UNIVERSIDAD TORCUATO DI TELLA

# Financial Analytics
# 2020

**Autores:**

- Corvalán Salguero, Rodrigo
- Juárez, Lucio Ignacio
- Juárez, Matías
- Saguier Padilla, Juan

**Profesor:**

- Roccatagliata, Pablo

**Ayudante:**

- Alba Chicar, Agustín

## Abstract

El siguiente trabajo busca aplicar metodologías de modelado de datos innovadoras para afrontar decisiones de inversión con presupuestos limitados enfocado principalmente en criptomonedas. En este caso, se buscar hacer la aplicación sobre el Bitcoin y se intentará entender qué justificaría una decisión de inversión en la cripto, por qué montos y en qué momentos (montándonos en un pipeline de meta labelling dónde un modelo dice el size y el otro dice el size).

Para realizar el trabajo, mucho contenido se basó en *Advances in Financial Machine Learning* de Marcos López de Prado, quién explica cómo aplicar estas prácticas acordes a estrategias de inversión[1], utilización de grandes volúmenes de datos y escisiones de roles entre equipos.

Nos abocamos a seguir estas prácticas en pos de obtener buenos resultados en nuestros modelos, desde la definición del dataset a analizar, las variables predictivas, la aplicación de PCA, validación cruzada, modelos predictivos de clasificación con árboles y comprensión de variables explicativas a través de *feature importance*.

Este trabajo nos permite afirmar que a través de feature engineering, selección de features y optimización de hiperparámetros, se puede mejorar el retorno de nuestra estrategia momentum. Bajo nuestro análisis, la tasa de 2 años de los Treasury Bills de USA, la variable *Stock to Flow* y la cantidad de cuentas en el ecosistema Bitcoin contienen poder predictivo significativo para armar una estrategia de trading de Bitcoin.

## Exploratory Data Analysis (EDA) y Feature Engineering

Para afrontar el Análisis Exploratorio del proyecto, comenzamos por definir integraciones vs. distintas APIs que nos proveyeron los datos necesarios para tener un buen dataset sobre el cual podamos aplicar los modelos predictivos.

Realizamos integraciones con la API de Coinmetrics para obtener datos de precios y volúmenes de Bitcoin cómo también adyacentes de criptomonedas. A su vez, utilizamos la API de la FRED (Federal Reserve Economic Data) para obtener datos de tasas de interés de Bonos del Tesoro Americano.

A partir de esto comenzamos a estructurar nuestro dataset e intentar comprender en mayor detalle las correlaciones entre variables, cuáles eran más explicativas y cuáles podríamos construir para que nuestro Feature Engineering mejore los resultados modelados.

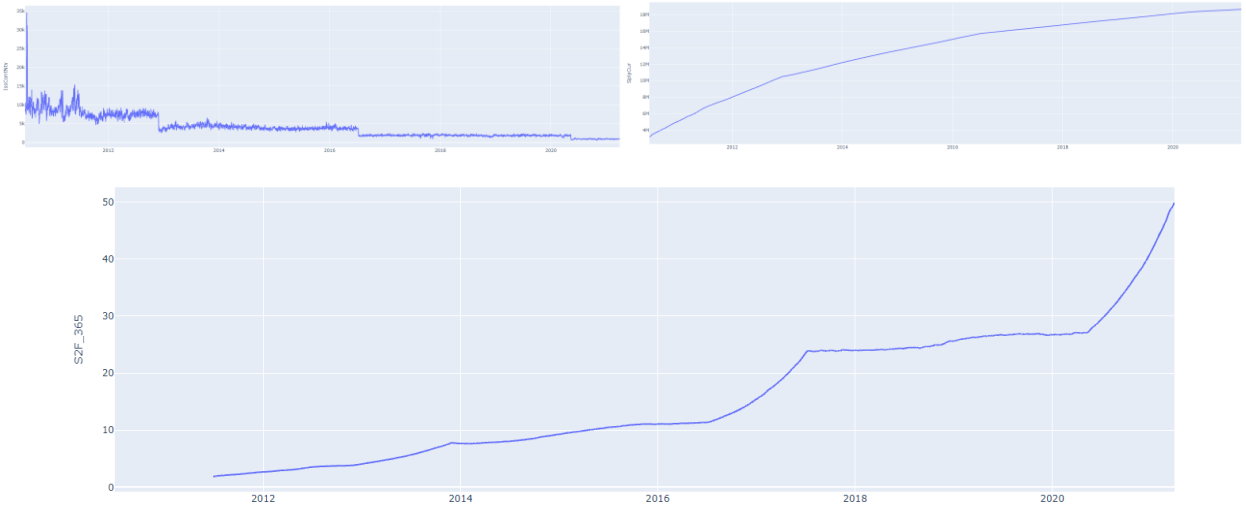
Luego de obtener las métricas clave cómo el precio del Bitcoin o el volumen transaccionado, comenzamos a investigar qué métricas son analizadas en el mercado para intentar explicar el precio y la potencial evolución. En uno de los artículos sugeridos en la consigna se trata la métrica denominada *Stock to Flow*, la cual indica la escasez existente del volumen del activo. Se puede paralelizar con metales o el oro que también suelen analizarse de esta manera. De hecho, los comparables que se utilizan en el mercado para valuar las criptomonedas por sus particularidades, son los activos monetarios cómo puede ser el Oro, Platino, Paladio u otros.

Optamos por construir e incluir esta métrica en nuestro dataset con una periodicidad diaria. La misma se calcula cómo el cociente entre la cantidad de unidades emitidas diarias (*IssContNtv*) sobre la oferta existente (*SplyCur*). Luego realizamos sobre la misma una media móvil diaria de los últimos 365 días para suavizar el trend. Como puede observarse en los gráficos debajo, la tendencia de emisiones ha venido bajando, por el mayor costo. Esto puede notarse claramente en saltos abruptos coincidentes con los halvings. Mientras tanto el Stock se incrementó, pero su crecimiento marginal es decreciente. A su vez, Stock to Flow nos muestra la combinación de ambas tendencias, con un

[1] Queremos aclarar que siempre que nos refiramos a la visión de López De Prado en este trabajo, las referencias aluden a éste libro en particular.

crecimiento sostenido, pero con saltos abruptos.

Evolución de la emisión diaria de BTC (a la izq.), Stock diario de BTC (derecha), y construcción de Stock to Flow (debajo):




También nos pareció importante incluir de forma directa los *halvings*, que son de gran impacto ante la emisión monetaria de Bitcoins (por ende, afectando la variable *stock to flow*) y generando que el valor de mercado de la cripto aumente. Para ello, construimos cuatro variables *Dummies* acorde a cada plazo en el que transcurrieron los denominados halving.

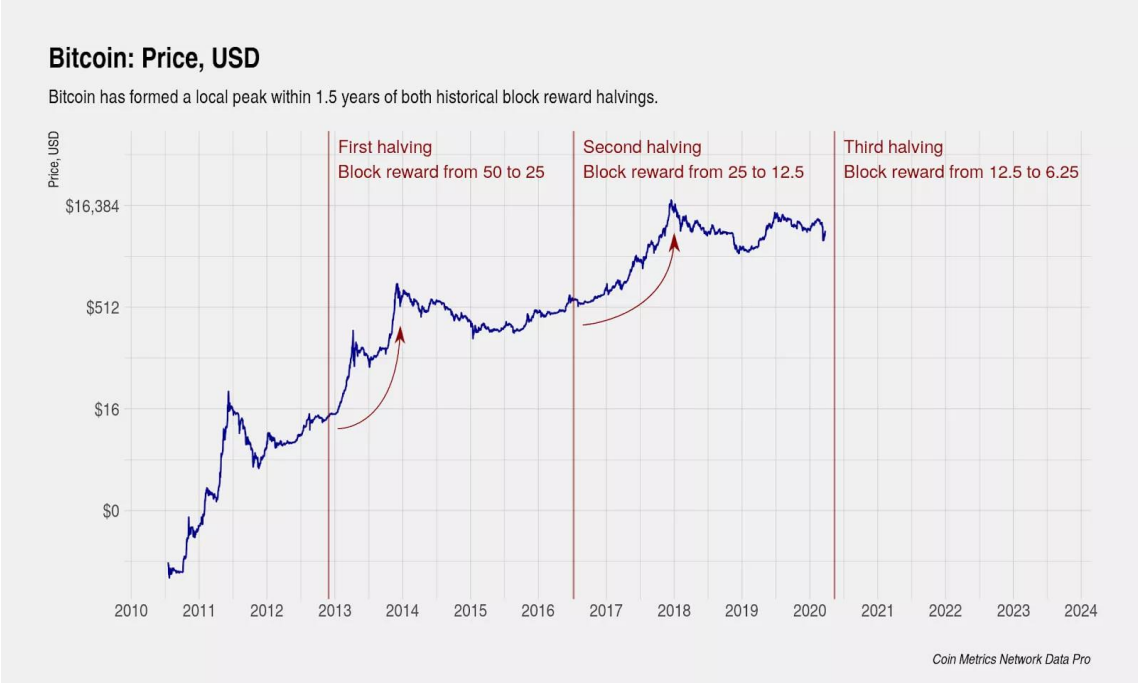
Halving events:

**Bitcoin halvings: key events**

| Event | Date | Block number | Block reward | Total new bitcoins between events |
|---|---|---|---|---|
| Bitcoin launches | 3 January 2009 | 0 (genesis block) | 50 new BTC | 10,500,000 BTC |
| First halving | 28 November 2012 | 210,000 | 25 new BTC | 5,250,000 BTC |
| Second halving | 9 July 2016 | 420,000 | 12.5 new BTC | 2,625,000 BTC |
| Third halving | Expected week commencing 18 May 2020 | 630,000 | 6.25 new BTC | 1,312,500 BTC |
| Fourth halving | Expected 2024 | 740,000 | 3.125 new BTC | 656,250 BTC |
| Fifth halving | Expected 2028 | 850,000 | 1.5625 new BTC | 328,125 BTC |

Un gráfico tomando en consideración los hitos de los halving, y la serie de precios de Bitcoin, apriorísticamente parecería sugerir que cada hito ha tenido algún impacto en la evolución del precio. El gráfico se encuentra en escala logarítmica. El lector puede observar como los primeros tres halving que ya transcurrieron parecerían haber tenido un fuerte efecto en el precio del activo.

Halving events and BTC Price:



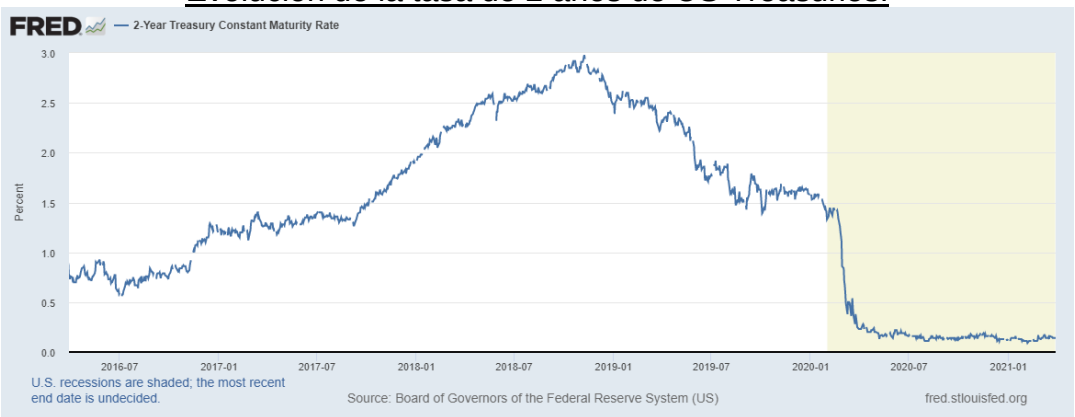

Desafortunadamente, posteriormente terminamos excluyendo el uso de *Halvings* en nuestro modelo ya que no mejoraba la performance predictiva.

Continuamos por analizar varias métricas de mercado para intentar entender que podría llegar a explicar un *risk on* en activos invertidos cómo Bitcoin. Es un hecho conocido que distintas corporaciones de prestigio, individuos e incluso inversores institucionales han hecho una especie de migración de portafolios hacia las criptomonedas. En el boom del 2021 hemos leído en las noticias que cada vez una mayor cantidad de empresas tecnológicas salen a respaldar las mismas, e incluso Facebook ha emitido una.

Para intentar entender qué puede explicar este fenómeno, analizamos las tasas de 2 (DGS2) y 10 (DGS10) años (en pos de ver cuando el apetito del mercado está por invertir a corto o largo plazo) y también la diferencia entre las mismas (T10Y2Y), el "yield spread", que nos daría una noción simplificada de curva de tasas de interés (y por ende ciclo económico, o expectativas de crecimiento/recesión).

Nuestra hipótesis, a priori, se centró en que al estar las tasas tan bajas (cómo se puede observar en el gráfico debajo) y por ende haber mayor liquidez, los inversores están buscando invertir en activos más rentables. Esto podría ser un predictor para nuestro modelo. En nuestro modelo Random Forest finalmente funcionó la tasa de 2 años, y no así la de 10 años y el yield spread.

Evolución de la tasa de 2 años de US Treasuries:



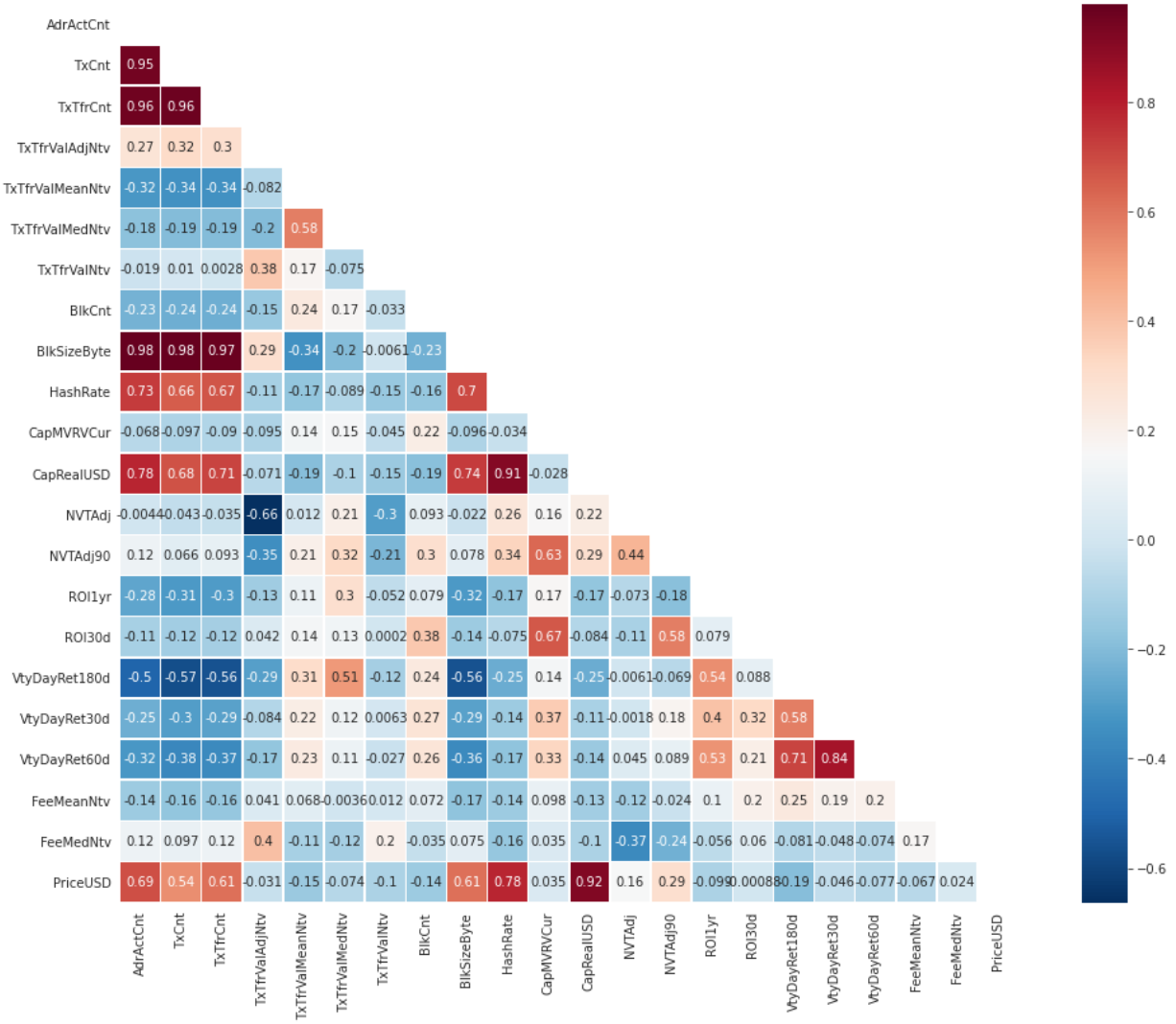

Por otro lado, con el fin de aumentar el análisis de variables predictivas, también hicimos una integración vía *requests* con un dataset ya armado extraído de la página de Coinmetrics.

Para facilitar la selección de variables, agrupamos las mismas según su naturaleza (índices transaccionales, de minería, o asociados a la capitalización del mercado, magnitud y volatilidad de los retornos, tarifas, así como relacionados al stock). Luego, evaluamos estas métricas en una matriz de correlación, de manera de comprender las interrelaciones existentes, y elegir las variables más apropiadas.

Dado que en este punto ya incluimos la variable Stock To Flow, decidimos remover las variables de stock. A su vez, quitamos las variables con información duplicada, expresadas en USD, para evitar correlaciones con variables asociadas a la tasa libre de riesgo de la Federal Reserve Board, las cuales añadiremos posteriormente, y las métricas correlacionadas entre sí.

Basándonos en esto, visualizamos la siguiente matriz:

Matriz de correlaciones entre las variables evaluadas:

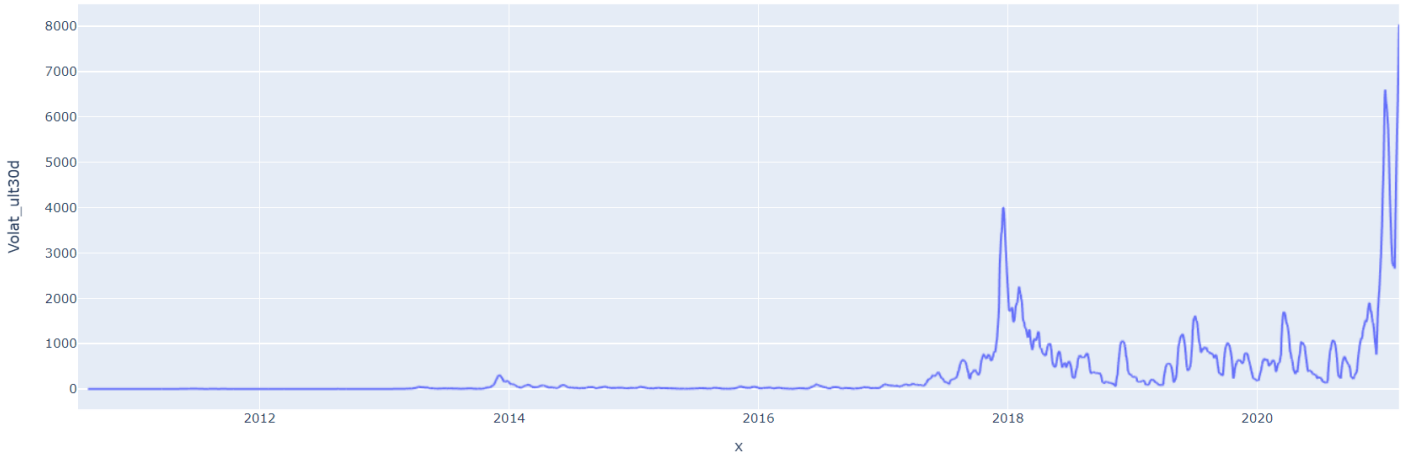



Finalmente, a partir de todo este análisis decidimos utilizar las variables *AdrActCnt*, la cual muestra la cantidad diaria de cuentas interactuando en la red Bitcoin, e intentamos implementar en nuestro modelo *HashRate*, asociada a la velocidad de minado promedio de la criptomoneda para el intervalo estudiado. Dejamos de lado las demás variables, ya sea por no poseer un gran poder explicativo en relación al precio de Bitcoin, o por tener una asociación muy fuerte con los dos indicadores mencionados anteriormente. *AdrActCnt* terminó entrando en nuestro modelo, mientras que *HashRate* no.

Con respecto a la historia de las variables, decidimos dejar como horizonte de trabajo todo aquello período en el que tuviéramos datos para Bitcoin a partir de Coinmetrics. Los datos provistos por

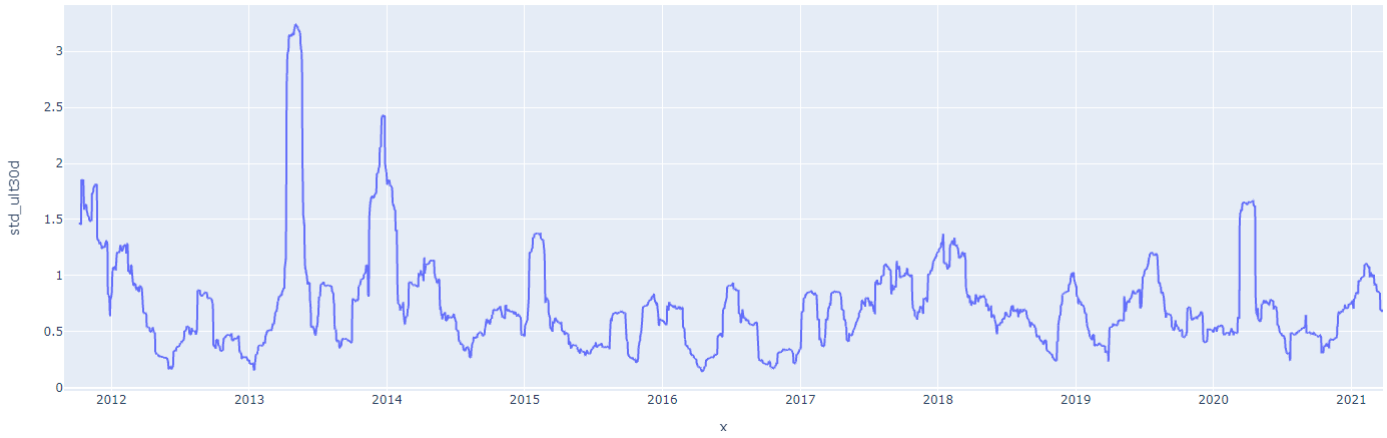
Coinmetrics comienzan aproximadamente un año después del lanzamiento de esta criptomoneda al mercado (Julio 2010). Evaluamos recortar la serie desde 2017 por distintos motivos: corresponde a la era del segundo halving (lo cual debería generar homogeneidad estructural en el costo de producción de BTC), y porque a primera vista no parecería que el trading de centavos por cada unidad de BTC tuviera algo que ver con el de miles de dólares de la actualidad.

Serie de Precios de BTC




Sin embargo, finalmente optamos por mantener toda la serie histórica. Esto porque, al margen del nivel de los precios en la serie, la volatilidad en cada momento ha sido elevada (ver gráfico debajo - incluso en el principio de la moneda, a niveles bajos de precios), y entendíamos que esto podía tener contenido informativo rico para analizar, si éramos capaces de lidiar exitosamente con el noise to signal ratio. Además, era necesario contar con una muestra extensa que luego se iba a reducir por ser que nuestro modelo se basa en cruces de medias móviles.

Volatilidad anualizada de los retornos logarítmicos diarios de BTC en la ventana móvil de 30 días




*¿Cómo podría incorporar al modelo las fases por las que ha pasado el bitcoin? ¿Los halvings del Bitcoin no podrían generar cambios estructurales? Quizá debemos tomar en cuenta en qué régimen estamos.*

Recapitulando, intentamos incorporar las fases por las que ha atravesado Bitcoin a través del feature engineering de Stock to Flow y la variable Dummy de Halvings. Complementamos éstas con el contexto de las tasas de mercado de referencia en las finanzas globales (yields de US Treasuries), ya que BTC es un activo financiero, o moneda, que no puede escindirse de este contexto. Analizamos qué otras variables propias del ecosistema Crypto (cantidad de cuentas, hash rate y muchas otras). Las variables que entraron exitosamente a nuestro modelo fueron:

- S2F_Dif: Stock to Flow
- Yield de 2 años (DGS2)

- 'AdrActCnt': Cantidad de cuentas activas

## Metalabeling

Para implementar la estrategia de trend following decidimos emplear ventanas de tiempo de 3 y 7 días como medias móviles corta (*fast_window_num_days*) y larga (*slow_window_num_days*), respectivamente. Aclaramos que no utilizamos la propuesta de la consigna de 50 y 200 debido a que con esos parámetros se obtienen pocos cruces. Adicionalmente, las notebooks de referencia que se nos proveyeron utilizaban estas ventanas, lo cual nos dio más seguridad reafirmar que este era un seteo de parámetros adecuado.

Una vez utilizado el método *getEwmEvents* se generaron las 189 observaciones donde sucedió un cruce de medias, con su signo y fecha.

Luego, nos pareció consistente setear el parámetro *num_days* que refiere a la barrera vertical (expiration limit del triple barrier) en 3 días. Esta magnitud es consistente con las ventanas de 3 y 7 días. A continuación, con *getVerticalBarrier* obtuvimos el final de la ventana desde el inicio del evento.

Con *getDailyVol* obtuvimos barreras de profit-taking y stop-loss en función de la volatilidad diaria. Esta función viene a dotar de un límite ajustado al riesgo a estas barreras.

Posteriormente, con la función *getEvents* encontramos los primeros toques para las barreras. Dentro de esta función se aloja de forma anidada la función *applyPtSlOnT1*, que es la que realmente aplica el stop loss o profit taking si es que alguno de estos se da antes de la barrera vertical. Entonces, con esto obtendremos un dataset con los 189 cruces con los datos estructurados como en el cuadro a continuación (ejemplo para la primera fila de observaciones).

| date | t1 | trgt | side |
|---|---|---|---|
| 2010-07-31 | 2010-08-01 | 0.140588 | 1 |

Lo guardamos en el dataframe *triple_barrier_events.* Hacemos mención a este dataframe, que arroja valores +1 o -1, ya que será conveniente tener presente cómo se compara con uno que implementaremos luego de la diferenciación fraccionaria (*labels*).

## Diferenciación fraccionaria

*¿Resulta necesario diferenciar las series con las que va a trabajar?*

Para introducir los features en el modelo de Machine Learning para el size, necesitaremos que éstos sean estacionarios. En consecuencia, debemos realizar algún tipo de diferenciación sobre ellos. No obstante, como propone De Prado, la serie debe ser estacionaria borrando la menor cantidad de memoria posible. Aquí es cuando aparece la técnica de **diferenciación fraccionaria**.

Una aclaración relevante es que, en un modelo de cruces de medias móviles tal como el que desarrollamos, donde los labels indican categorías tales como +1 y -1, la diferenciación fraccionaria es solo sobre los features. En la econometría tradicional de series de tiempo suelen diferenciarse ambos miembros pero en este caso solo estaremos diferenciando el lado de los features.

*¿Qué orden de diferenciación fraccionaria usaría?*

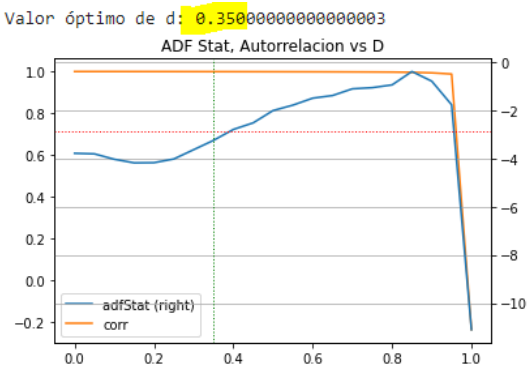
Como se detalló en el apartado sobre Feature Engineering, redujimos a un conjunto las variables que nos parecían fértiles para construir el modelo Random Forest. No obstante, finalmente no terminamos utilizando todas ellas, ya que curiosamente el modelo performó mejor con una subselección. Las variables que finalmente usamos como features fueron:

- Yield de 2 años (DGS2)

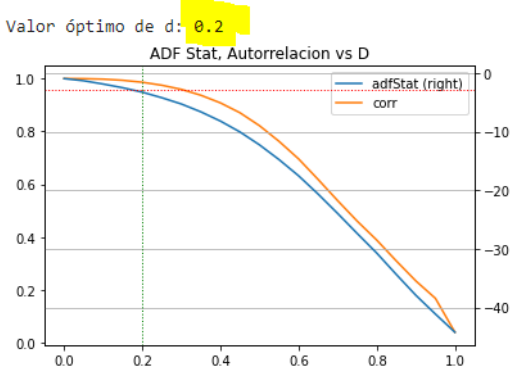
- Stock to Flow (S2F_Dif)
- Cantidad diaria de cuentas activas interactuando en la red de Bitcoin (AdrActCnt)

A continuación podemos observar el resultado de la diferenciación fraccionaria que conseguimos con la función *compute_multiple_ffd* para cada feature del modelo. En la parte superior del cuadro se encuentra el nivel óptimo de diferenciación, que es siempre menor a 1 (diferenciación unitaria). Además, el lector podrá encontrar en el script que la correlación entre las variables diferenciadas y sin diferenciar sigue siendo bastante elevada, con lo cual se mantiene la memoria requerida para no llegar a "descubrimientos falsos".

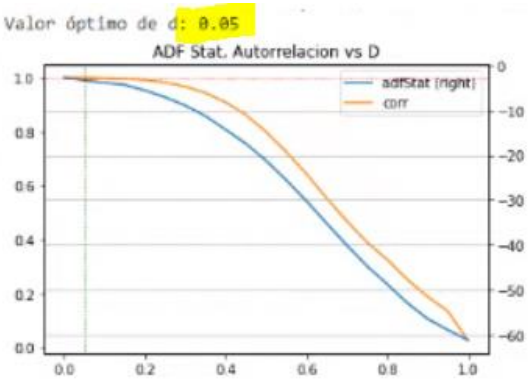
Resultados de la diferenciación fraccionaria de Stock to Flow




Resultados de la diferenciación fraccionaria de DGS2 (yield bonos del Tesoro US 2y)




Resultados de la diferenciación fraccionaria de AdrActCnt (Cantidad de cuentas activas)



## Sample weights y Uniqueness

Antes de adentrarnos en el tratamiento de Sample Weights es menester mencionar que al utilizar la función *getBins* sobre nuestro objeto *triple_barrier_events* generamos un nuevo dataframe que nos indica, sobre los cruces de medias de la regla de análisis técnico, cuál fue el outcome del bet.

De esta manera este dataframe, llamado *labels,* nos queda de la siguiente manera (para la primera fila de observaciones). Notar que los bins ya tienen como valor 0 o 1.

| | ret | bin |
|---|---|---|
| date | | |
| **2010-07-31** | 0.000000 | 0.0 |

La distribución para los 188 labels nos quedó como a continuación.

| | |
|---|---|
| 0.0 | 132 |
| 1.0 | 56 |

### Pondere por uniqueness de las observaciones.

Vale recordar que en finanzas las observaciones no surgen de procesos IID. En concreto, los labels de nuestro trabajo pueden ser dependientes entre sí, ya que podría haber al menos un retorno común al cual estén asociados (situación de concurrencia).

En ese caso, De Prado propone un tratamiento que no implica restringir la cantidad de datos para evitar solapamiento, sino más bien considerar cuánto estos se solapan.

En ese contexto, con el uso de la función *mpSampleTW* (la cual se apalanca en *mpPandasObj* para realizar procesamiento paralelo), logramos derivar el **average uniqueness**. Con éste cada feature recibe un valor de uniqueness entre 0 y 1, y se reduce el contenido con información redundante.

### ¿Cuántas formas había de hacer esto?

En el caso en el no hubieramos utilizado average uniqueness, podríamos o bien dejado de lado los labels concurrentes, o bien empleado sequencial bootstrap. De Prado no recomienda perder datos (lo cual sucede en la primera opción). La segunda opción implica muestrear por un criterio probabilístico que asigna menor probabilidad a aquellos registros que presentan especial repetición.

## Modelo para el Bet Size (en $) - Random Forest

### ¿Qué ventajas o desventajas tienen en finanzas los siguientes esquemas de ensambles? Bagging vs Random forest vs Boosting

Es un hecho conocido que tanto Random Forest como Bagging y Boosting pertenecen todos a la familia de métodos de “Ensemble” (López de Prado, capítulo 6).

*Bagging* o también llamado bootstrap aggregation es un tipo de algoritmo de ensemble. El mismo es una técnica para reducir la varianza de una función de predicción estimada (predictor). Esta idea consiste en promediar los modelos de aprendizaje de diferentes training sets o bien (si no se dispone de suficientes training sets) del bootstraping de los training sets disponibles. Para modelos de regresión del tipo árbol (CART - Classification and Regression Trees), se realizan *B* regresiones de árbol con B bootstrapped training sets y se promedian las predicciones. Este mecanismo **reduce la varianza del modelo** (*variance reduction*).

En *Random Forest* el procedimiento es similar, dada una observación del test, se guarda la clase predicha por cada uno de los *B* árboles, y luego por votación de mayoría se elige la predicción que más ocurrió en todos los modelos (una suerte de promedio de predicción). En estos casos, el parámetro de muestras y árboles *B* no genera sobreajuste (*overfitting*) aunque el mismo sea grande, por el contrario, se busca que el *B* sea lo suficientemente grande tal que el error (varianza) del modelo sea lo suficientemente bajo.

El principal problema de Bagging que resuelve Random Forest es que en el clásico bagging con árboles, cada árbol aleatorio se verá similar uno con otro ya que ponderarán mucho aquellas variables predictoras fuertes, dado que cada vez que se hace un split en un árbol, se elige una muestra aleatoria de *m* predictores candidatos como un split de los *p* predictores totales. Y en cada muestra se toman *m* nuevos predictores de los *p*. En suma, la idea del Random Forest está en forzar a que en cada split en lugar de tomar predictores de los *p* totales, se tomen predictores de un subconjunto aleatorio de tamaño *m* (nuevo hiperparámetro). Esto permite que **los árboles no sean similares, y que el modelo resultante tenga menor varianza gracias a la "descorrelación" de los árboles**, al permitir que no sobre-ponderen predictores fuertes y otros predictores tengan alguna oportunidad.

En otras palabras, Random Forest es un algoritmo de ensemble similar a bagging, en particular consiste en una "versión mejorada" de este tipo de modelos tree-based regression/classification over-bagged (es decir cuando "metemos en la bolsa" un número grande de árboles), en el cual se descorrelacionan los árboles generados (Hastie et al, 2017).

En resumen, en **RF (como en Bagging) se reduce la varianza del forecast sin overfitear**. Otra ventaja es que RF provee estimaciones out-of-bag precisas. Adicionalmente, se puede destacar que RF permite evaluar la importancia de los Features (tal como se desarrolla en el capítulo 8 de López De Prado).

Los modelos del tipo *Boosting* consisten en tomar una set de testeo aleatorio con reposición teniendo en cuenta las ponderaciones de las muestras (sample weights), se fitea un modelo estimador con ese training set, luego si el estimador logra cierto grado de score/accuracy/precisión mayor al threshold se lo almacena o sino se lo descarta; en cuarto lugar, se le da más ponderación a las observaciones mal clasificadas y menos ponderación a las correctamente predichas; en quinto lugar repite esto N veces produciendo N estimadores; y por último, el modelo final predice un promedio ponderado de las predicciones de esos N modelos

La **ventaja de Boosting** respecto de los modelos del estilo Bagging, es que reduce tanto varianza como bias en las predicciones. La ventaja de este tipo de algoritmos para modelizar es que reduce tanto la varianza como el sesgo (bias y variance) de las predicciones. Con la contra que, mejorando el bias, aumente el riesgo de overfitting. Las ventajas de los modelos de boosting se ven cuando el problema es el underfitting, mientras que los modelos de **Bagging y RF** evitan el problema del overfitting (6.6 -López De Prado).

En el presente trabajo en particular, dados los features que utulizamos, el enfoque está en evitar el overfitting de los modelos. Por lo cual evaluamos diferentes modelos de Random Forest y Bagging. En particular nos enfocaremos en un modelo de **Random Forest con múltiples features y teniendo en cuenta los Sample Weights.**

## Cross validation del modelo

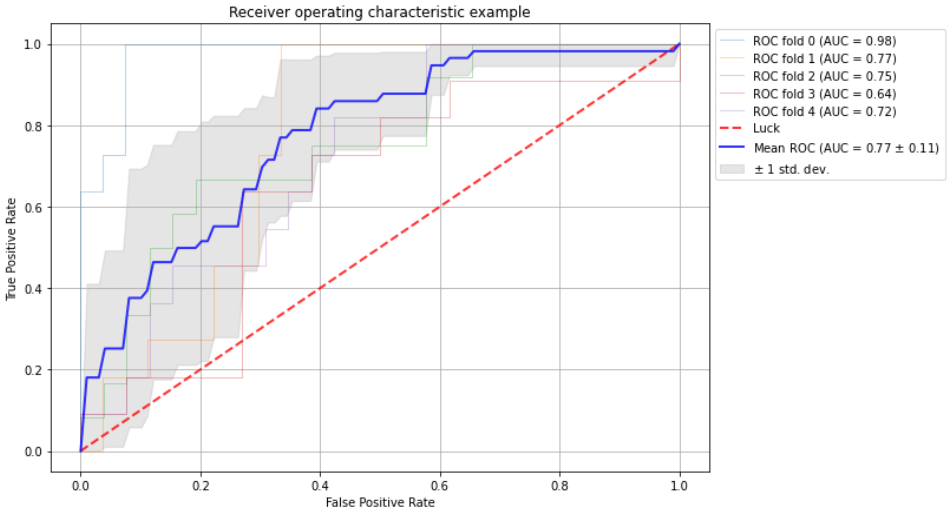
El propósito del proceso de cross validation se asocia a determinar el error de generalización de un algoritmo de Machine Learning para prevenir overfitting. Según De Prado, el cross validation que tradicionalmente se utiliza en ML al implementarse en finanzas causa problemas (contribuyendo al

overfitting antes que previniendolo).

Vale recordar que tradicionalmente se supone que los splits de cross validation se retiran de un proceso de variables aleatorias independientes e idénticamente distribuidas (IID) que se separan entre el training set y testing set. Esta separación nos aseguraría evitar el emblemático problema de *data leakage*.

El cuadro debajo exhibe la performance del modelo con cross validation tradicional. Podemos observar que el AUC promedio indica que, al clasificar una observación aleatoriamente y sin especificar un cutoff en particular, 77% de las veces nuestro modelo asignará la clasificación correcta.

Cross validation tradicional (sin embargo ni purga)




Por otro lado, De Prado argumenta que el principal déficit que tiene cross validation tradicional en finanzas se asocia a el supuesto de que las observaciones se obtienen de un proceso de variables IID. Justamente en los casos en los que los modelos tratan con features que provienen de series de tiempo correlacionadas temporalmente y labels que se construyen a partir de datos solapados (como sucede en nuestro trabajo) podríamos estar cayendo en el data leakage y encontrándonos con falsos descubrimientos.

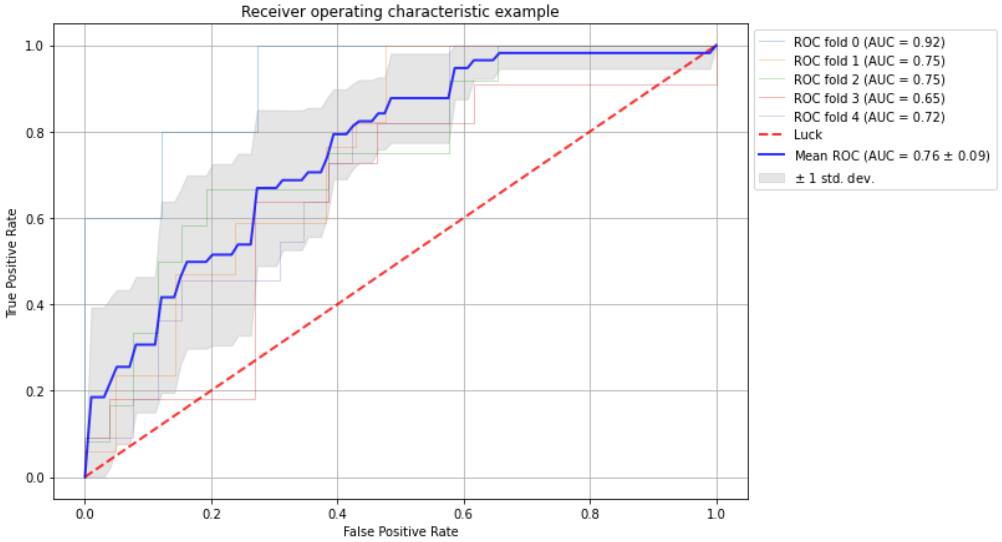
En este contexto De Prado propone dos técnicas que mitigarían el data leakage. El primero es la ***purga***, que implica excluir labels del training set que estén solapados temporalmente con labels del testing set. El segundo es el ***embargo***, que implica eliminar del training set observaciones de features con autocorrelación que sigan inmediatamente observaciones del testing set.

¿Mejoran los resultados al hacer purged k-fold y embargo en relación a la validación tradicional en series de tiempo?

El uso de cross validation con embargo y purga en teoría debería empeorar la performance con respecto a la validación tradicional. Esto se produce porque podría haber data leakage en los labels y features que mejoren espuriamente la performance de nuestro modelo.

El cuadro debajo exhibe la performance del modelo con cross validation con embargo y purga, tal como lo propone De Prado. Podemos observar que el AUC promedio ahora se redujo, como esperábamos. Ahora el AUC 76% es de de las veces nuestro modelo asignará la clasificación correcta.

Cross validation con embargo y purga




¿Esperaría que el shuffle mejore o empeore los resultados sobre el train set?

En el caso en el que hubiéramos optado por un modelo Bagging no esperaríamos que el shuffle mejore los resultados. Esto es consecuencia de que en finanzas las observaciones no pueden asumirse como generadas a partir de un proceso IID, como ya fue mencionado en este apartado. Lo que sucede es que por las características de las series financieras se observa una precisión espuriamente elevada al estimar el *test error* vía *out of bag.* El muestreo por reposición adolece de asignar al training set muestras muy similares a las del testing set (out of bag), lo que genera data leakage.

¿Qué métrica usar para evaluar en cross validation? ¿Log-loss o accuracy?

De acuerdo a De Prado, lo correcto es utilizar log-loss, y no accuracy. Esto debido a que accuracy es de alguna manera "insensible" a la probabilidad de la clasificación (por ejemplo, cuenta igual una venta equivocada con alta probabilidad a una con baja). Por el contrario, *log-loss* sí contempla las probabilidades de las predicciones.

¿Se anima hacer tanto un grid-search como un random-search de los mejores hiperparámetros del modelo?

Con respecto a la optimización de hiper parámetros, nuestro enfoque se basó en probar ambas ténicas: random search y gridsearch. Consideramos que la grilla era apropiada para descubrir valores razonables, ya que no contábamos con una noción prejuiciosa sobre su valor óptimo. Por otro lado, la grilla nos servía para trabajar con mayor intensidad sobre una zona en particular.

En randomsearch, creamos listas con valores seleccionados para los el número de árboles (*n_estimator)*, el cual es un hiper parámetro clave del modelo Random Forest, y para la profundidad que tiene cada árbol (*max_depth)*. Luego, los unimos en una grilla y empleamos la clase

*RandomizedSearchCV.* El objeto *best_randomsearch* contiene el mejor modelo con la búsqueda aleatoria.

En el caso de gridsearch realizamos el proceso análogo con la clase *GridSearchCV.* En ese caso utilizamos una grilla mucho más reducida para el número y profundidad de los árboles.

Finalmente nos quedamos con los mejores parámetros para randomsearch, ya que éstos generaron bets en varios casos (en el caso de gridsearch no logramos encontrar valores razonables que generasen bets).

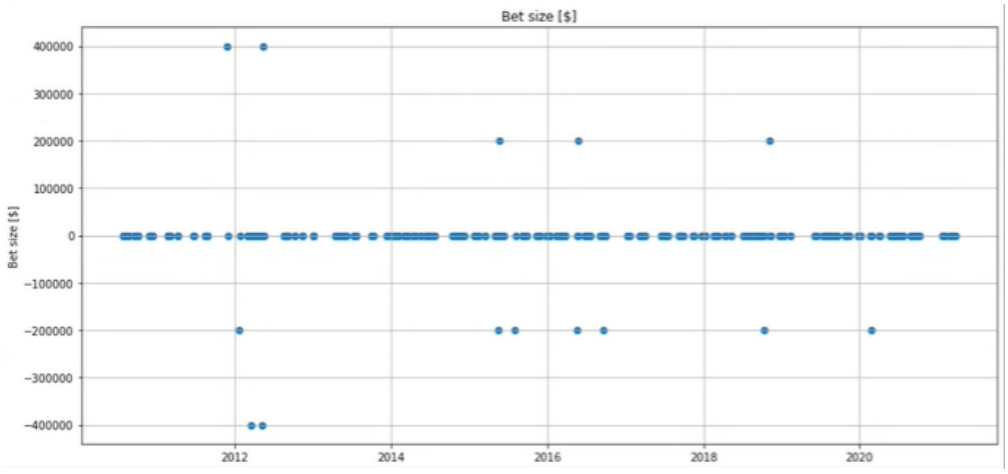
## Random Forest: modelo final y bets

Modificamos la función getSignal de los snippets de De Prado con el objetivo de corregir las probabilidades utilizadas en el estadístico del bet size. El autor confunde probabilidades muestrales con poblacionales en el denominador (en amarillo debajo).

$$z = \frac{\tilde{p} - \frac{1}{\|X\|}}{\sqrt{\tilde{p}(1-\tilde{p})}} \sim Z$$

En nuestro modelo el clasificador posee solamente dos categorías, con lo cual el valor *p* para la hipótesis nula es 0.5.

Una vez ejecutado nuestro modelo con los hiperparámetros del modelo random forest final (de random search), obtuvimos el conjunto de apuestas del cuadro debajo.




Para transformar nuestro segundo modelo en unidades monetarias utilizamos el enfoque de De Prado. El concepto es llegar a una cantidad en dinero que sea función tanto de la probabilidad de acertar (ya conociendo el side, indicado por medias móviles), como de un presupuesto.

## Feature Importance

Citando a De Prado, "*Feature Importance is a Research Tool*":

Al poner en práctica este ejercicio, buscamos comprender mejor el poder predictivo de los algoritmos aplicados. En este caso, al hacer aplicación de distintos métodos, también podemos identificar los efectos de "multicolinealidad" entre variables.

Para abordar los efectos de *feature importance*, aplicamos PCA en los datos crudos y optamos por filtrar aquellas variables que presentaban el fenómeno mostrando elevados grados de correlación.

Además de abordar los efectos de *feature importance*, trabajar con características ortogonales nos proporcionó dos beneficios adicionales:

1. Reducir la dimensionalidad de las características de la matriz X, eliminando características asociadas con pequeños autovalores propios. Esto generalmente acelera la convergencia de los algoritmos ML;
2. El análisis se realizó sobre características diseñadas para explicar la estructura de los datos.

*Mean Decrease Impurity*

Aplicamos sobre los resultados modelados MDI para explicar la importancia de las variables en la muestra del clasificador (*RandomForestClassifier*). En cada nodo de cada árbol de decisión, la característica seleccionada divide el subconjunto que recibió de tal manera que se reduce la impureza.

Por lo tanto, podemos derivar para cada árbol de decisión cuánto de la disminución general de impurezas se puede asignar a cada característica. Y dado que tenemos un bosque de árboles, podemos promediar esos valores en todos los estimadores y clasificar las características en consecuencia.

MDI hace un buen trabajo en clasificar más bajo las características ruidosas. Por lo que en nuestro modelo se puede observar que, la variable más explicativa en primer medida son las tasas diferenciadas de 2 años de treasuries, en segundo lugar la variable de stock to flow y por último la cantidad de cuentas activas en la red de Bitcoin.

```
[89] # SNIPPET 8.2 MDI (MEAN DECREASE IMPURITY) FEATURE IMPORTANCE
     def featImpMDI(fit, featNames):
       # feat importance based on IS mean impurity reduction
       df0 = {i:tree.feature_importances_ for i, tree in enumerate(fit.estimators_)}
       df0 = pd.DataFrame.from_dict(df0,orient='index')
       df0.columns = featNames
       df0 = df0.replace(0,np.nan) # because max_features=1
       imp = pd.concat({'mean':df0.mean(),
                        'std':df0.std()*df0.shape[0]**-.5},
                       axis=1)
       imp/=imp['mean'].sum()

       return imp
```

```
featImpMDI(best_randomsearch, ['trgt', 'side', 'w','S2F_Dif','yield_Dif','AdrActCnt'])
```

| | mean | std |
|---|---|---|
| trgt | 0.176976 | 0.007815 |
| side | 0.087818 | 0.002978 |
| w | 0.209622 | 0.008164 |
| S2F_Dif | 0.169024 | 0.006571 |
| yield_Dif | 0.193675 | 0.009470 |
| AdrActCnt | 0.162884 | 0.007150 |

*Mean Decrease Accuracy (Feature Permutation)*

Mean Decrease Accuracy (MDA) es un método de feature importance más lento (por su naturaleza de cálculo Out Of Sample (OOS). Tiene beneficios de aplicación sobre más de un tipo de clasificador (a diferencia de MDI que es aplicable a árboles).

Hacer la aplicación de este modelo sobre el dataset previo a la aplicación de los modelos nos resultó inconcluso, dado que le aplicaba la misma relevancia a las variables analizadas cómo predictiva y por ende insignificantes para el análisis. Esto nos llevó a quedarnos con la aplicación de Mean Decrease Impurity por sobre Mean Decrease Accuracy.

## Backtesting

Backtesting es una simulación histórica del impacto de una estrategia de haber sido ejecutada en un período pasado de tiempo. No debe utilizarse como una herramienta de research, ya que no provee información sobre las razones por las que una estrategia particular habría sido rentable; su objetivo real es el de descartar modelos, no mejorarlos.

Los errores más comunes incluyen:

1. *Sesgo de supervivencia*: Utilizar como universo de inversión el actual, lo cual conlleva ignorar que algunas empresas quebraron, y los valores asociados que dejaron de cotizar en el camino.
2. *Sesgo de anticipación*: Utilizar información que no era pública en el momento en que se habría tomado la decisión simulada. Por ello, es importante asegurar la imputación de la fecha exacta a cada dato, teniendo en cuenta las fechas de publicación, los retrasos en la distribución y las correcciones de relleno.
3. *Narración de la historia*: Inventar una historia ex-post para justificar algún patrón aleatorio.
4. *Minería de datos y data leakage*: Entrenar el modelo con filtraciones de datos de validación.
5. *Costes de transacción*: La imputación de los costos de transacción de cada período es difícil, ya que la única forma de tener certeza sobre ese coste habría sido realizando la operación real con la cartera de negociación.
6. *Presencia de outliers*: la existencia de valores extremos en el pasado podría llevar a que se diagrame una estrategia equivocada, ya que estos valores pueden no volver a ocurrir.
7. *Shorting*: La implementación de estrategias donde se tomen posiciones puede simplificar la dinámica real de shorting. La toma de una posición corta en productos líquidos requiere encontrar un prestamista. El coste del préstamo y la cantidad disponible son generalmente desconocidos, y dependen de las relaciones, el inventario, la demanda relativa, etc.

El overfitting de backtest puede definirse como un sesgo de selección en múltiples backtests. Toma lugar cuando se desarrolla una estrategia de manera de que tenga un buen rendimiento en un backtest, monetizando patrones históricos aleatorios. Dado que es poco probable que esos patrones aleatorios se repitan en el futuro, esta estrategia fracasará.

A su vez, puede considerarse que todas las estrategias sometidas a backtesting conllevan overfitting, en cierta medida, como resultado del "sesgo de selección", ya que los únicos backtests que la mayoría de la gente comparte son los que retratan estrategias de inversión supuestamente ganadoras.

Obtenga las trading rules de la estrategia definidas por los niveles de stop loss y take profit, recuerde que estos son en %, en base a simulaciones de Montecarlo de los caminos de la serie de precios. Puede usar el esquema que prefiera para esas simulaciones.

**Trading Rules**

Las estrategias de inversión pueden definirse como algoritmos que postulan la existencia de una ineficiencia del mercado. A su vez, cada estrategia de inversión requiere una táctica de aplicación, a menudo denominada "reglas de negociación", o *trading rules*.

Mientras que las estrategias pueden ser muy heterogéneas por naturaleza, las tácticas son relativamente homogéneas. Las reglas de negociación proporcionan el algoritmo que debe seguirse para entrar y salir de una posición. Por ejemplo, se entrará en una posición cuando la señal de la estrategia alcance un determinado valor. Las condiciones para salir de una posición suelen definirse a través de umbrales para *profit-taking* y *stop-loss*. Estas reglas de entrada y salida se basan en parámetros que suelen calibrarse mediante simulaciones históricas. Esta práctica conlleva el problema de *backtest overfitting*, ya que estos parámetros se centran en observaciones específicas de la muestra, hasta el punto de que la estrategia de inversión está tan apegada al pasado que no se adapta al futuro.

Aunque la evaluación de la probabilidad de sobreajuste del backtest es una herramienta útil para descartar estrategias de inversión superfluas, sería mejor evitar el riesgo de sobreajuste, al menos en el contexto de la calibración de una regla de negociación. Conforme a lo planteado por Lopez De Prado, apuntamos a conseguir esto derivando los parámetros óptimos de la trading rule directamente del proceso estocástico que genera los datos.

*Utilizando la muestra histórica completa, caracterizamos el proceso estocástico que genera la serie de rendimientos observada y obtenemos los valores óptimos de los parámetros de la regla de negociación sin necesidad de realizar una simulación histórica.*

El objetivo es encontrar una regla de negociación óptima (OTR) para aquellos escenarios en los que el sobreajuste sería más perjudicial, como cuando los retornos presentan correlación serial.

*Determinación numérica de las OTR*

Partiendo desde la estrategia de inversión optimizada por nuestro modelo, S, la cual

Utilizamos una especificación O-U para caracterizar el proceso estocástico que genera los rendimientos de la estrategia S.

$$P_{i,t} = (1-\varphi)\,\mathrm{E}_0[P_{i,T_i}] + \varphi P_{i,t-1} + \sigma\varepsilon_{i,t}$$

Estimamos los parámetros de entrada $\{\sigma,\varphi\}$, linealizando ecuación:

$$P_{i,t} = \mathrm{E}_0[P_{i,T_i}] + \varphi(P_{i,t-1} - \mathrm{E}_0[P_{i,T_i}]) + \xi_t$$

¿Nos faltaría determinar algún otro parámetro de la estrategia? ¿Cuál sería el holding period máximo?

Para poder avanzar con el cálculo de la OTC, requerimos seleccionar no solamente los parámetros profit-take y stop-loss, los cuales simulamos a partir de las combinación de los valores 0,10,21, sino que debemos configurar el retorno a largo plazo de la estrategia de inversión, el half-life de la serie de bets, así como la cantidad máxima de períodos donde se permitirá mantener una misma posición, llamado *holding period*.

Para realizar esto, configuramos el retorno de largo plazo como 0, dando lugar a un equilibrio estacionario de estabilidad para la criptomoneda, con un *half-life* de 5, y un *holding period* igual a la

cantidad de días que permanecen en la serie luego de aplicar la diferenciación fraccionaria de los regresores del modelo (2500), estableciendo así una estrategia que obedece netamente a las barreras establecidas por los parámetros profit-take y stop-loss.

Luego, el modelo determina los parámetros profit-take y stop-loss óptimos dentro de las combinaciones utilizadas, proveyendo a su vez la media y desvío del bet asociado a dicha estrategia, y un cálculo del Sharpe Ratio.

Como resultado, el modelo nos indica que la estrategia que surge de establecer el límite de profit-take en 1.0, y el de stop-loss en 7.0, es la óptima, obteniendo un bet de media -1381.58 y desvío 1.20, que con llevan un Sharpe Ratio de -1154.51.

## Strategy Risk

De Prado introduce la discusión en torno a Strategy Risk explicando que existen condiciones o límites inherentes a las estrategias de inversión. En concreto, el stop loss o take gain se terminan dando en la práctica aunque no estén explicitados.

En ese contexto, De Prado propone modelar los resultados como un proceso binomial. Con ello el autor indica que se pueden descartar estrategias que impliquen ciertas combinaciones de frecuencia en los bets, pagos y probabilidades.

Otro concepto fundamental a comprender en torno al *Strategy Risk* es que éste considera la noción de cómo la estrategia es exitosa o no al repetirse en el tiempo. Esta variable no es es controlable y no es comúnmente abordada por la academia tradicional (la cual se centra en medidas de riesgo de portafolio -en vez de riesgo de estrategia).

Si bien desafortunadamente no llegamos a completar esta sección con datos propios de nuestra estrategia igualmente queremos afirmar ciertos aspectos cualitativos y teóricos sobre Strategy Risk.

La precisión de la estrategia *p* (la probabilidad de ganar con el bet) termina quedando determinada por la dinámica de mercado, y el Sharpe Ratio objetivo es un requisito del cliente. Por ende, ninguna de las variables es directamente controlable por el portfolio manager.

En ese contexto, el número de apuestas aparece como un parámetro fundamental para definir si la estrategia será viable o no.

## Portafolio Risk

Con el objetivo de monitorear el portfolio risk definimos que la métrica más conveniente era el Value at Risk (VaR). Ésta es una medida tradicional de riesgos (ampliamente utilizada en la industria financiera para portafolios de trading) que indica la máxima pérdida posible a un nivel de confianza dado.

Desarrollamos dos funciones en la Jupyter Notebook. La primera función es *historicalVaR*, y está

preparada para devolver el VaR histórico a un nivel de confianza dado.

La segunda función que desarrollamos devuelve la métrica *Expected Shortfall*. Ésta es una medida de portafolio risk que es "coherente" (por cumplir ciertas propiedades matemáticas), y que la diferencia que presenta con VaR es que arroja una probabilidad ponderada de nuestra cola de pérdidas luego de un nivel de confianza dado. La métrica se calcula con la función *historicalCVaR*.

## Hedging Tail Risk

El debate entre Nassim Taleb y Clifford Asness surge en el contexto en el que AQR (el fondo de inversiones fundado por Clifford) afirmó que el Tail Risk hedging es inútil debido a que las opciones serían demasiado caras. Ante esto, Taleb salió a la defensiva ya que gran parte de su carrera como trader quant se ha basado en comprar opciones out of the money (OTM) baratas, apostando a beneficiarse de la ocurrencia de un cisne negro.

¿Qué posiciones tomaríamos respectivamente dados los precios de las opciones en Deribit, el día que este terminando este proyecto, si no toleramos caídas superiores al 20% en el valor del portafolio?

En el caso en el que nuestro porfolio quedase **long en BTC** al momento de armar el hedge por un evento Tail Risk entonces nuestras posiciones en derivados podrían ser:

- Long Put en BTC/USD
- Short Call en BTC/USD
- Short Futuros de BTC/USD
- Short Perpetual Swap BTC/USD

Las posiciones inversas en términos de Long y Short podrían usarse para cubrir una posición **short en BTC** de nuestro portafolio.

¿Cuánto se aseguraría usted? Esta pregunta apunta a que reflexionemos sobre que el timing del hedge importa, cuando el seguro ya está caro quizá no conviene hedgearnos, y que cuánto nos hedgeamos tiene efecto en los retornos

Con respecto a la calibración del ***size, timing y impacto en profitability*** del hedge en el contexto de Tail Risk hay una serie de consideraciones a tener en cuenta.

En primer lugar, es necesario evaluar si el ***timing*** del hedge es oportuno antes de realizar una cobertura. Idealmente, un inversor desearía comprar cobertura en un contexto de mercado en el que la volatilidad implícita en los precios de las opciones es baja.

Ahora bien, para explicar mejor qué significa esto hay que primero diferenciar los conceptos de volatilidad implícita de volatilidad histórica o realizada. La volatilidad implícita se deriva a partir de precios de mercado observables de las opciones y el uso de modelos de pricing (por ejemplo, el de Black, Scholes y Merton). Esta indica la expectativas de los agentes de mercado con respecto a la volatilidad futura en el precio del subyacente. Es decir, la volatilidad implícita de las opciones habla sobre una volatilidad esperada, desconocida y aún no realizada (en el precio del subyacente). Es posible que luego esa volatilidad no se materialice. Pero lo relevante aquí es que ya está priceada en el mercado.

Por otro lado, vale recordar que la volatilidad esperada para el precio del subyacente -BTC en este caso-, valoriza a las opciones, al amplificar la posibilidad de que estas lleguen a su fecha de ejercicio in the money

(ITM). Si los participantes de mercado esperan un contexto volátil para el BTC, entonces la volatilidad implícita en los precios de las opciones será alta. Es decir, será "cara" la cobertura.

Por consiguiente, el ***timing*** del hedge se asocia a si estamos por generar una cobertura en el contexto en el cual el resto de los participantes del mercado ya anticipan posible volatilidad futura, y por ende la pricean. Si la volatilidad prevista es elevada, entonces no será un buen contexto para cubrirse ya que pagaremos una prima alta. Es deseable contraer cobertura pagando primas bajas, a precios que reflejen volatilidad implícita baja.

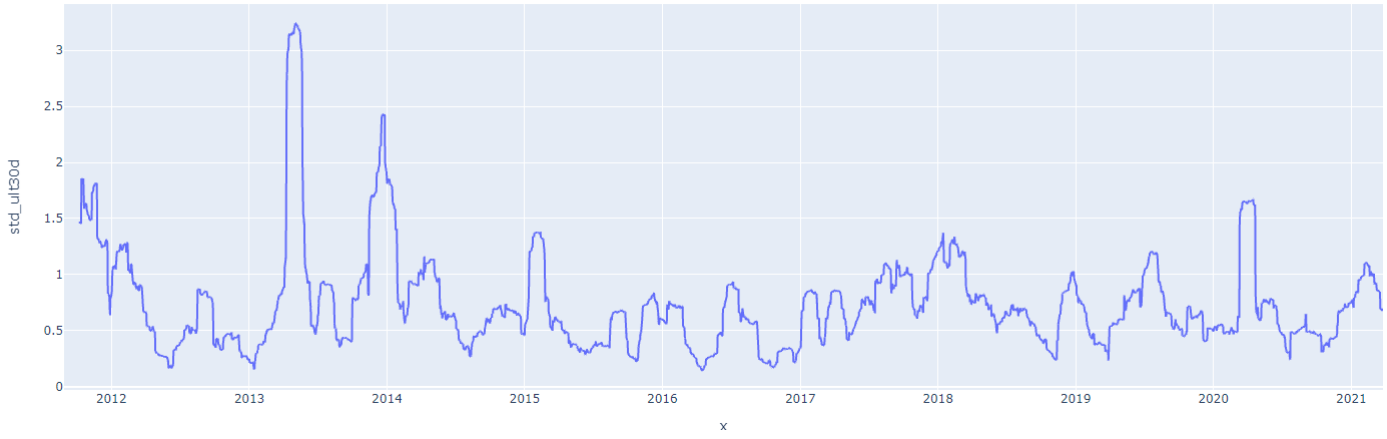
Adicionalmente, para dimensionar cuánto es una volatilidad implícita alta o baja idealmente debe compararse la volatilidad implícita reflejada en precios actualmente, con la volatilidad implícita que suele reflejarse en precios para esas opciones. Es decir, debe compararse la volatilidad implícita actual con la volatilidad implícita histórica (la volatilidad que históricamente los agentes han esperado sobre el precio).

Generalmente, para llegar a esta medida relativa de volatilidad implícita se suelen usar las métricas "IV Rank" o "IV Percentile"[2] -siendo IV la sigla de Implied Volatility. IV Rank es una estandarización entre máximos y mínimos de la volatilidad implícita actual que arroja un número entre 0 y 100%. Esta última característica del IV Rank es beneficiosa para comparar opciones en distintas ventanas del tiempo, pero fundamentalmente también para opciones con distinto tenor o subyacente, ya que cada una tiene su propia volatilidad. Además, el lector debe recordar que la volatilidad implícita puede superar el 100% mientras que el IV Rank está acotado entre 0 y 100%. El IV Percentile toma todos los percentilos en consideración, con lo cual la estandarización tiende a arrojar resultados entre 0 y 100% más suavizados. Pero en concepto es similar al IV Rank.

Desafortunadamente Deribit no provee información gratuita sobre la volatilidad implícita histórica de las opciones (calls, puts y perpetual swaps). Por consiguiente, en nuestra calibración del hedge para evaluar el timing usamos como proxy sustituto de la volatilidad implícita histórica de las opciones a la volatilidad histórica del subyacente (BTC).

El gráfico debajo (que ya hemos utilizado en la sección del EDA) ilustra la volatilidad histórica que ha mostrado Bitcoin a lo largo de su historia. En este gráfico podemos observar que BTC en una ventana corta de trading (30 días), tiene una volatilidad muy cambiante. Los mínimos y máximos históricos han llegado a ser del 11.87% y 304%, respectivamente. Mientras tanto, la media histórica ha sido de 69%.

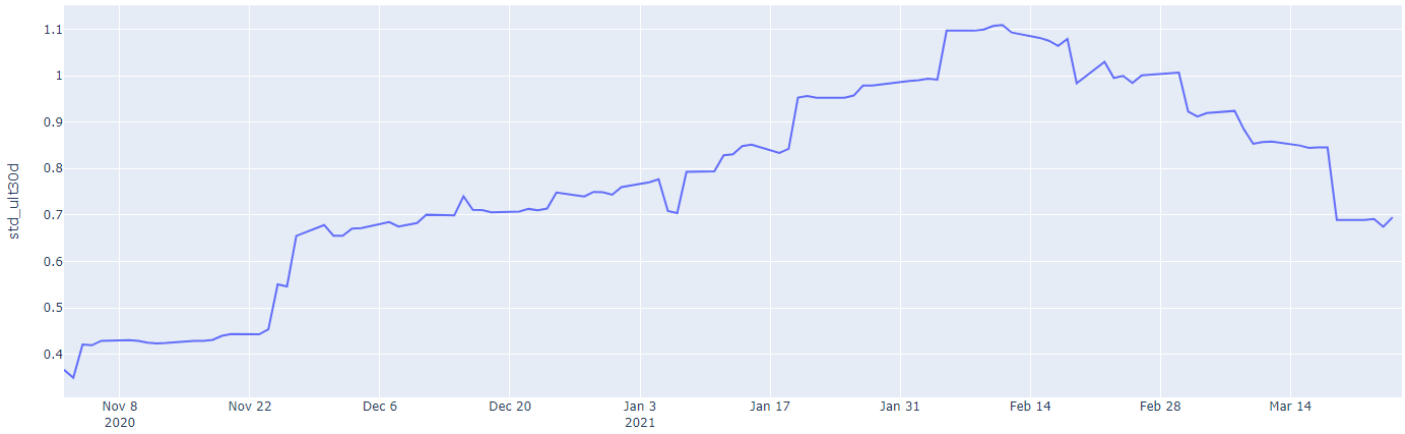
Volatilidad anualizada de los retornos logarítmicos diarios de BTC en la ventana móvil de 30 días




[2] *How high is high? The IV percentile*. https://besensibull.medium.com/how-high-is-high-the-iv-percentile-11c8d80840b5. *Implied Volatility Rank | What is IV Rank?*. http://tastytradenetwork.squarespace.com/tt/blog/implied-volatility-rank .

Luego, para analizar la tendencia reciente de la volatilidad, que es el contexto relevante para armar nuestra estrategia de Tail Risk hedging, tomamos el período desde Noviembre 2020 al presente. El lector puede notar que luego de un período de volatilidad en ascenso que llegó hasta 110%, desde Febrero de 2021 la tendencia en la volatilidad realizada es hacia la baja. Actualmente la volatilidad se encuentra en el nivel de 60%, es decir, sutilmente por debajo de la media histórica del 69%.

Volatilidad anualizada de los retornos logarítmicos diarios de BTC en la ventana móvil de 30 días (Nov-20 al presente)




Ahora bien, en el momento de armar el hedge podemos observar que las volatilidades implícitas para Calls y Puts de Bitcoin del contrato más cercano a estar ATM, para dentro de un mes (Abril 2021) son similares al trend de volatilidad realizada en el período reciente. Es decir, el contrato más cercano a estar ATM en Calls tiene una volatilidad de 73% y el de Puts de 68%, lo cual se acerca al 60% evidenciado en el subyacente en los últimos 30 días, y al 69% de media histórica de toda la serie.

Date 3/25/2021
BTC Price USD) 54,675

CONTRATO 4/30/2021

| Extrinsic Value | Intrinsic Value | | CALLS Last | Size | IV | Bid | Ask | IV | Size | Vol | Δ\|Delta | Strike | PUTS Last | Size | IV | Bid | Ask | IV | Size | Vol | Δ\|Delta | | Intrinsic Value | Extrinsic Value |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1,620 | 34,675 | | 0.6545 | 9 | 0.00% | 0.6415 $35,861.48 | 0.657 $36,727.97 | 229.90% | 9 | 0.2 | 0.99 | 20,000 | 0.0005 | 0.5 | 142.20% | 0.001 $55.94 | 0.002 $111.88 | 156.70% | 95.5 | 11.6 | -0.01 | | 0 | 84 |
| 1,607 | 30,675 | | 0.587 | 9 | 0.00% | 0.571 $31,960.25 | 0.5825 $32,603.93 | 181.40% | 9 | 9.2 | 0.99 | 24,000 | 0.0015 | 65.1 | 130.80% | 0.002 $111.90 | 0.003 $167.85 | 139.60% | 30 | 71.4 | -0.01 | | 0 | 140 |
| 1,745 | 26,675 | | - | 8 | 101.60% | 0.501 $28,042.18 | 0.5145 $28,797.81 | 162.90% | 8 | - | 0.97 | 28,000 | 0.0025 | 90.5 | 119.70% | 0.0035 $195.88 | 0.0055 $307.80 | 130.30% | 90 | 10 | -0.03 | | 0 | 252 |
| 1,591 | 22,675 | | 0.4785 | 7 | 0.00% | 0.421 $23,552.36 | 0.4465 $24,978.93 | 145.20% | 7 | - | 0.96 | 32,000 | 0.0065 | 30.7 | 110.20% | 0.006 $335.75 | 0.0075 $419.68 | 115.60% | 51.3 | 26.3 | -0.04 | | 0 | 378 |
| 2,471 | 18,675 | | 0.3985 | 7 | 93.10% | 0.363 $20,293.16 | 0.3935 $21,998.23 | 152.70% | 7 | - | 0.94 | 36,000 | 0.009 | 13.5 | 100.40% | 0.0095 $531.47 | 0.011 $615.38 | 104.20% | 57 | 196.6 | -0.06 | | 0 | 573 |
| 2,809 | 14,675 | | 0.3 | 6 | 82.90% | 0.295 $16,491.25 | 0.3305 $18,475.79 | 137.50% | 6 | - | 0.9 | 40,000 | 0.014 | 0.5 | 90.90% | 0.0145 $811.09 | 0.015 $839.05 | 91.80% | 25 | 763.5 | -0.1 | | 0 | 825 |
| 3,290 | 10,675 | | 0.25 | 6 | 74.70% | 0.23 $12,860.34 | 0.2695 $15,068.96 | 123.30% | 6 | - | 0.86 | 44,000 | 0.0185 | 17.6 | 79.20% | 0.02 $1,118.20 | 0.021 $1,174.11 | 80.60% | 23 | 159.1 | -0.14 | | 0 | 1,146 |
| 4,044 | 6,675 | | 0.1695 | 6 | 67.30% | 0.1695 $9,487.32 | 0.2135 $11,950.11 | 111.80% | 6 | 345.4 | 0.78 | 48,000 | 0.0325 | 2.1 | 71.60% | 0.031 $1,734.69 | 0.0325 $1,818.62 | 73.20% | 21.9 | 727.6 | -0.22 | | 0 | 1,777 |
| 4,108 | 4,675 | | 0.1425 | 7 | 48.10% | 0.1255 $7,020.95 | 0.1885 $10,545.42 | 108.10% | 7 | 148.4 | 0.73 | 50,000 | 0.0415 | 1.3 | 69.40% | 0.04 $2,238.79 | 0.042 $2,350.73 | 71.30% | 24.5 | 637.8 | -0.27 | | 0 | 2,295 |
| 4,390 | 2,675 | | 0.137 | 19.7 | 68.70% | 0.1235 $6,911.37 | 0.129 $7,219.16 | 73.50% | 4.5 | 326.9 | 0.67 | 52,000 | 0.054 | 1.6 | 68.10% | 0.052 $2,910.62 | 0.054 $3,022.57 | 69.90% | 14 | 365.7 | -0.33 | | 0 | 2,967 |
| 5,022 | 0 | ATM | 0.095 | 19 | 70.60% | 0.0885 **$4,952.25** | 0.091 **$5,092.14** | 72.60% | 10.2 | 370.1 | 0.54 | 56,000 | 0.0845 | 1.3 | 68.10% | 0.086 **$4,813.39** | 0.0885 **$4,953.32** | 70.00% | 38.8 | 324.1 | -0.46 | ATM | 1,325 | 3,558 |
| 3,554 | 0 | | 0.0635 | 19 | 72.60% | 0.063 $3,526.09 | 0.064 $3,582.06 | 73.40% | 13.7 | 220.2 | 0.43 | 60,000 | 0.124 | 7 | 66.80% | 0.128 $7,162.56 | 0.1385 $7,750.11 | 75.30% | 1 | 14.7 | -0.57 | | 5,325 | 2,131 |
| 2,546 | 0 | | 0.0455 | 20.7 | 75.00% | 0.045 $2,517.85 | 0.046 $2,573.80 | 75.90% | 14.1 | 115.9 | 0.33 | 64,000 | 0.1375 | 5 | 25.60% | 0.1455 $8,141.59 | 0.1985 $11,107.25 | 83.40% | 5 | - | -0.67 | | 9,325 | 299 |
| 1,874 | 0 | | 0.0355 | 20 | 77.90% | 0.033 $1,846.42 | 0.034 $1,902.38 | 78.90% | 13.2 | 238 | 0.25 | 68,000 | 0.243 | 7.1 | 0.00% | 0.2145 $12,005.50 | - | - | - | - | -0.75 | | 13,325 | |
| 1,399 | 0 | | 0.027 | 24.3 | 80.60% | 0.0245 $1,371.08 | 0.0255 $1,427.04 | 81.70% | 11.9 | 317.7 | 0.2 | 72,000 | - | - | - | - | - | - | - | - | -0.8 | | 17,325 | |
| 1,077 | 0 | | 0.023 | 22 | 83.40% | 0.0185 $1,034.56 | 0.02 $1,118.44 | 85.40% | 15.9 | 8.5 | 0.15 | 76,000 | - | - | - | - | - | - | - | - | -0.84 | | 21,325 | |

En este sentido, en principio parecería que el mercado no anticipa mayor volatilidad que la que se ha venido observando, y que la volatilidad implícita en los precios en contratos cerca de ATM se encuentra cerca del promedio histórico de volatilidad realizada. En consecuencia, este contexto no parecería ser ni especialmente conveniente (protección barata) para hedgear Tail Risk ni inconveniente (protección cara).

Con respecto al ***impacto en profitability*** del hedge, hay una relación directa entre el costo de comprar la protección, y el retorno final del portafolio (incluyendo el retorno por el hedge). Es decir, como se expuso, lo ideal sería comprar cobertura a baja volatilidad implícita, ya que si un Tail Risk event se materializa, entonces nuestra cobertura se valorizará de forma significativa en relación a su costo. Adicionalmente, si el Tail Risk

no se materializa, y no ejercemos la opción, el costo de cobertura que habremos pagado (a baja volatilidad implícita), no sustraerá rendimiento de forma material.

Con respecto al ***size*** del hedge**,** aquí buscaríamos cubrir una caída del 20% del portafolio en el plazo de un mes (por esto elegimos el contrato de Abril 2021). Aquí se presentan dos técnicas posibles para preparar el sizing del hedge: 1) lograr una posición delta neutral, es decir, que nos cubra con una aproximación lineal a cambios en el precio del BTC; 2) no basarnos en las técnicas de manual (delta y delta-gamma hedging), y en cambio ejercer la filosofía de Nassim Taleb (comprar opciones baratas OTM).

Creemos que el espíritu de la consigna no era diseñar un hedge de acuerdo a la primera opción sino a la segunda. En ese contexto, los puts OTM mostraban volatilidades implícitas relativamente más caras que la de los calls OTM. Por ejemplo si consideramos puts long a USD 20k menos que el contrato ATM (USD 56k), estos tienen una volatilidad implícita de 104% (el de USD 36k). Mientras tanto los calls OTM a USD 20k más que el contrato ATM presentaron una volatilidad implícita de 85% (el de USD 76k). En consecuencia, para un portafolio long podría ser conveniente acumular los short call necesarios para compensar una caída del 20% en el valor del portafolio.