la correlación lineal de los rendimientos de las acciones ganadoras y las perdedoras (Lewinson, 2020).
- El modelo se expresa de la siguiente forma:

$$re = r_f + \beta_1(r_m - r_f) + \beta_2(SMB) + \beta_3(HML) + \beta_4(WML) + \epsilon$$

- A continuación, se desarrolla el modelo de cuatro factores en Python para la acción de Amazon (AMZN):

```
# importar librerias
import pandas as pd
import yfinance as yf
import statsmodels.formula.api as smf
import pandas_datareader.data as web

# establecer la accion y el periodo de tiempo para el analisis
RISKY_ASSET = 'AMZN'
START_DATE = '2018-02-28'
END_DATE = '2023-02-28’
```

Explicación

Continuación del código Python para el modelo CAPM de cuatro factores:

```
# descargar los precios históricos de la accion de Yahoo Finance
asset_df = yf.download(RISKY_ASSET,
                       start=START_DATE,
                       end=END_DATE)

# calcular los retornos mensuales
y = asset_df['Close'].resample('M') \
                     .last() \
                     .pct_change() \
                     .dropna()
y.index = y.index.strftime('%Y-%m')
y.name = 'return'

# tres factores
df_three_factor = web.DataReader('F-F_Research_Data_Factors',
                                 'famafrench', start=START_DATE)[0]
df_three_factor.index = df_three_factor.index.format()

# descargar el factor momentum
df_mom = web.DataReader('F-F_Momentum_Factor', 'famafrench',
                        start=START_DATE)[0]
df_mom.index = df_mom.index.format()
```

Explicación

Continuación del código Python para el modelo CAPM de cuatro factores:

```
# integrar los factores para el modelo con cuatro factores
four_factor_data = df_three_factor.join(df_mom).join(y)

# renombrar columnas
four_factor_data.columns = ['mkt', 'smb', 'hml', 'rf', 'mom',
                            'rtn']

# dividir entre 100, excepto rendimientos
four_factor_data.loc[:, four_factor_data.columns != 'rtn'] /= 100

# obtener el periodo de analisis
four_factor_data = four_factor_data.loc[START_DATE:END_DATE]

# calcular diferencias de rendimientos
four_factor_data['excess_rtn'] = four_factor_data.rtn - four_factor_data.rf

# estimar el modelo con cuatro factores
four_factor_model = smf.ols(formula='excess_rtn ~ mkt + smb + hml + mom',
data=four_factor_data).fit()
print(four_factor_model.summary())
```

Explicación

A continuación, se muestra parcialmente el dataframe (four_factor_model) generado:

Inde	mkt	smb	hml	rf	wml	rtn	excess_rtn
2023-01	0.0665	0.0503	-0.0408	0.0035	-0.1598	0.227738	0.224238
2022-12	-0.0641	-0.0068	0.0132	0.0033	0.0452	-0.129894	-0.133194
2022-11	0.046	-0.034	0.0138	0.0029	-0.0201	-0.0575947	-0.0604947
2022-10	0.0783	0.0009	0.0805	0.0023	0.0387	-0.0934513	-0.0957513
2022-09	-0.0935	-0.0079	0.0006	0.0019	0.0347	-0.108622	-0.110522
2022-08	-0.0377	0.0137	0.003	0.0019	0.0199	-0.060615	-0.062515
2022-07	0.0957	0.028	-0.041	0.0008	-0.0394	0.270596	0.269796
2022-06	-0.0843	0.021	-0.0598	0.0006	0.0077	-0.116459	-0.117059
2022-05	-0.0034	-0.0183	0.0839	0.0003	0.0246	-0.0327643	-0.0330643
2022-04	-0.0946	-0.014	0.0617	0.0001	0.0487	-0.237525	-0.237625

Esta pantalla se obtuvo directamente del software que se está explicando en la computadora, para fines educativos.

Y el resumen de los parámetros del modelo es el siguiente:

Se observa que solamente el factor mercado (mkt) es positivo, indicando una influencia positiva al rendimiento esperado. El resto de los factores (smb, hml y wml) son negativos, indicando que reducen el riesgo y, por lo tanto, también el rendimiento.

OLS Regression Results

Dep. Variable:	excess_rtn	R-squared:	0.655
Model:	OLS	Adj. R-squared:	0.629
Method:	Least Squares	F-statistic:	25.61
Date:	Sat, 18 Mar 2023	Prob (F-statistic):	6.31e-12
Time:	20:57:23	Log-Likelihood:	84.516
No. Observations:	59	AIC:	-159.0
Df Residuals:	54	BIC:	-148.6
Df Model:	4		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	-0.0014	0.008	-0.181	0.857	-0.017	0.015
mkt	1.1688	0.164	7.139	0.000	0.841	1.497
smb	-0.0717	0.310	-0.231	0.818	-0.693	0.550
hml	-0.9563	0.186	-5.130	0.000	-1.330	-0.583
wml	-0.2254	0.211	-1.068	0.290	-0.649	0.198

Omnibus:	2.211	Durbin-Watson:	2.239
Prob(Omnibus):	0.331	Jarque-Bera (JB):	1.503
Skew:	-0.148	Prob(JB):	0.472
Kurtosis:	2.276	Cond. No.	41.0

Esta pantalla se obtuvo directamente del software que se está explicando en la computadora, para fines educativos.

Explicación

Con los coeficientes calculados, se utilizan para determinar el rendimiento anual esperado de la acción de Amazon.

- **# calculo del rendimiento anual esperado con el modelo de cuatro factores**
- **avg_rf = four_factor_data['rf'].mean() * 12**
- **mkt_factor = 1.1688 * four_factor_data['excess_rtn'].mean() * 12**
- **smb_factor = -0.0717 * four_factor_data['smb'].mean() * 12**
- **hml_factor = -0.9563 * four_factor_data['hml'].mean() * 12**
- **wml_factor = -0.2252 * four_factor_data['wml'].mean() * 12**
- **re = avg_rf + mkt_factor + hml_factor + wml_factor**
- **print("rendimiento esperado anual : ",re)**

Y el resultado es un rendimiento esperado anual: 0.1460078870299672, o sea 14.60 %.

Modelo CAPM con cinco factores

En 2014, Fama y French adaptaron su modelo para incluir cinco factores. Agregan el cuarto elemento denominado el factor diferenciador de rentabilidad (RMW, *high minus low*), y el quinto factor, denominado como el factor diferenciador de la inversión (CMA, *conservative minus aggersive*), con la intención de mejorar la estimación del riesgo y, por lo tanto, el cálculo del rendimiento esperado.

El modelo se representa de la siguiente forma:

$$re = r_f + \beta_1(r_m - r_f) + \beta_2(SMB) + \beta_3(HML) + \beta_4(RMW) + \beta_4(CMA) + \epsilon$$

Explicación

A continuación, se desarrolla el código Python para obtener el modelo CAPM de cinco factores, para la misma acción de Amazon.

```
# importar librerias
import pandas as pd
import yfinance as yf
import statsmodels.formula.api as smf
import pandas_datareader.data as web

# establecer la accion y el periodo de tiempo para el analisis
RISKY_ASSET = 'AMZN'
START_DATE = '2018-02-28'
END_DATE = '2023-02-28’

# descarga de los cinco factores del sitio web Fama-French
df_five_factor = web.DataReader('F-F_Research_Data_5_Factors_2x3',
                                'famafrench',
                                start=START_DATE)[0]
df_five_factor.index = df_five_factor.index.format()

# descargar la informacion de la accion de Yahoo Finance y calcular los retornos
mensuales
asset_df = yf.download(RISKY_ASSET,
                       start=START_DATE,
                       end=END_DATE)
y = asset_df['Close'].resample('M').\
                        last().\
                        pct_change().\
                        dropna()
y.index = y.index.strftime('%Y-%m')
y.name = 'return'
```

Explicación

Continuación del código Python para el modelo CAPM de cinco factores

```
# integrar los factores para el modelo con cinco factores
five_factor_data = df_five_factor.join(y)
# renombrar columnas
five_factor_data.columns = ['mkt', 'smb', 'hml', 'rmw', 'cma',
                            'rf', 'rtn']

# dividir entre 100, excepto rendimientos
five_factor_data.loc[:, five_factor_data.columns != 'rtn'] /= 100
# obtener el periodo de analisis
five_factor_data = five_factor_data.loc[START_DATE:END_DATE]
# calcular el exceso de rendimientos
five_factor_data['excess_rtn'] = five_factor_data.rtn - five_factor_data.rf

# estimar el modelo con cinco factores
five_factor_model = smf.ols(
    formula='excess_rtn ~ mkt + smb + hml + rmw + cma',
    data=five_factor_data
).fit()
print(five_factor_model.summary())
```

El resultado del modelo se muestra en la siguiente imagen. Solamente el factor del mercado es positivo, y el resto de los cuatro factores contribuyen a reducir el riesgo y, por lo tanto, el rendimiento esperado de la acción de Amazon.

OLS Regression Results

Dep. Variable:	excess_rtn	R-squared:	0.683
Model:	OLS	Adj. R-squared:	0.653
Method:	Least Squares	F-statistic:	22.85
Date:	Tue, 14 Mar 2023	Prob (F-statistic):	3.78e-12
Time:	13:21:14	Log-Likelihood:	87.044
No. Observations:	59	AIC:	-162.1
Df Residuals:	53	BIC:	-149.6
Df Model:	5		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	0.0038	0.008	0.472	0.639	-0.012	0.020
mkt	1.3267	0.165	8.040	0.000	0.996	1.658
smb	-0.5890	0.356	-1.657	0.103	-1.302	0.124
hml	-0.4575	0.267	-1.713	0.093	-0.993	0.078
rmw	-0.7990	0.395	-2.024	0.048	-1.591	-0.007
cma	-0.5857	0.405	-1.447	0.154	-1.397	0.226

Omnibus:	0.692	Durbin-Watson:	2.241
Prob(Omnibus):	0.707	Jarque-Bera (JB):	0.762
Skew:	0.095	Prob(JB):	0.683
Kurtosis:	2.477	Cond. No.	68.6

Esta pantalla se obtuvo directamente del software que se está explicando en la computadora, para fines educativos.

Explicación

Continuación del código Python para el modelo CAPM de cinco factores:

- **# calculo del rendimiento anual esperado con el modelo de cuatro factores**
- **avg_rf = five_factor_data['rf'].mean() * 12**
- **mkt_factor = 1.3267 * five_factor_data['excess_rtn'].mean() * 12**
- **smb_factor = -0.5890 * five_factor_data['smb'].mean() * 12**
- **hml_factor = -0.4575 * five_factor_data['hml'].mean() * 12**
- **rmw_factor = -0.7990 * five_factor_data['rmw'].mean() * 12**
- **cma_factor = -0.5857 * five_factor_data['cma'].mean() * 12**
- **re = avg_rf + mkt_factor + hml_factor + rmw_factor + cma_factor**
- **print("rendimiento esperado anual : ",re)**

Y el resultado es un rendimiento esperado anual de 0.08641254312788998, es decir, 8.64%.

Es importante mencionar que el rendimiento calculado es el resultado del riesgo sistemático del mercado, y que los inversionistas exigen mayor rendimiento cuando mayor es el riesgo. Por lo tanto, un valor menor de "*re*" significa que el comportamiento de la acción es más similar al del mercado, y se interpreta como menor preocupación por su futuro comportamiento.

Actividad

Para incrementar tu comprensión del tema, realiza lo siguiente:

1. Explica la manera en que aplicarías el modelo CAPM para determinar el rendimiento esperado de un portafolio de inversiones, en lugar de una sola acción.
2. Desarrolla el código Python para los modelos CAPM básico, de tres, cuatro y cinco factores, considerando la acción de la empresa Metlife Inc (MET) para contestar los siguientes incisos:
    a) Determina el rendimiento esperado para una inversión en la acción de la empresa, tomando como base un período de 3 años a la fecha y suponiendo un rendimiento libre de riesgo del 4.5 %.
    b) Elabora una tabla comparativa con los resultados de los cuatro modelos.

Presenta un reporte con los resultados y elabora tus comentarios y observaciones sobre el análisis que desarrollaste.

Cierre

El análisis de los rendimientos esperados con base en el riesgo sistemático o de mercado ha incrementado el interés de los analistas e inversionistas para mejorar la precisión en las estimaciones de los rendimientos esperados en una acción o portafolio y tomar decisiones más acertadas en la selección de los activos financieros a invertir. Históricamente, el mercado de valores ha tenido un gran desarrollo en el análisis técnico mediante herramientas estadísticas que perfeccionan la interpretación de patrones históricos para predecir patrones futuros, gracias al legado de investigadores y académicos que han desarrollado métodos cuantitativos para evaluar el riesgo sistemático.

Bibliografía

- Fama, E., y French, K. (1992). The cross-section of expected stock returns. *The Journal of Finance, 47*(2).
- Kenton, W. (2022). *Beta: Definition, Calculation and Explanation for Investors*. Recuperado de https://www.investopedia.com/terms/b/beta.asp
- Lewinson, E. (2020). *Python for Finance Cookbook*. Packt Publishing.
- Ming, J. (2021). The Capital Asset Pricing Model. *Encyclopedia of Social Sciences, 1(3)*. Recuperado de https://www.mdpi.com/2673-8392/1/3/70

SKILLING CENTER TECMILENIO

Aplicaciones Financieras

Simulación Monte Carlo en Finanzas

Introducción

La empresa administradora de fondos de inversión La Mezcla Inteligente ofrece una amplia gama de alternativas de inversión para que sus clientes tengan opciones para obtener atractivos rendimientos. De todos es sabido el comportamiento histórico de los precios de cada uno de los fondos, pero los clientes se preguntan ¿cuál será el futuro comportamiento de su patrimonio?

Explicación

- La simulación en términos estadísticos y financieros es como tener la experiencia de explorar en la información y analizarla, desarrollar valuación de activos financieros y construir portafolios, experiencia como “trader”, incluso experiencia en perder dinero, todo de manera virtual.
- La simulación denominada *Monte Carlo* es el proceso de modelar y simular un sistema, en el cual se generan escenarios aleatorios por medio de herramientas estadísticas para establecer, por ejemplo, la valuación de un activo financiero.

Explicación

Simulación dinámica del precio de acciones mediante movimiento browniano geométrico

- Para introducir el proceso de simulación Monte Carlo, considera el caso de determinar el valor futuro *VF* que tiene una inversión *P*. Para efectos prácticos, supón que el único factor que afecta el valor futuro es el mercado, representado como una variable aleatoria β con un comportamiento sobre la distribución Normal, por lo tanto, se representa como *f(β)*. El valor futuro en el tiempo *T* lo puedes representar de la siguiente forma:

$$VF_T = P^T * f(\beta)$$

- Y por medio de la simulación Monte Carlo se hacen interacciones para generar valores aleatorios de β por medio de la distribución Normal para pronosticar el valor esperado de la inversión en el tiempo *T*.

- El comportamiento del precio de una acción en el tiempo se compara a un movimiento secuencial aleatorio, en donde el valor actual está muy relacionado con su valor anterior. Este fenómeno de movimiento aleatorio es denominado como movimiento browniano, en honor al escoses Robert Brown (Whiteside, 2008), quien realizó experimentos en los movimientos del polen en el agua.

Explicación

- Un proceso estocástico es cuando el estado actual del sistema depende del estado anterior y, por lo tanto, se utiliza para entender futuros resultados que dependen sobre distribuciones de probabilidad, y se dice que sigue un movimiento browniano geométrico (*GBM, geometric brownian motion*) cuando se puede representar mediante la ecuación estocástica diferencial (Lewinson, 2020):

$$dS = \mu S dt + \sigma S dW_t$$

En donde *d* significa la derivada de la variable, *S* es el precio de la acción, μ es el retorno promedio sobre un periodo de tiempo, σ es la desviación estándar del precio y W_t es el movimiento browniano. Los incrementos brownianos son calculados mediante la variable aleatoria *t* con distribución Normal *N(0,t)*, donde *t* es el incremento de tiempo.

- Utilizando el cálculo diferencial, se puede determinar el precio en el tiempo *t*:

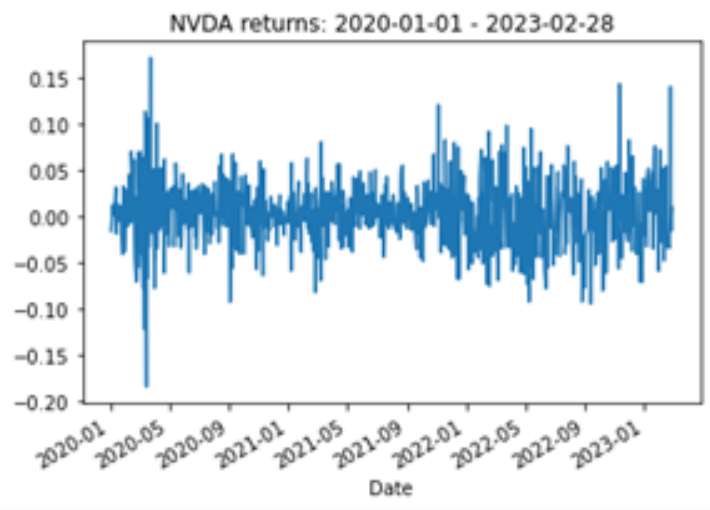
$$S(t) = S_0 e^{\left(\mu - \frac{1}{2}\sigma^2\right)t + \sigma W_t}$$

- A continuación, se usará Monte Carlo y el movimiento geométrico browniano para simular los precios de la acción de NVIDIA Corporation, empresa fabricante de semiconductores que cotiza en la bolsa de valores Nasdaq de Estados Unidos. El código Python está tomado como referencia en la publicación de Lewinson (2020).

Explicación

Código Python para simulación Monte Carlo:

```
# Importar las librerias
import numpy as np
import pandas as pd
import yfinance as yf

# Definir la accion, periodo y descargar la informacion
RISKY_ASSET = 'NVDA'
START_DATE = '2020-01-01'
END_DATE = '2023-02-28'
df = yf.download(RISKY_ASSET, start=START_DATE,
                 end=END_DATE)

# calculo de los retornos diarios
adj_close = df['Adj Close']
returns = adj_close.pct_change().dropna()
print(f'Average return: {100 * returns.mean():.2f}%')
returns.plot(title=f'{RISKY_ASSET} returns: {START_DATE} - {END_DATE}')

# Separar los datos de entrenamiento y prueba
train = returns['2020-01-01':'2022-6-30']
test = returns['2022-07-01':'2023-02-28']
```



Esta pantalla se obtuvo directamente del software que se está explicando en la computadora, para fines educativos.

Explicación

Se determinan los parámetros necesarios para la simulación:

- El promedio (mu) y desviación estándar (sigma) de los retornos diarios del periodo de entrenamiento.
- T: horizonte del pronóstico, en este caso el número de días en los datos de prueba.
- N: número de incrementos en el periodo del pronóstico, en este caso, diario.
- S_0: el precio inicial. Para esta simulación se toma el último dato del periodo de entrenamiento.
- N_SIM: número de rutas simuladas, en este caso se determina como 100.

```
# especificar los parametros de la simulacion
T = len(test)
N = len(test)
S_0 = adj_close[train.index[-1]]
N_SIM = 100
mu = train.mean()
sigma = train.std()
```

El siguiente paso es crear el módulo de las simulaciones, para lo cual se define la función simulate_gbm con los parámetros previamente definidos. Se determina el incremento de tiempo (dt), en este caso de 1 día, y los incrementos brownianos (dw) utilizando el módulo np.random.normal de Python. Se calculan las rutas brownianas (W) por medio de una suma acumulativa (np.cumsum). Se crea una matriz que contiene los intervalos de tiempo (time_steps) espaciados uniformemente dentro de un intervalo (el horizonte de la simulación), usando np.linspace. Luego, se muestra la matriz con la forma deseada usando np.broadcast.

Explicación

Se define la función simulate_gbm para generar las simulaciones:

```
# definir la funcion para las simulaciones
def simulate_gbm(s_0, mu, sigma, n_sims, T, N):
    dt = T/N
    dW = np.random.normal(scale = np.sqrt(dt),
                          size=(n_sims, N))
    W = np.cumsum(dW, axis=1)
    time_step = np.linspace(dt, T, N)
    time_steps = np.broadcast_to(time_step, (n_sims, N))
    S_t = s_0 * np.exp((mu - 0.5 * sigma ** 2) * time_steps
                       + sigma * W)
    S_t = np.insert(S_t, 0, s_0, axis=1)
    return S_t

# ejecutar las simulaciones
gbm_simulations = simulate_gbm(S_0, mu, sigma, N_SIM, T, N)
```

La matriz gbm_simulations de 100 renglones con los valores pronosticados para los días en el periodo de prueba. Se muestra parcialmente a continuación:

	0	1	2	3	4	5	6
0	151.48	157.505	159.643	165.995	168.573	176.376	171.918
1	151.48	154.399	146.788	145.278	142.99	146.717	138.095
2	151.48	156.536	151.757	153.428	153.073	155.313	141.385
3	151.48	153.237	145.254	146.709	137.852	133.291	134.02
4	151.48	148.656	149.535	144.262	138.911	144.651	147.734
5	151.48	151.969	155.845	159.906	161.286	163.604	160.95
6	151.48	144.79	145.706	140.182	144.389	140.748	141.552
7	151.48	149.662	148.461	154.207	151.536	142.212	142.065
8	151.48	154.852	162.802	167.723	172.349	176.118	175.801
9	151.48	147.311	136.551	124.5	124.185	131.108	131.126

Esta pantalla se obtuvo directamente del software que se está explicando en la computadora, para fines educativos.

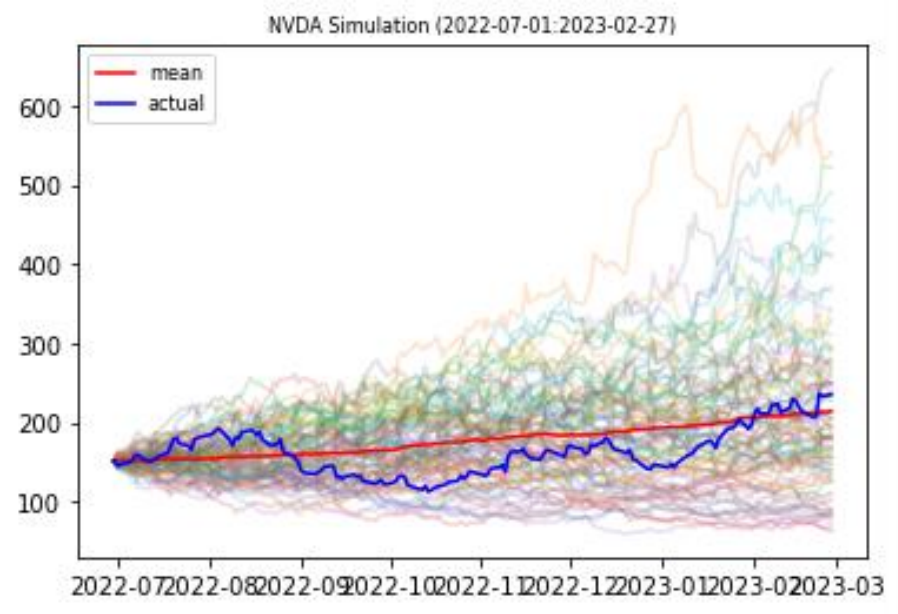
Explicación

Se prepara la configuración de la gráfica:

```
# preparar la configuracion para la grafica de la simulacion
LAST_TRAIN_DATE = train.index[-1].date()
FIRST_TEST_DATE = test.index[0].date()
LAST_TEST_DATE = test.index[-1].date()
PLOT_TITLE = (f'{RISKY_ASSET} Simulation '
              f'({FIRST_TEST_DATE}:{LAST_TEST_DATE})')
selected_indices = adj_close[LAST_TRAIN_DATE:LAST_TEST_DATE].index
index = [date.date() for date in selected_indices]
gbm_simulations_df = pd.DataFrame(np.transpose(gbm_simulations),
                                  index=index)

# elaboración de las graficas
ax = gbm_simulations_df.plot(alpha=0.2, legend=False)
line_1, = ax.plot(index, gbm_simulations_df.mean(axis=1),
                  color='red')
line_2, = ax.plot(index, adj_close[LAST_TRAIN_DATE:LAST_TEST_DATE],
                  color='blue')
ax.set_title(PLOT_TITLE, fontsize=16)
ax.legend((line_1, line_2), ('mean', 'actual'))
```

Explicación

Y el resultado final se muestra en la gráfica con las 100 trayectorias simuladas, el promedio de las simulaciones y el precio real en el periodo de prueba:



Esta pantalla se obtuvo directamente del software que se está explicando en la computadora, para fines educativos.

- Esta visualización es visible para un número razonable de rutas simuladas. En casos de la vida real, se necesita usar significativamente más de 100 rutas. Mientras mayor sea el número de rutas en la simulación, los resultados serán de mayor veracidad/confiabilidad.

Explicación

Fijación de opciones europeas mediante simulaciones

- Las opciones son un tipo de instrumento financiero derivado porque su precio está ligado al precio subyacente del activo financiero. Por medio de la compra/venta de contratos de opciones se garantiza el derecho, pero no la obligación, de comprar o vender (*call/put*) un activo en un precio establecido (conocido como strike) con cierto tiempo de anticipación.
- Las opciones son un instrumento de cobertura contra la exposición de los movimientos de un activo en forma no deseada (Lewinson, 2020).
- En particular, las opciones europeas son diferentes porque tienen una fórmula establecida para su valuación, por lo tanto, es viable utilizar Monte Carlo para simular su precio.
- Utilizando Monte Carlo, se asume que el premio de la opción incrementa al mismo ritmo que la tasa libre de riesgo, la cual se utilizará para descontar el pago futuro y calcular su valor presente. Para cada una de las rutas simuladas, se calcula el pago futuro a vencimiento, se determina el promedio de todas las rutas y se descuenta este promedio a valor presente.

Explicación

Código Python para simular opciones europeas mediante Monte Carlo:

- **# 1 importar la librerias**
- **import numpy as np**
- **from scipy.stats import norm**

- **# 2 definir los parametros para la valuacion**
- **S_0 = 100**
- **K = 100**
- **r = 0.05**
- **sigma = 0.50**
- **T = 1 # 1 año**
- **N = 252 # 252 dias en el año**
- **dt = T / N # pasos de tiempo**
- **N_SIMS = 5000**
- **discount_factor = np.exp(-r * T)**

El siguiente paso es definir la función para calcular el premio de la opción de compra call (*C(St,t)*) y la opción de venta put (*P(St,t)*) (Lewinson, 2020):

$$C(S_t,t) = N(d_1)S_t - N(d_2)Ke^{-r(T-t)}$$
$$P(S_{t,},t) = N(-d_2)Ke^{-r(T-t)} - N(-d_1)S_t$$

N(d_1) y N(d_2) = son valores de la función de distribución acumulada (CDF) normal estándar para ponderar el precio del activo en el tiempo t y al vencimiento. Se plantean d1 y d2 como funciones estocásticas de movimiento browniano.
St= precio del activo al tiempo t.
K = precio al vencimiento (strike price). Ke^(-r(T-t)) es el valor presente del precio al vencimiento.
r = tasa libre de riesgo.
σ = desviación estándar.
(T-t) = plazo para el vencimiento en el tiempo t.

Explicación

Continuación del código Python para simular opciones europeas mediante Monte Carlo:

```
# 3 define la funcion utilizando la ecuación del movimiento browniano
def black_scholes_analytical(S_0, K, T, r, sigma, type='call'):
    d1 = (np.log(S_0/K) + (r + 0.5 * sigma**2) * T) / (sigma * np.sqrt(T))
    d2 = (np.log(S_0/K) + (r - 0.5 * sigma**2) * T) / (sigma * np.sqrt(T))
    if type == 'call':
        val = (S_0*norm.cdf(d1, 0, 1)-K*np.exp(-r*T)
            *norm.cdf(d2, 0, 1))
    elif type == 'put':
        val = (K*np.exp(-r*T)*norm.cdf(-d2, 0, 1)-S_0
            *norm.cdf(-d1, 0, 1))
    return val

# 4 evaluación de la opcion call usando los parametros especificados
black_scholes_analytical(S_0=S_0, K=K, T=T, r=r,
                         sigma=sigma, type='call')

# 5 se define la función simulate_gbm para la simulacion de rutas
def simulate_gbm(s_0, mu, sigma, n_sims, T, N):
    dt = T/N
    dW = np.random.normal(scale = np.sqrt(dt),
                          size=(n_sims, N))
    W = np.cumsum(dW, axis=1)
    time_step = np.linspace(dt, T, N)
    time_steps = np.broadcast_to(time_step, (n_sims, N))
    S_t = s_0 * np.exp((mu - 0.5 * sigma ** 2) * time_steps
                       + sigma * W)
    S_t = np.insert(S_t, 0, s_0, axis=1)
    return S_t
gbm_sims = simulate_gbm(s_0=S_0, mu=r, sigma=sigma,
                        n_sims=N_SIMS, T=T, N=N)
```

Explicación

Continuación del código Python para simular opciones europeas mediante Monte Carlo:

- **# 6 calculamos el premio de la opcion**
- **premium = discount_factor * np.average(np.maximum(0,**
- **gbm_sims[:, -1] - K))**

El premio resultante es $22.31.

Estimación del valor en riesgo VaR utilizando Monte Carlo

- El valor en riesgo (*VaR*) es un métrico financiero importante que mide el riesgo asociado con un portafolio de activos financieros. Determina la pérdida potencial de la cartera con un nivel determinado de confianza, regularmente 95 %, sobre un predeterminado horizonte de tiempo y bajo condiciones normales de mercado.

- Por ley, las administradoras de fondos de inversión publican el valor VaR de cada fondo a todo el público inversionista desde el prospecto de colocación.

- Los administradores de riesgo lo utilizan para medir y controlar el nivel de exposición al riesgo. Usualmente, hay dos métodos para estimar el VaR. El primero se basa en el supuesto de que los retornos del portafolio siguen una distribución normal, y el segundo depende del ranking de los retornos históricos (Yan, 2017).

- Por ejemplo, supón que el VaR de la cartera es de $100, significa que con un nivel de confianza del 95 % (bajo condiciones normales de mercado) no se perderá más de $100 si se mantiene un portafolio por un plazo determinado.

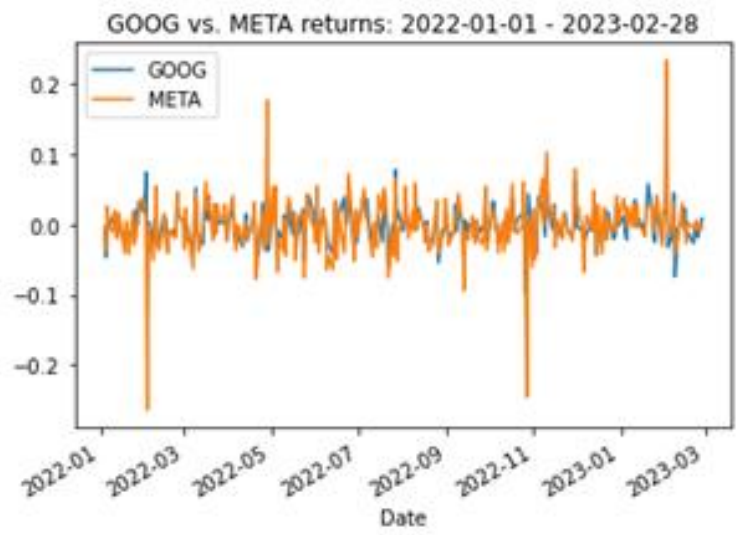
Explicación

Como ejemplo, se desarrollará el código Python para calcular el VaR de un día para un portafolio compuesto por los activos de Facebook y Google.

```
# importar las librerias
import numpy as np
import pandas as pd
import yfinance as yf
import seaborn as sns
from scipy import stats

# definir los parametros a utilizar
RISKY_ASSETS = ['GOOG', 'META']
SHARES = [5, 5] # numero de acciones de cada activo en el portafolio
START_DATE = '2018-01-01'
END_DATE = '2018-12-31'
T = 1
N_SIMS = 10 ** 5

# descargar los precios históricos diarios de Yahoo Finance
df = yf.download(RISKY_ASSETS, start=START_DATE, end=END_DATE)

# calcular los retornos diarios
adj_close = df['Adj Close']
returns = adj_close.pct_change().dropna()
plot_title = f'{" vs. ".join(RISKY_ASSETS)} returns: {START_DATE} - {END_DATE}'
returns.plot(title=plot_title)
(beta,alpha,r_value,p_value,std_err)=stats.linregress(returns['GOOG'],returns['META'])
print("coeficiente de correlacion : ", r_value)
```

Explicación

El coeficiente de correlación de los retornos diarios entre las dos series, Facebook y Google resulta en 0.68 en el periodo de análisis.



Rendimientos diarios de las acciones de Google y Facebook.

Esta pantalla se obtuvo directamente del software que se está explicando en la computadora, para fines educativos.

Explicación

- Para estimar el precio de un activo, Monte Carlo emplea valores aleatorios sobre la distribución Normal estándar. En el caso del VaR del portafolio, se aplica la covarianza histórica de los retornos diarios a los valores aleatorios generados por la simulación.

```
# calculo de la matriz de covarianza
cov_mat = returns.cov()

# desarrollar la descomposicion Cholesky de la matriz de covarianza
chol_mat = np.linalg.cholesky(cov_mat)

# obtener de los numeros aleatorios de la distribucion Normal Estandar
rv = np.random.normal(size=(N_SIMS, len(RISKY_ASSETS)))
correlated_rv = np.transpose(np.matmul(chol_mat, np.transpose(rv)))

# definir los parametros a utilizar para la simulacion
r = np.mean(returns, axis=0).values
sigma = np.std(returns, axis=0).values
S_0 = adj_close.values[-1, :]
P_0 = np.sum(SHARES * S_0)

# calcular el precio terminal de los activos considerados
S_T = S_0 * np.exp((r - 0.5 * sigma ** 2) * T +
                   sigma * np.sqrt(T) * correlated_rv)
```

- Se determinan los posibles valores del portafolio para un día posterior como P_T multiplicando la cantidad de acciones por los posibles precios S_T, y se calcula la diferencia de valores proyectados menos el valor actual del portafolio (P_T – P0) y se sortea de menor a mayor. Después se calcula el VaR del portafolio a diferentes intervalos de confianza (99 %, 98 % y 95 %).

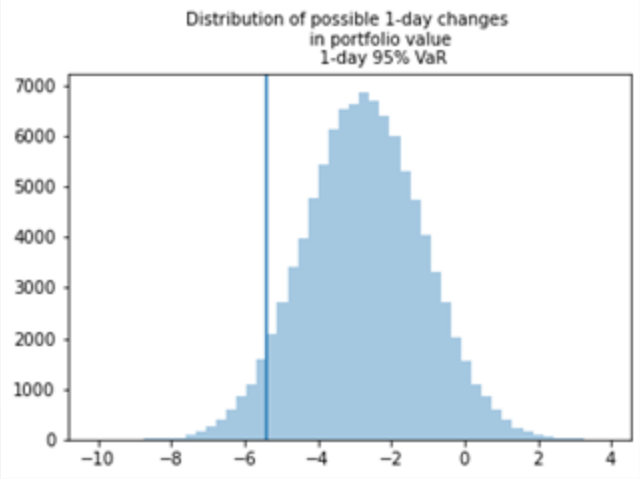
Explicación

- Continuación del código Python para calcular el VaR:
  - **# calcular el valor terminal y los retornos del portafolio**
  - **P_T = np.sum(SHARES * S_T, axis=1)**
  - **P_diff = P_T - P_0**

  - **# calcular el valor del VaR con base en el intervalo de confianza seleccionado**
  - **P_diff_sorted = np.sort(P_diff)**
  - **percentiles = [1.0,2.0, 5.0]**
  - **var = np.percentile(P_diff_sorted, percentiles)**
  - **for x, y in zip(percentiles, var):**
  - **print(f'1-day VaR with {100-x}% confidence: {-y:.2f}$')**

- Como resultados, se obtienen los valores VaR para diferentes intervalos de confianza:
  a) 1-day VaR with 99.0% confidence: 6.52%
  b) 1-day VaR with 98.0% confidence: 6.09%
  c) 1-day VaR with 95.0% confidence: 5.41%

- Se prepara la gráfica con los resultados:
  - **# presentar los resultados sobre una grafica**
  - **ax = sns.distplot(P_diff, kde=False)**
  - **ax.set_title('"Distribution of possible 1-day changes**
  - **in portfolio value**
  - **1-day 95% VaR"', fontsize=10)**
  - **ax.axvline(var[2], 0, 10000);**



Esta pantalla se obtuvo directamente del software que se está explicando en la computadora, para fines educativos.

Actividad

Para reforzar tu comprensión del tema, realiza lo siguiente:

1. Comenta cinco beneficios que puedes obtener al aplicar la simulación Monte Carlo en la valuación de activos financieros.
2. Desarrolla el código Python para resolver los siguientes incisos:
   a) Desarrolla la simulación Monte Carlo para simular los precios de la acción de Adobe Inc., (ADBE) considerando los siguientes parámetros:
      - El periodo de análisis de 3 años.
      - El periodo de entrenamiento de los primeros 30 meses y el de prueba de 6 últimos meses.
      - Realizar 500 simulaciones.
      - Muestra la gráfica con las trayectorias de la simulación.
   b) Desarrolla tus comentarios y observaciones sobre el análisis realizado.
3. Investiga la información de un fondo de inversión público que tú elijas, comenta sus principales características incluyendo el valor en riesgo contemplado.

Cierre

Como pudiste observar, una herramienta para medir el riesgo de inversión en activos financieros es la simulación Monte Carlo, para tratar de encontrar un balance entre riesgo y rendimiento. En cuanto al riesgo, se sabe que el riesgo total tiene dos componentes: riesgo del mercado y riesgo de un activo especifico (Yan, 2017). En ambos casos, una vez identificas el comportamiento de sus fluctuaciones en periodos de tiempo, puedes ajustar ese comportamiento a una distribución de probabilidad, y con esto, se expanden las posibilidades de aplicaciones de la simulación para estimar posibles valores futuros que, a final de cuentas, ayudan a incrementar la precisión de las decisiones de inversión.

Desde el día que asistió el director de estrategia a la conferencia, la empresa La Mezcla Inteligente ha mejorado mucho en su proceso de selección de activos para incluirlos en sus portafolios y mayor prestigio entre el público inversionista, lo cual le ha redituado en mayores ingresos.

Bibliografía

- Lewinson, E. (2020). *Python for Finance Cookbook*. Packt Publishing.
- Whiteside, J. (2008). *A Practical Application of Monte Carlo Simulation in Forecasting*. Recuperado de https://www.proquest.com/docview/208193177
- Yan, Y. (2017). *Python for Finance* (2ª ed.). Packt Publishing.

Tecmilenio no guarda relación alguna con las marcas mencionadas como ejemplo. Las marcas son propiedad de sus titulares conforme a la legislación aplicable, estas se utilizan con fines académicos y didácticos, por lo que no existen fines de lucro, relación publicitaria o de patrocinio.
