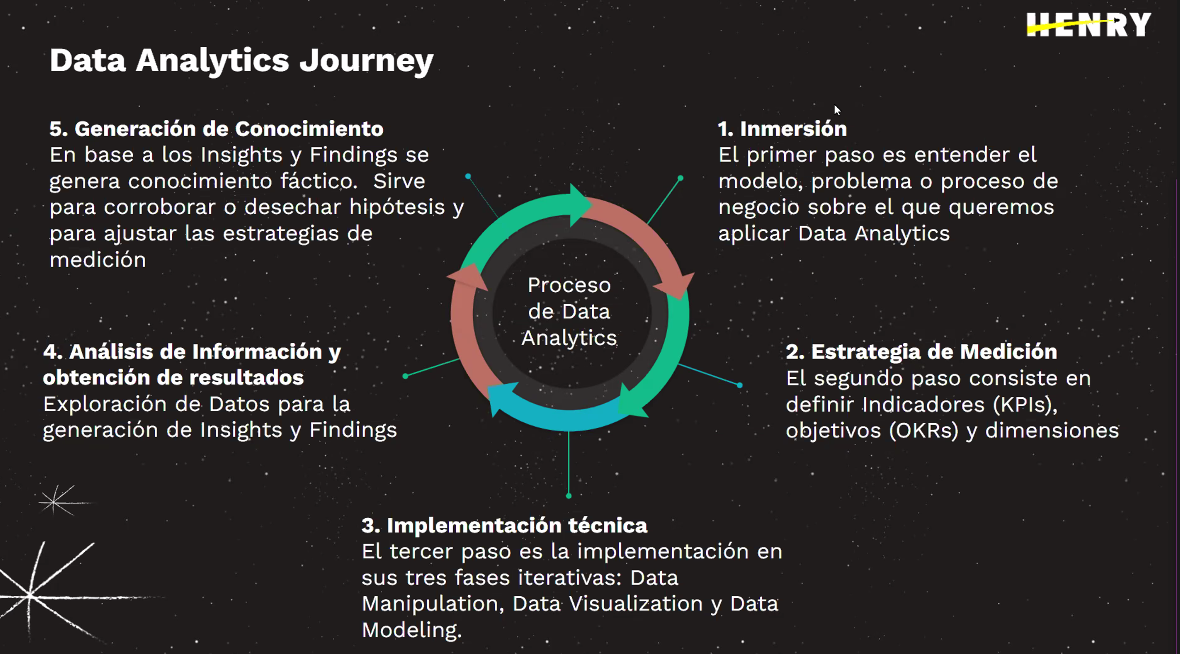
# Investigación sobre el ML en el criptotrading.

Algo que podemos hacer para mejorar nuestros resultados es un poco de data analytics:



## Areas de aplicación del ML al criptotrading:

* Predicción de precios: Los algoritmos de machine learning pueden entrenarse para analizar datos históricos de precios y otros indicadores relacionados con las criptomonedas, y luego realizar predicciones sobre los movimientos futuros de los precios. Esto puede ayudar a los traders a tomar decisiones más informadas sobre cuándo comprar o vender criptomonedas.
* Análisis de sentimiento: El machine learning también se utiliza para analizar el sentimiento del mercado en las redes sociales, foros y otras fuentes de datos en línea. Estos análisis de sentimiento pueden proporcionar información sobre las opiniones y expectativas de los inversores en relación con ciertas criptomonedas, lo que puede ser útil para predecir movimientos de precios y tomar decisiones comerciales.
* Detección de anomalías: Los algoritmos de machine learning pueden ser entrenados para identificar patrones y anomalías inusuales en los datos del mercado de criptomonedas. Esto puede ayudar a detectar posibles eventos de manipulación del mercado, hackeos u otras actividades sospechosas que podrían afectar los precios de las criptomonedas.
* Gestión de riesgos: El machine learning se utiliza en el criptotrading para desarrollar modelos de gestión de riesgos. Estos modelos pueden evaluar la volatilidad del mercado, la liquidez, el tamaño de las operaciones y otros factores para determinar estrategias de gestión de riesgos óptimas, como la asignación de capital y el establecimiento de stop-loss.

De las areas anteriores, podemos ordenarlas por dificultad, de la más sencilla a la más compleja:

1. Predicción de Precios.
2. Detección de Anomalías.
3. Gestión de Riesgos
4. Análisis de Sentimiento.
5. Reconocimiento de Patrones.
6. Análisis de Hiperparametros.

A mi parecer, es una buena manera de proceder el desarrollo de la Predicción de Precios primeramente por obvias razones, y a la par se puede tratar el desarrollo de la Gestión de Riesgos y el desarrollo de la Detección de Anomalías. Aunque la Gestión de Riesgos me parece más importante.  
  
Pienso que el Análisis de Sentimiento debería de desarrollarse una vez que tengamos un MVP (hasta que haya ganancias). Esto porque el desarrollo supone un costo mayor al de otras áreas ya que se necesitan GPUs o TPUs de servicios Cloud.  
Aunque esto lo podemos pensar mejor porque podríamos hacer análisis de sentimiento con mi máquina a manera de prueba, lo que sería ideal antes de comenzar a pagar por un servicio ya que tendríamos mejor idea de cómo utilizar los recursos, además de que podría ser este análisis el más efectivo en la predicción de precios. Sin embargo, esto también sería un costo extra.

El área de reconocimiento de patrones se puede aplicar en diversas áreas, tal como el análisis de hiperparámetros.

Por último menciono el Análisis de Hiperparámetros puesto que es algo que se debe o se puede implementar desde el área de la predicción de precios, así como a las otras áreas. Pero sería bueno hablar si lo implementamos solo básicamente al principio, porque puede ser costoso en recursos.

## 1.- ML en la Predicción de Precios para Criptotrading:

En la predicción de precios para el criptotrading, el machine learning se utiliza para analizar datos históricos de precios y otros indicadores relacionados con las criptomonedas, con el objetivo de predecir los movimientos futuros de los precios. Aquí hay algunas técnicas comunes utilizadas en este campo:

* Regresión: La regresión es una técnica de machine learning que se utiliza para predecir valores numéricos, como el precio futuro de una criptomoneda. Se pueden aplicar diferentes algoritmos de regresión, como regresión lineal, regresión polinómica o regresión de vectores de soporte (SVM), para encontrar la relación entre los datos históricos y los precios futuros.
* Redes neuronales: Las redes neuronales son modelos de machine learning inspirados en la estructura y funcionamiento del cerebro humano. En la predicción de precios de criptomonedas, se pueden utilizar redes neuronales para aprender patrones complejos en los datos históricos y hacer predicciones basadas en esas relaciones aprendidas. Las redes neuronales pueden tener diferentes arquitecturas, como redes neuronales feedforward, redes neuronales recurrentes (RNN) o redes neuronales convolucionales (CNN).
* Modelos de series temporales: Los modelos de series temporales son específicos para el análisis y la predicción de datos secuenciales en el tiempo. En el criptotrading, los precios de las criptomonedas suelen ser datos de series temporales. Algunos modelos populares para el análisis de series temporales son el modelo autorregresivo integrado de media móvil (ARIMA), el modelo de promedio móvil autoregresivo (ARMA) y el modelo autorregresivo de media móvil (ARMA).
* Aprendizaje reforzado: El aprendizaje reforzado es otra técnica utilizada en la predicción de precios para el criptotrading. En este enfoque, un agente de aprendizaje interactúa con el entorno de trading y aprende a tomar decisiones de compra y venta basadas en recompensas recibidas. El agente utiliza algoritmos de aprendizaje reforzado, como Q-learning o algoritmos genéticos, para mejorar su rendimiento a lo largo del tiempo.

A como lo veo, la mejor forma de proceder es: desarrollar la técnica de Regresión primeramente y aplicarla a la testnet de binance, posteriormente, desarrollar Modelos de Series Temporales, lo que también implementaríamos a la testnet. Es buena idea probar por separado ambas herramientas y también tratar de unirlas en una sola predicción.

A partir de aquí también va a ser necesario un script que nos permita monitorear en tiempo real las compras, las ventas y los beneficios o pérdidas... Esto claro para los resultados de la testnet primeramente.

Después del desarrollo de estas herramientas, será una buena idea agregar redes neuronales a los datos obtenidos por las herramientas anteriores. Y más adelante, podría ser agregado un Modelo de Aprendizaje Reforzado. Y bueno, se pueden buscar otras herramientas.

La transición entre la testnet y la red real será poco a poco, pero solo hasta que se haya confirmado un porcentaje de trades ganados superior al 50-55% de manera constante en la testnet (aunque puede ser revisado el porcentaje mínimo aceptado), y además será necesario tener en operación, al menos, la predicción de precios y la gestión de riesgos. Podríamos intentar trabajar en la red real como una prueba con solo la predicción de precios, pero eso sería solo un experimento para ir haciendo pruebas en la red real y tener experiencia en ello.

Un buen modelo podría replicar los resultados de la testnet con una varianza “aceptable”, pero no es aceptable un rendimiento negativo en la red real.

### 1.1.- Técnicas de ML aplicables en el área de la predicción de precios:

Todas las técnicas que voy a mencionar en las secciones siguientes están sujetas a ser usadas o no, debemos de hacer una investigación más a profundidad de cada técnica y con dicha información tomar la decisión de cuáles van a ser las técnicas que vamos a utilizar al comienzo. Las que no se utilicen al comienzo pueden ser integradas más adelante si vemos que son necesarias, pero por eso es necesario una mejor comprensión de cada técnica para poder tomar decisiones adecuadas con base en los datos que tenemos.

Para cada técnica habrá que crear un nuevo documento en donde se detalle su funcionamiento, sus variantes, y a qué datos se suele aplicar, entre otras cosas que se puedan encontrar en la investigación.

Antes de comenzar con las descripciones de cada técnica, me gustaría presentar la siguiente tabla

| **Orden de Uso en Criptotrading** | **Mis Recomendaciones para Criptotrading** |
| --- | --- |
| Regresión Lineal | Regresión de Vectores de Soporte (SVM) |
| Regresión Polinómica | Redes Neuronales |
| Regresión de Vectores de Soporte (SVM) | Regresión Polinómica |
| Redes Neuronales | Regresión Lineal |
| ARIMA | ARIMA |
| ARMA | ARMA |
| Aprendizaje Reforzado | Aprendizaje Reforzado |

En donde podemos ver las técnicas más usadas en el criptotrading del lado derecho por orden de mayor uso, y en el izquierdo las técnicas recomendadas por chatgpt en orden de importancia para él.

Cómo vemos, son justamente las herramientas de ML descritas anteriormente para la predicción de precios.

Según chatgpt, esta es una posible forma de armar un sistema de predicción de precios para el criptotrading:

*“Si estás armando un sistema de Predicción de Precios para el criptotrading, aquí hay una posible forma de combinar diferentes técnicas de regresión utilizando el enfoque de ensamble y aprovechando las fortalezas de cada una:*

*1. Recopilación y preprocesamiento de datos: Comienza recopilando datos históricos de precios de criptomonedas y otros indicadores relevantes, como volumen de operaciones, indicadores técnicos, noticias relacionadas, etc. Asegúrate de limpiar y preprocesar los datos para eliminar valores atípicos y manejar cualquier problema de calidad de datos.*

*2. Técnicas de regresión individuales: Aplica varias técnicas de regresión, como regresión lineal, regresión polinómica, regresión de vectores de soporte (SVM), redes neuronales, ARIMA, ARMA, entre otras. Cada técnica puede generar su propia predicción de precio.*

*3. Ensamble de modelos: Utiliza un enfoque de ensamble para combinar las predicciones generadas por diferentes técnicas de regresión. Puedes aplicar técnicas como el promedio de las predicciones o utilizar métodos más sofisticados como el ensamble de votación o el ensamble apilado (stacking), donde se entrena un modelo adicional para combinar las predicciones de los modelos base.*

*4. Ajuste y optimización: Realiza ajustes y optimizaciones en los hiperparámetros de cada modelo individual y del ensamble. Esto implica realizar ajustes finos en cada técnica de regresión para mejorar su rendimiento individual y también ajustar la forma en que se combinan las predicciones en el ensamble.*

*5. Evaluación y validación: Utiliza datos de prueba o técnicas de validación cruzada para evaluar el rendimiento de tu sistema de predicción de precios. Puedes utilizar métricas como el error cuadrático medio (MSE) o la raíz del error cuadrático medio (RMSE) para comparar el rendimiento de los modelos individuales y del ensamble.*

*En cuanto a la aplicación de técnicas de ML a los resultados de otras, puede ser beneficioso si las técnicas son complementarias y abordan diferentes aspectos del problema de predicción de precios. Por ejemplo, puedes utilizar la salida de una técnica de regresión como características de entrada para otro modelo de machine learning, como una red neuronal. Esto puede ayudar a capturar relaciones no lineales más complejas y mejorar la precisión de las predicciones.*

*Sin embargo, es importante tener en cuenta que agregar más complejidad al sistema no siempre garantiza mejores resultados. Es crucial realizar pruebas y validaciones adecuadas para determinar si la incorporación de técnicas adicionales realmente mejora el rendimiento del sistema. Además, es fundamental tener en cuenta el riesgo y la volatilidad asociados con el criptotrading, y no confiar únicamente en los resultados de la predicción para tomar decisiones comerciales, sino también considerar otros factores y realizar análisis adicionales.”*

Ahora sí, veamos las técnicas que mencionamos anteriormente una a una. Comenzaré describiendo las regresiones, después los modelos de series temporales, luego las redes neuronales y al final el aprendizaje reforzado.

#### 1.1.1.- Regresiones:

La regresión es un enfoque de aprendizaje automático supervisado utilizado para predecir valores numéricos basados en la relación entre variables de entrada y salida. En el contexto del criptotrading, se utiliza para predecir los precios futuros de las criptomonedas en función de datos históricos y otros indicadores relevantes.

Existen varios algoritmos de regresión que se pueden aplicar en la predicción de precios de criptomonedas, y cada uno tiene sus propias características. Aquí hay algunos ejemplos:

1.- Regresión lineal: La regresión lineal es uno de los métodos más simples y ampliamente utilizados en la predicción de precios. Busca establecer una relación lineal entre las variables de entrada y salida. El modelo intenta ajustar una línea recta que mejor se ajuste a los datos históricos y luego utiliza esa línea para predecir los precios futuros. Sin embargo, la regresión lineal puede no ser suficientemente flexible para capturar relaciones no lineales entre los datos.

2.- Regresión polinómica: La regresión polinómica es una extensión de la regresión lineal que permite modelar relaciones no lineales. En lugar de ajustar una línea recta, se ajusta un polinomio de grado superior a los datos históricos. Esto permite capturar relaciones más complejas entre las variables de entrada y salida. Sin embargo, es importante tener en cuenta que el uso de un grado polinomial muy alto puede llevar al sobreajuste del modelo.

3.- Regresión de vectores de soporte (SVM): Los vectores de soporte son algoritmos de aprendizaje automático utilizados tanto para clasificación como para regresión. En el caso de la predicción de precios, se utiliza la regresión de vectores de soporte (SVR). Este enfoque busca encontrar una función que se ajuste a los datos históricos con un margen de error aceptable. El modelo SVM busca maximizar el margen entre los datos y la función de regresión, lo que lo hace menos sensible a valores atípicos.

Es importante tener en cuenta que estos algoritmos de regresión requieren datos históricos adecuados y relevantes para entrenar y evaluar el modelo. Además, el rendimiento del modelo puede verse afectado por la calidad de los datos, la elección de características relevantes y otros factores. Es recomendable realizar una exploración exhaustiva de los datos, aplicar técnicas de preprocesamiento adecuadas y ajustar los hiperparámetros del modelo para obtener los mejores resultados posibles en la predicción de precios para el criptotrading.

Podemos hacer entrenamiento de la regresión polinómica y la regresión de vectores de soporte en el largo plazo y entrenar la lineal en el corto plazo, debido a la naturaleza de los datos, los cuales son no lineales, pero pueden llegar a aproximarse mejor linealmente en periodos cortos. Sería pensar si en realidad vale la pena hacer una investigación al respecto sobre combinar de alguna manera las predicciones de largo plazo de la polinómica y la SVM y las de corto plazo de la lineal y tomar una decisión. Porque en realidad podría no valer la pena el tiempo implementado si por separado nos dan resultados suficientemente buenos.

Aquí entra la técnica de *Ensamble de modelos* que obtuvimos de chatgpt. Así como también la técnica de *Ajuste y optimización*, la cual es básicamente un Análisis de Hiperparámetros, el cual debe ser automatizado por obvias razones y puede aplicarse en diferentes estadios del desarrollo.

Señale solo 3 métodos (en amarillo), que son los que vamos a ver, sin embargo, podemos investigar cualquiera de los siguientes métodos para aplicarlo:

* Regresión lineal
* Regresión polinómica
* Regresión logística
* Regresión de Ridge
* Regresión de Lasso
* Regresión elástica net
* Regresión de mínimos cuadrados parciales (PLS)
* Regresión de mínimos cuadrados ordinarios (OLS)
* Regresión de Huber
* Regresión de Theil-Sen
* Regresión de cuantiles
* Regresión robusta
* Regresión bayesiana
* Regresión no paramétrica
* Regresión por vecinos más cercanos (KNN Regression)
* Regresión por árboles de decisión
* Regresión por bosques aleatorios (Random Forest Regression)
* Regresión por máquinas de soporte vectorial (SVM Regression)
* Regresión por redes neuronales
* Regresión por modelos de ensamblaje (ensemble models)

Los que marqué en azul turquesa son las regresiones que me parecen buena opción para probarlas junto con las otras 3.

##### 1.1.1.1.- Regresión lineal:

La regresión lineal es un método de aprendizaje automático supervisado utilizado para predecir o modelar la relación entre una variable dependiente continua y una o más variables independientes, que pueden ser continuas o categóricas. El objetivo es encontrar una función lineal que mejor se ajuste a los datos observados y pueda usarse para hacer predicciones sobre nuevos datos.

El modelo de regresión lineal asume que la relación entre la variable dependiente y las variables independientes puede aproximarse mediante una combinación lineal de los coeficientes de las variables independientes. La forma general de un modelo de regresión lineal simple es:

y = β0 + β1\*x1 + β2\*x2 + ... + βn\*xn + ε

Donde:

- y es la variable dependiente que queremos predecir.

- β0 es el término de intersección o sesgo (intercept).

- β1, β2, ..., βn son los coeficientes o pesos de las variables independientes x1, x2, ..., xn.

- x1, x2, ..., xn son las variables independientes.

- ε es el término de error, que captura la diferencia entre el valor predicho y el valor real de y.

La regresión lineal se basa en encontrar los mejores valores para los coeficientes β0, β1, β2, ..., βn que minimicen la suma de los cuadrados de los errores (método de mínimos cuadrados). Esto se puede lograr utilizando técnicas de optimización como la minimización de la función de costo o mediante la aplicación de métodos matriciales.

Una vez que se han estimado los coeficientes, se puede utilizar el modelo de regresión lineal para hacer predicciones sobre nuevas observaciones. Simplemente se sustituyen los valores de las variables independientes en la ecuación del modelo para obtener el valor estimado de la variable dependiente.

Es importante tener en cuenta que la regresión lineal asume una relación lineal entre las variables independientes y la variable dependiente. Si la relación es no lineal, se pueden aplicar transformaciones a las variables para capturar mejor la relación no lineal o se pueden explorar otros métodos de regresión más flexibles.

Además, es fundamental evaluar la calidad del modelo de regresión lineal. Se utilizan diversas métricas de evaluación, como el coeficiente de determinación (R^2), el error cuadrático medio (MSE) o la raíz del error cuadrático medio (RMSE), para determinar qué tan bien se ajusta el modelo a los datos observados y qué tan bien generaliza a nuevos datos.

La regresión lineal es un método ampliamente utilizado debido a su simplicidad y facilidad de interpretación. Sin embargo, su rendimiento puede ser limitado en situaciones donde la relación entre las variables es no lineal o hay fuertes violaciones de los supuestos de regresión lineal, como la presencia de valores atípicos o la falta de linealidad en los residuos. En tales casos, se pueden explorar técnicas más avanzadas, como las regresiones polinómicas, regresiones de máxima verosimilitud o técnicas de aprendizaje automático más complejas.

Un pequeño código en python con la librería sklearn:

import pandas as pd

from sklearn.linear\_model import LinearRegression

from sklearn.model\_selection import train\_test\_split

from sklearn.metrics import mean\_squared\_error

# Cargar los datos de criptotrading

data = pd.read\_csv('datos\_criptotrading.csv') # Asegúrate de reemplazar 'datos\_criptotrading.csv' con tu archivo de datos

# Dividir los datos en variables independientes (X) y variable dependiente (y)

X = data[['Variable1', 'Variable2', 'Variable3']] # Reemplaza 'Variable1', 'Variable2', 'Variable3' con los nombres de tus variables independientes

y = data['Precio'] # Reemplaza 'Precio' con el nombre de tu variable dependiente

# Dividir los datos en conjuntos de entrenamiento y prueba

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

# Crear el modelo de regresión lineal

model = LinearRegression()

# Entrenar el modelo utilizando los datos de entrenamiento

model.fit(X\_train, y\_train)

# Realizar predicciones en el conjunto de prueba

y\_pred = model.predict(X\_test)

# Calcular el error cuadrático medio (MSE)

mse = mean\_squared\_error(y\_test, y\_pred)

print(f'Error cuadrático medio (MSE): {mse}')

# Obtener los coeficientes y el término de intersección

coefficients = model.coef\_

intercept = model.intercept\_

print('Coeficientes:', coefficients)

print('Término de intersección:', intercept)

Aún falta probarlo y adaptarlo a nuestros datos.

##### 1.1.1.2.- Regresión Polinómica:

Al respecto de la regresión polinómica podemos decir lo siguiente:

La regresión polinómica es una extensión del modelo de regresión lineal en el cual se permite ajustar relaciones no lineales entre la variable dependiente y las variables independientes. En lugar de ajustar una línea recta, la regresión polinómica utiliza una función polinómica para modelar los datos.

La forma general de un modelo de regresión polinómica de grado "n" es:

y = β0 + β1x + β2x^2 + ... + βn\*x^n + ε

Donde:

y es la variable dependiente que queremos predecir.

β0, β1, β2, ..., βn son los coeficientes o pesos de los términos polinómicos.

x es la variable independiente.

x^2, x^3, ..., x^n son las potencias de x hasta el grado n.

ε es el término de error.

La regresión polinómica busca encontrar los mejores valores para los coeficientes β0, β1, β2, ..., βn que minimicen la suma de los cuadrados de los errores, de manera similar a la regresión lineal.

La principal ventaja de la regresión polinómica es que permite capturar relaciones no lineales entre las variables, lo que puede ser especialmente útil cuando los datos muestran patrones curvilíneos o no se ajustan bien a una línea recta. Al agregar términos polinómicos de grado superior, el modelo puede ajustarse mejor a los datos y proporcionar predicciones más precisas.

Sin embargo, es importante tener en cuenta que la regresión polinómica también tiene algunas limitaciones. A medida que aumenta el grado del polinomio, el modelo se vuelve más complejo y puede llevar al sobreajuste (overfitting), lo que significa que el modelo se ajusta demasiado a los datos de entrenamiento y no generaliza bien a nuevos datos. Por lo tanto, es fundamental realizar una validación adecuada y considerar técnicas de regularización, como la regularización de Ridge o LASSO, para controlar la complejidad del modelo y evitar el sobreajuste.

Es importante hacer hincapie en las técnicas de regularización con el objetivo de mejorar el modelo.

Un código sencillo de este método es el siguiente:

import numpy as np

from sklearn.linear\_model import LinearRegression

from sklearn.preprocessing import PolynomialFeatures

# Datos de ejemplo

X = np.array([1, 2, 3, 4, 5]).reshape(-1, 1)

y = np.array([2, 3, 6, 10, 15])

# Crear características polinómicas de grado 2

poly\_features = PolynomialFeatures(degree=2)

X\_poly = poly\_features.fit\_transform(X)

# Crear el modelo de regresión lineal

model = LinearRegression()

# Entrenar el modelo utilizando las características polinómicas

model.fit(X\_poly, y)

# Realizar predicciones

X\_test = np.array([6]).reshape(-1, 1)

X\_test\_poly = poly\_features.transform(X\_test)

y\_pred = model.predict(X\_test\_poly)

print(f'Predicción: {y\_pred}')

Básicamente se utiliza el método de la regresión lineal pero se le añaden características polinomiales.

##### 1.1.1.3.- Regresión de Vectores de Soporte:

Ahora veamos más sobre la última técnica, de la cual encontramos que:

La Regresión de Vectores de Soporte (SVR, por sus siglas en inglés) es una técnica de aprendizaje automático utilizada para problemas de regresión, donde el objetivo es predecir un valor numérico continuo en lugar de clasificar en clases discretas. SVR es una variante del algoritmo SVM (Support Vector Machine) adaptada para problemas de regresión.

A diferencia de la regresión lineal, que busca ajustar una línea o un hiperplano a los datos, la SVR busca encontrar una función no lineal que se ajuste a los datos de manera óptima. Utiliza los vectores de soporte, que son puntos de datos cercanos a la función de regresión, para definir la función objetivo.

El objetivo principal de la SVR es minimizar el error de regresión y, al mismo tiempo, controlar la complejidad del modelo para evitar el sobreajuste. La complejidad del modelo se controla mediante parámetros como C y ε, que permiten establecer un equilibrio entre la precisión del ajuste y la flexibilidad del modelo.

El proceso básico para aplicar SVR es el siguiente:

1. Preparación de los datos: Al igual que en cualquier problema de regresión, es necesario preparar los datos de entrada y salida en un formato adecuado.

2. Selección del kernel: La SVR utiliza el "kernel trick", que permite mapear los datos a un espacio de características de mayor dimensión para encontrar hiperplanos no lineales de separación. Los kernels más comunes utilizados en SVR son el kernel lineal, el kernel polinómico y el kernel radial (RBF).

3. Definición de la función de pérdida: La función de pérdida en SVR tiene como objetivo minimizar el error de regresión. La forma específica de la función de pérdida depende del tipo de SVR utilizado, como SVR lineal, SVR no lineal, SVR con kernel polinómico, etc.

4. Optimización: La optimización en SVR implica encontrar los vectores de soporte y los coeficientes correspondientes que minimicen la función de pérdida. Esto se puede lograr utilizando técnicas de optimización como el descenso de gradiente o métodos de programación cuadrática.

5. Predicción: Una vez que se ha entrenado el modelo SVR, se puede utilizar para realizar predicciones en nuevos datos de entrada y estimar los valores continuos correspondientes.

Es importante destacar que la SVR tiene en cuenta los vectores de soporte, que son los puntos de datos más relevantes y cercanos a la función de regresión. Estos puntos desempeñan un papel crítico en la definición del modelo y tienen un impacto directo en la precisión de las predicciones.

La implementación de SVR se puede realizar utilizando bibliotecas populares de aprendizaje automático en Python, como scikit-learn, que proporciona una implementación flexible y fácil de usar de SVR. La biblioteca scikit-learn ofrece una variedad de opciones y parámetros para ajustar y personalizar el modelo SVR según los requisitos del problema de regresión específico.

Un código sencillo en python con sklearn sería el siguiente:

import numpy as np

from sklearn.svm import SVR

# Datos de entrenamiento

X\_train = np.array([[1], [2], [3], [4], [5]]) # Características

y\_train = np.array([2, 4, 6, 8, 10]) # Etiquetas

# Crear el modelo SVR

svr\_model = SVR(kernel='linear', C=1.0, epsilon=0.2)

# Entrenar el modelo

svr\_model.fit(X\_train, y\_train)

# Datos de prueba

X\_test = np.array([[6], [7], [8]]) # Características

# Realizar predicciones

y\_pred = svr\_model.predict(X\_test)

# Imprimir las predicciones

print("Predicciones:")

for i in range(len(X\_test)):

print(f"Característica: {X\_test[i][0]}, Predicción: {y\_pred[i]}")

En este ejemplo, se utiliza una implementación de SVR con un kernel lineal. Se crea una instancia de la clase SVR de scikit-learn y se especifican los parámetros kernel='linear' para utilizar un kernel lineal, C=1.0 para el parámetro de regularización y epsilon=0.2 para controlar el margen de error permitido.

Luego, se entrena el modelo utilizando los datos de entrenamiento (X\_train y y\_train), y se realiza la predicción en nuevos datos de prueba (X\_test). Finalmente, se imprimen las predicciones correspondientes.

Es importante tener en cuenta que este es solo un ejemplo básico para ilustrar cómo se utiliza la técnica SVR. En la práctica, es posible ajustar los parámetros, explorar diferentes kernels y realizar una validación cruzada para encontrar el modelo óptimo para el problema de regresión específico.

#### 1.1.2.- Modelos de Series Temporales:

Los modelos de series temporales son técnicas de inteligencia artificial (AI) ampliamente utilizadas en el análisis y la predicción de datos secuenciales, como los precios de criptomonedas en el criptotrading. Estos modelos se centran en la estructura temporal de los datos y buscan capturar patrones, tendencias y estacionalidad para realizar predicciones futuras.

A continuación, describiré algunas de las técnicas de modelos de series temporales que se aplican comúnmente al criptotrading:

1.- Modelos ARIMA (Autoregressive Integrated Moving Average): Los modelos ARIMA son una clase de modelos estadísticos que se utilizan para predecir valores en series temporales estacionarias. Estos modelos consideran la autocorrelación y utilizan una combinación de componentes autoregresivos (AR), de promedio móvil (MA) y de integración (I) para modelar la tendencia, la estacionalidad y los errores. Los modelos ARIMA son flexibles y se pueden ajustar a diferentes patrones temporales.

2.- Modelos de suavizamiento exponencial: Los modelos de suavizamiento exponencial son métodos que se basan en ponderar los valores pasados para realizar una predicción. Estos modelos son adecuados para series temporales con tendencia y estacionalidad. Algunas variantes populares de los modelos de suavizamiento exponencial incluyen el suavizamiento exponencial simple (SES), el suavizamiento exponencial doble (Holt) y el suavizamiento exponencial triple (Holt-Winters).

3.- Redes Neuronales Recurrentes (RNN): Las RNN son un tipo de arquitectura de redes neuronales especialmente diseñadas para modelar secuencias de datos, como las series temporales. Estas redes tienen conexiones recurrentes que les permiten recordar información anterior y utilizarla en la predicción futura. Las variantes populares de las RNN incluyen las Long Short-Term Memory (LSTM) y las Gated Recurrent Unit (GRU), que han demostrado ser efectivas en la predicción de series temporales debido a su capacidad para capturar dependencias a largo plazo.

4.- Modelos de Estado Espacial (State Space Models): Los modelos de estado espacial son técnicas estadísticas que representan una serie temporal como una combinación de componentes observables y no observables. Estos modelos se utilizan para capturar la tendencia, la estacionalidad y otros factores que pueden influir en la serie temporal. Los modelos de estado espacial, como el filtro de Kalman y el suavizador de Kalman, se utilizan comúnmente en la predicción de series temporales.

5.- Modelos de Aprendizaje Profundo (Deep Learning): Los modelos de aprendizaje profundo, como las redes neuronales convolucionales (CNN) y las redes neuronales generativas (GAN), también se aplican en la predicción de series temporales en el criptotrading. Estos modelos utilizan capas y conexiones más profundas que las redes neuronales tradicionales, lo que les permite aprender representaciones más complejas de los datos y capturar patrones temporales más sofisticados.

Es importante señalar aquí que el punto 3 y el 5 son redes neuronales, por lo que no las vamos a describir en esta sección. Solo vamos a dejar la información puesto que se relaciona con la aplicación de redes neuronales a las series de tiempo, sin embargo, abordaremos 5 puntos, los otros 2 son los siguientes:

6.- GARCH (Generalized Autoregressive Conditional Heteroskedasticity):

GARCH es un modelo estadístico utilizado para modelar y predecir la volatilidad en las series temporales financieras. Este modelo se basa en la idea de que la varianza de una serie temporal puede cambiar con el tiempo y estar influenciada por los valores pasados de la serie. El modelo GARCH captura tanto la autocorrelación en los residuos como la heteroscedasticidad condicional, lo que lo hace adecuado para el modelado de la volatilidad en el criptotrading y otros mercados financieros.

7.- Prophet:

Prophet es una librería de código abierto desarrollada por Facebook para el análisis y la predicción de series temporales. Utiliza un enfoque aditivo basado en descomposición de tendencia y estacionalidad. Prophet puede manejar automáticamente la detección de cambios en las tendencias, la identificación de componentes estacionales y la gestión de valores atípicos en los datos. Esta herramienta es especialmente popular en el campo del criptotrading debido a su facilidad de uso y capacidad para modelar series temporales con patrones complejos y no lineales.

Dentro de los modelos de series temporales tenemos la siguiente lista:

* ARIMA (Autoregressive Integrated Moving Average)
* SARIMA (Seasonal Autoregressive Integrated Moving Average)
* GARCH (Generalized Autoregressive Conditional Heteroskedasticity)
* VAR (Vector Autoregression)
* LSTM (Long Short-Term Memory)
* GRU (Gated Recurrent Unit)
* RNN (Recurrent Neural Network)
* Prophet
* Wavelet transform
* Holt-Winters
* Kalman filter
* Exponential Smoothing
* Fourier Transform
* Seasonal Decomposition of Time Series (STL)
* TBATS (Trigonometric Exponential Smoothing State Space Model)
* State Space Models

En color amarillo están las que abordaremos y en morado son las redes neuronales que se proponen como modelos de series temporales puesto que también son útiles a la hora de modelar series temporales. Las redes neuronales las abordaremos en la siguiente sección.

Ahora pasemos a la descripción de los métodos que obtuvimos:

##### 1.1.2.1.- ARIMA:

ARIMA (Autoregressive Integrated Moving Average) es un modelo estadístico utilizado para analizar y predecir series temporales. El modelo ARIMA combina componentes de autocorrelación, integración y medias móviles para modelar la estructura temporal de los datos.

El acrónimo ARIMA representa los siguientes componentes:

1. AR (Autoregressive): El componente autoregresivo se refiere a la dependencia de una observación actual de valores pasados de la misma serie temporal. En un modelo ARIMA, el término AR indica el número de observaciones pasadas que se utilizarán en la predicción actual.

2. I (Integrated): El componente integrado se refiere a la diferenciación de la serie temporal para hacerla estacionaria. La estacionariedad implica que las propiedades estadísticas de la serie, como la media y la varianza, no cambian a lo largo del tiempo. La diferenciación implica tomar la diferencia entre las observaciones sucesivas para eliminar las tendencias o patrones no estacionarios.

3. MA (Moving Average): El componente de media móvil utiliza los errores de predicción pasados para modelar la influencia de los residuos en la predicción actual. Al igual que el componente autoregresivo, el término MA indica el número de errores de predicción pasados que se tendrán en cuenta en la predicción actual.

El modelo ARIMA se ajusta a los datos utilizando métodos estadísticos como la estimación de máxima verosimilitud. Una vez ajustado, se puede utilizar para hacer predicciones futuras en la serie temporal.

ARIMA es una técnica ampliamente utilizada en el análisis de series temporales, incluido el criptotrading. Sin embargo, la selección adecuada de los parámetros AR, I y MA requiere un análisis cuidadoso de los datos y puede implicar pruebas y ajustes para obtener un modelo óptimo.

Un código sencillo para aplicar este método es el siguiente:

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

from statsmodels.tsa.arima.model import ARIMA

# Cargar los datos

data = pd.read\_csv('datos.csv', index\_col='fecha', parse\_dates=True)

series = data['precio']

# Ajustar el modelo ARIMA

model = ARIMA(series, order=(p, d, q))

model\_fit = model.fit()

# Obtener las predicciones

predictions = model\_fit.predict(start='2023-01-01', end='2023-12-31')

# Visualizar los resultados

plt.plot(series, label='Datos reales')

plt.plot(predictions, color='red', label='Predicciones')

plt.legend()

plt.xlabel('Fecha')

plt.ylabel('Precio')

plt.title('Modelo ARIMA - Predicciones de precios')

plt.show()

En este código, debes reemplazar 'datos.csv' con el nombre y la ubicación de tu archivo de datos. También debes ajustar los valores de 'p', 'd' y 'q' en la línea order=(p, d, q) para especificar los órdenes del modelo ARIMA que deseas utilizar.

Una vez que el modelo está ajustado, puedes obtener las predicciones llamando al método predict y especificando el rango de fechas para el cual deseas hacer las predicciones.

Aun no se ha probado el código.

##### 1.1.2.2.- SARIMA:

SARIMA (Seasonal Autoregressive Integrated Moving Average) es una extensión del modelo ARIMA que se utiliza para modelar y predecir series temporales que exhiben tanto componentes de estacionalidad como de tendencia. Al igual que ARIMA, el modelo SARIMA combina componentes de autocorrelación, integración y medias móviles, pero también incorpora un componente de estacionalidad para capturar patrones repetitivos a lo largo del tiempo.

El acrónimo SARIMA representa los siguientes componentes:

1. AR (Autoregressive): El componente autoregresivo captura la dependencia de una observación actual de los valores pasados de la misma serie temporal, teniendo en cuenta tanto la estacionalidad como la tendencia.

2. I (Integrated): El componente integrado se refiere a la diferenciación de la serie temporal para hacerla estacionaria, similar a ARIMA. La diferencia con ARIMA es que SARIMA también considera la diferenciación estacional para eliminar patrones estacionales.

3. MA (Moving Average): El componente de media móvil modela la influencia de los errores de predicción pasados en la predicción actual, teniendo en cuenta tanto la estacionalidad como la tendencia.

4. S (Seasonal): El componente estacional se refiere a la incorporación de patrones estacionales en el modelo. Esto implica la inclusión de términos autoregresivos, diferencias y medias móviles en un período estacional.

El modelo SARIMA se ajusta a los datos utilizando métodos estadísticos similares a ARIMA. Los parámetros clave que se deben especificar al ajustar un modelo SARIMA son: el orden del componente AR, I y MA (representados como p, d, q), y el orden del componente estacional (representado como P, D, Q, s).

SARIMA es especialmente útil cuando se trabaja con datos que exhiben patrones estacionales, como en el caso de muchas series temporales financieras, incluido el criptotrading. Al modelar tanto la estacionalidad como la tendencia, SARIMA puede proporcionar predicciones más precisas y capturar mejor las fluctuaciones a largo plazo en los datos.

Un pequeño código de esta técnica sería el siguiente:

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

from statsmodels.tsa.statespace.sarimax import SARIMAX

# Cargar los datos

data = pd.read\_csv('datos.csv', index\_col='fecha', parse\_dates=True)

series = data['precio']

# Ajustar el modelo SARIMA

model = SARIMAX(series, order=(p, d, q), seasonal\_order=(P, D, Q, s))

model\_fit = model.fit()

# Obtener las predicciones

predictions = model\_fit.predict(start='2023-01-01', end='2023-12-31')

# Visualizar los resultados

plt.plot(series, label='Datos reales')

plt.plot(predictions, color='red', label='Predicciones')

plt.legend()

plt.xlabel('Fecha')

plt.ylabel('Precio')

plt.title('Modelo SARIMA - Predicciones de precios')

plt.show()

Al igual que en el código del modelo ARIMA, debes reemplazar 'datos.csv' con el nombre y la ubicación de tu archivo de datos. Además, debes ajustar los valores de 'p', 'd', 'q' para el componente ARIMA y los valores de 'P', 'D', 'Q', 's' para el componente estacional.

Una vez que el modelo SARIMA está ajustado, puedes obtener las predicciones llamando al método predict y especificando el rango de fechas para el cual deseas hacer las predicciones.

Tampoco este código se ha probado.

##### 1.1.2.3.- Exponential Smoothing:

Exponential Smoothing (Suavizado Exponencial) es una técnica utilizada para predecir y suavizar series temporales, especialmente aquellas que no presentan patrones estacionales claros. Esta técnica se basa en la idea de que los valores más recientes tienen más peso o importancia en la predicción que los valores anteriores.

El suavizado exponencial calcula la predicción futura como una combinación ponderada de los valores pasados de la serie temporal. La ponderación se realiza mediante la asignación de pesos exponenciales decrecientes a los valores pasados, donde los valores más recientes tienen un peso mayor.

Existen diferentes variantes del suavizado exponencial, cada una con diferentes características y suposiciones. Aquí se describen tres de las variantes más comunes:

1. Simple Exponential Smoothing (Suavizado Exponencial Simple): En esta variante, la predicción futura se calcula como una combinación ponderada de los valores pasados de la serie temporal, donde todos los valores tienen el mismo peso. La fórmula general es:

Predicción futura = Peso \* Valor actual + (1 - Peso) \* Predicción anterior

El peso es un parámetro que determina la importancia relativa de los valores pasados y puede ser ajustado para optimizar la predicción.

2. Double Exponential Smoothing (Suavizado Exponencial Doble): Esta variante se utiliza para series temporales que también exhiben una tendencia en los datos. Además de considerar los valores pasados, también se incorpora una estimación de la tendencia en la predicción. Se utiliza una fórmula similar a la del suavizado exponencial simple, pero con dos pesos diferentes para los valores pasados y la estimación de la tendencia.

3. Triple Exponential Smoothing (Suavizado Exponencial Triple o Holt-Winters): Esta variante se utiliza para series temporales que también presentan una componente estacional. Además de considerar los valores pasados y la tendencia, también se incorpora una estimación de la estacionalidad en la predicción. La fórmula general incluye tres pesos diferentes para los valores pasados, la tendencia y la estacionalidad.

El suavizado exponencial es una técnica sencilla pero efectiva que puede proporcionar predicciones suaves y estables para series temporales. Sin embargo, no es adecuada para todas las situaciones y puede no capturar patrones complejos en los datos. Es importante tener en cuenta las características de los datos y las suposiciones de cada variante de suavizado exponencial al aplicar esta técnica.

El Double Exponential Smoothing (Suavizado Exponencial Doble) es una técnica de suavizado utilizada para predecir y suavizar series temporales que presentan una tendencia en los datos. Es una extensión del Simple Exponential Smoothing que incorpora una estimación de la tendencia en la predicción.

Aquí vemos que tenemos 3 variantes del suavizado exponencial. A primera vista, podemos ver que la suavización exponencial doble y la triple son los métodos que más nos conviene utilizar, puesto que las series de trading pueden contener tendencia y estacionalidad.

###### 1.1.2.3.1.- Double Exponential Smoothing:

El Double Exponential Smoothing utiliza dos componentes principales: nivel y tendencia. El componente de nivel representa el nivel base de la serie temporal, mientras que el componente de tendencia representa la dirección y la velocidad de cambio de la serie temporal.

La fórmula general para calcular la predicción en Double Exponential Smoothing es la siguiente:

Predicción futura = Nivel actual + Tendencia actual

El nivel y la tendencia se actualizan en cada paso de tiempo utilizando las siguientes ecuaciones:

Nivel actual = (1 - Alfa) \* (Valor actual) + Alfa \* (Nivel anterior + Tendencia anterior)

Tendencia actual = (1 - Beta) \* (Nivel actual - Nivel anterior) + Beta \* Tendencia anterior

Donde:

- Valor actual: El valor actual de la serie temporal.

- Nivel anterior: El nivel en el paso de tiempo anterior.

- Tendencia anterior: La tendencia en el paso de tiempo anterior.

- Alfa: El factor de suavizado para el nivel, que determina el peso dado al valor actual en la actualización del nivel. Un valor más alto de Alfa da más peso al valor actual y hace que la serie se ajuste más rápidamente a los cambios.

- Beta: El factor de suavizado para la tendencia, que determina el peso dado a la diferencia entre los niveles en la actualización de la tendencia. Un valor más alto de Beta hace que la tendencia se ajuste más rápidamente a los cambios.

El Double Exponential Smoothing es útil cuando se desea predecir y suavizar una serie temporal que presenta una tendencia lineal. Sin embargo, no es adecuado para capturar cambios no lineales o patrones más complejos en los datos. En tales casos, pueden ser necesarias técnicas más avanzadas, como el Triple Exponential Smoothing (Holt-Winters).

Es importante ajustar adecuadamente los parámetros Alfa y Beta para obtener las mejores predicciones. Esto puede hacerse mediante métodos de optimización, como la minimización del error cuadrado medio o mediante validación cruzada en diferentes conjuntos de datos.

De acuerdo con lo que encontré, esta técnica funciona bien para comportamientos lineales, sin embargo, no es buena para capturar comportamientos no lineales. Recomienda la técnica de Holt-Winters para casos no lineales.

###### 1.1.2.3.2.- Triple Exponential Smoothing or Holt-Winters:

Holt-Winters es una técnica de suavizado exponencial triple utilizada para modelar y predecir series temporales que exhiben componentes de tendencia y estacionalidad. Esta técnica es una extensión del suavizado exponencial doble (Double Exponential Smoothing) al que se agrega un componente adicional para capturar patrones estacionales en los datos.

El método de Holt-Winters tiene tres componentes principales: nivel (level), tendencia (trend) y estacionalidad (seasonality). Cada componente se actualiza en función de los valores observados y los valores estimados en el paso de tiempo anterior. Estos componentes se calculan utilizando fórmulas recursivas y se combinan para producir las predicciones futuras.

Existen dos variantes principales del método Holt-Winters: aditiva y multiplicativa, que se eligen en función de la naturaleza de la estacionalidad en los datos.

1. Holt-Winters Aditivo:

En esta variante, se asume que la estacionalidad tiene un efecto aditivo constante en la serie temporal. Los pasos para calcular las predicciones son los siguientes:

- Actualización del nivel (level):

Nivel actual = Alfa \* (Valor actual - Estacionalidad anterior) + (1 - Alfa) \* (Nivel anterior + Tendencia anterior)

- Actualización de la tendencia (trend):

Tendencia actual = Beta \* (Nivel actual - Nivel anterior) + (1 - Beta) \* Tendencia anterior

- Actualización de la estacionalidad (seasonality):

Estacionalidad actual = Gamma \* (Valor actual - Nivel actual) + (1 - Gamma) \* Estacionalidad anterior

- Predicción futura:

Predicción futura = Nivel actual + Tendencia actual + Estacionalidad actual

2. Holt-Winters Multiplicativo:

En esta variante, se asume que la estacionalidad tiene un efecto multiplicativo constante en la serie temporal. Los pasos para calcular las predicciones son similares a los de Holt-Winters aditivo, pero se utiliza una multiplicación en lugar de una suma en la actualización del nivel y la predicción futura.

El método de Holt-Winters es particularmente útil cuando se trabaja con series temporales que muestran tanto una tendencia como una estacionalidad, como los datos financieros y económicos. Permite capturar y modelar estos patrones, proporcionando predicciones más precisas.

Al aplicar el método de Holt-Winters, es necesario ajustar los parámetros Alfa, Beta y Gamma para obtener las mejores predicciones. Esto se puede lograr utilizando técnicas de optimización o validación cruzada en diferentes conjuntos de datos. Además, es importante tener en cuenta que el método Holt-Winters asume que los patrones de tendencia y estacionalidad son estables a lo largo del tiempo y puede no ser adecuado para series con cambios bruscos o patrones irregulares.

Y tomando en cuenta nuestros datos, los cuales pueden contener cambios bruscos y patrones irregulares, podemos decir que incluso esta técnica no es una buena aproximación a los datos de criptoactivos. Sin embargo podríamos probarla.

##### 1.1.2.4.- State Space Models:

State Space Models (Modelos de Espacio de Estados), también conocidos como modelos de espacio de estados ocultos (Hidden Markov Models), son una clase de modelos estadísticos utilizados para describir y predecir series temporales. Estos modelos son particularmente útiles cuando se trabaja con datos que exhiben una estructura dinámica compleja y están sujetos a incertidumbre.

En un State Space Model, se considera que una serie temporal es el resultado de dos procesos diferentes:

1. Proceso oculto (Hidden process): Representa la dinámica subyacente o no observada de la serie temporal. Este proceso se modela mediante una secuencia de estados ocultos que evolucionan a lo largo del tiempo. Cada estado oculto representa una combinación de factores que determina el comportamiento de la serie temporal.

2. Proceso observable (Observable process): Representa las observaciones o mediciones realizadas en el tiempo. Estas observaciones están relacionadas con los estados ocultos a través de una función de observación que puede ser determinística o probabilística.

Un State Space Model se compone de dos ecuaciones principales:

1. Ecuación de estado (State equation): Describe cómo evolucionan los estados ocultos a lo largo del tiempo. Puede ser un sistema de ecuaciones lineales o no lineales que relaciona el estado en el tiempo t con el estado en el tiempo t-1 y posiblemente con otras variables externas.

2. Ecuación de observación (Observation equation): Relaciona las observaciones en el tiempo t con los estados ocultos en el mismo tiempo t. Esta ecuación puede ser lineal o no lineal y puede contener términos de ruido o error.

La estimación de los parámetros en un State Space Model generalmente se realiza mediante técnicas de inferencia estadística, como el filtro de Kalman o el algoritmo de Expectation-Maximization (EM). Estas técnicas permiten estimar los estados ocultos y los parámetros del modelo en función de las observaciones disponibles.

Los State Space Models son ampliamente utilizados en diferentes aplicaciones, como pronóstico de series temporales, detección de anomalías, modelado de sistemas dinámicos y reconocimiento de patrones. Son especialmente útiles cuando se trabaja con datos que presentan estructuras complejas y cambios en el tiempo, lo que los hace aplicables en áreas como finanzas, economía, procesamiento de señales y muchas otras.

Bajo estas condiciones, es esta técnica la que mejor se ajusta a la naturaleza de nuestros datos. Además de ello, también se puede aplicar en el área de detección de anomalías y en reconocimiento de patrones. Lo que nos permitiría reutilizar el código.

Algunos modelos de este tipo son los siguientes:

1.- Modelo de espacio de estados con volatilidad estocástica (Stochastic Volatility State Space Model): Este modelo se utiliza para modelar la volatilidad de los precios en el criptotrading. Incorpora un componente oculto que representa la volatilidad, que puede variar a lo largo del tiempo de manera estocástica. Este modelo ayuda a capturar la naturaleza cambiante de la volatilidad en los mercados de criptomonedas.

2.- Modelo de espacio de estados con cambio de régimen (Switching State Space Model): Este modelo se utiliza para identificar y modelar diferentes regímenes o estados en los precios de las criptomonedas. Puede ayudar a capturar cambios abruptos en la dinámica de los precios y ajustar el modelo en consecuencia.

3.- Modelo de espacio de estados con componentes de tendencia y estacionalidad (State Space Model with Trend and Seasonality): Este modelo se utiliza para modelar y predecir la tendencia y la estacionalidad en los precios de las criptomonedas. Puede incorporar componentes de tendencia lineal o no lineal, así como patrones estacionales recurrentes, lo que permite capturar los patrones cíclicos y de largo plazo en los precios.

4.- Modelo de espacio de estados con saltos (Jump State Space Model): Este modelo se utiliza para modelar eventos de salto o cambios bruscos en los precios de las criptomonedas. Los saltos pueden ser causados por eventos importantes como noticias, anuncios regulatorios o cambios en la percepción del mercado. Este modelo ayuda a capturar y modelar estos eventos de manera efectiva.

5.- Modelo de espacio de estados con información externa (State Space Model with External Information): Este modelo se utiliza para incorporar información externa relevante en el análisis de criptomonedas, como datos económicos, noticias o indicadores financieros. Puede ayudar a mejorar la precisión de las predicciones al aprovechar información adicional que pueda afectar los precios de las criptomonedas.

6.- Modelo de espacio de estados de volatilidad heterocedástica condicional (GARCH): El modelo GARCH es utilizado para modelar la volatilidad de los precios en el criptotrading. A diferencia de los modelos de volatilidad estocástica mencionados anteriormente, el GARCH asume que la volatilidad es condicionalmente heterocedástica, lo que significa que puede variar con el tiempo y estar correlacionada con los errores pasados.

7.- Modelo de espacio de estados con cambio de nivel (Level Shift State Space Model): Este modelo se utiliza para detectar y modelar cambios de nivel en los precios de las criptomonedas. Puede ser útil cuando hay eventos significativos que provocan cambios permanentes en los precios, como cambios en la política regulatoria o anuncios importantes.

8.- Modelo de espacio de estados con cambio de pendiente (Slope Change State Space Model): Este modelo se utiliza para detectar y modelar cambios en la pendiente o la tendencia de los precios de las criptomonedas. Puede ser útil para identificar cambios en la dirección de la tendencia y ajustar el modelo en consecuencia.

9.- Modelo de espacio de estados de cambio de régimen oculto (Hidden Regime Switching State Space Model): Este modelo es similar al modelo de cambio de régimen mencionado anteriormente, pero incorpora un proceso oculto adicional que indica el régimen o estado actual del mercado. Puede ser útil para capturar cambios no observables en los precios y ajustar el modelo de manera adecuada.

Estos son solo algunos ejemplos de modelos de espacio de estados aplicados al criptotrading.

Ahora veamos una tabla de ellos ordenados por importancia y por más usados:

| **Más utilizados en criptotrading** | **Mayor potencial de ganancias** |
| --- | --- |
| Modelo de espacio de estados lineal (Linear State Space Model) | Modelo de espacio de estados con cambio de régimen (Switching State Space Model) |
| Modelo de espacio de estados no lineal (Nonlinear State Space Model) | Modelo de espacio de estados con volatilidad estocástica (Stochastic Volatility State Space Model) |
| Modelo de espacio de estados con cambio de régimen (Switching State Space Model) | Modelo de espacio de estados no lineal (Nonlinear State Space Model) |
| Modelo de espacio de estados con volatilidad estocástica (Stochastic Volatility State Space Model) | Modelo de espacio de estados con cambio de nivel (Level Shift State Space Model) |
| Modelo de espacio de estados con cambio de nivel (Level Shift State Space Model) | Modelo de espacio de estados lineal (Linear State Space Model) |
| Modelo de espacio de estados con cambio de pendiente (Slope Change State Space Model) | Modelo de espacio de estados con cambio de pendiente (Slope Change State Space Model) |
| Modelo de espacio de estados de cambio de régimen oculto (Hidden Regime Switching State Space Model) | Modelo de espacio de estados de cambio de régimen oculto (Hidden Regime Switching State Space Model) |
| Modelo de espacio de estados con ruido aditivo (State Space Model with Additive Noise) | Modelo de espacio de estados con ruido multiplicativo (State Space Model with Multiplicative Noise) |
| Modelo de espacio de estados con ruido multiplicativo (State Space Model with Multiplicative Noise) | Modelo de espacio de estados con ruido aditivo (State Space Model with Additive Noise) |

Va a ser importante enfocarnos en esta técnica puesto que se ve prometedora con respecto a la descripción que se dio más arriba.

Aquí me gustaría parar para decir lo siguiente:

Veo un poco a las regresiones como un nivel básico de predicción, el cual muy probablemente no sea suficiente para obtener grandes ganancias pero sí tal vez un retorno moderado como el que se espera de un banco o instituciones similares, o quizá incluso pérdidas si no se desarrollan bien.

Por su parte, veo mayor potencial de retorno en los modelos de series temporales, los cuales hasta ahora han sido más complejos, como podemos comparar de las descripciones de ambas técnicas. Esto porque claramente abordan mejor la naturaleza de nuestros datos que las regresiones.

Y aunque no hemos llegado a esa sección, es muy probable que las redes neuronales tengan más potencial de retorno que las dos técnicas anteriores.

Sin embargo, lo que yo pienso más va a ser rentable será la combinación adecuada de herramientas de estas 3 técnicas. Siendo posible agregar una red neuronal a los resultados de las regresiones y las series temporales. Aunque puede que no mejore mucho la rentabilidad, pero ya lo veremos poco a poco con el desarrollo paulatino de estas áreas.

Ahora veamos algunos de los modelos mencionados en la tabla:

###### 1.1.2.4.1.- Modelo de Espacio de Estados Lineal:

El Modelo de Espacio de Estados Lineal (Linear State Space Model) es un enfoque estadístico utilizado para modelar series temporales y otras secuencias de datos. Se basa en la idea de descomponer la serie temporal en un estado no observado (oculto) y un proceso de observación.

En este modelo, se asume que el sistema está gobernado por un conjunto de ecuaciones lineales. El estado no observado evoluciona en el tiempo de acuerdo con una ecuación de transición, mientras que las observaciones se generan a partir del estado oculto a través de una ecuación de observación. Estas ecuaciones están definidas por parámetros que se pueden estimar a partir de los datos.

La ecuación de transición describe cómo evoluciona el estado no observado a lo largo del tiempo. Puede haber una relación lineal entre el estado en el tiempo actual y el estado en el tiempo anterior, así como la influencia de otros factores. Esta ecuación puede tener en cuenta cambios de nivel, tendencia, estacionalidad u otros patrones que se observen en los datos.

La ecuación de observación describe cómo las observaciones se generan a partir del estado oculto. Generalmente, se asume una relación lineal entre el estado y las observaciones, pero también pueden incorporarse otros factores como ruido o errores.

Una vez que se ha definido el modelo de espacio de estados lineal, se utiliza un algoritmo de filtrado, como el filtro de Kalman, para estimar los estados ocultos en cada punto de tiempo y obtener estimaciones de las variables de interés.

Este modelo es útil en el criptotrading ya que permite capturar patrones lineales en los datos y ajustarse a diferentes condiciones del mercado. Se puede utilizar para realizar predicciones, detección de anomalías y análisis de tendencias en los precios de las criptomonedas.

Es importante destacar que el Modelo de Espacio de Estados Lineal asume linealidad en las relaciones y puede no ser adecuado si los datos presentan patrones no lineales.

Aquí cabe recalcar que este modelo puede no ser de utilidad en periodos largos, pero puede funcionar en periodos cortos de tiempo para ofrecernos una aproximación más ajustada a lo real en el corto plazo. Lo cual podría combinarse bien con los siguientes modelos que incorporan otros factores que representan mejor a la naturaleza de nuestros datos.

El código de este método sería el siguiente:

import numpy as np

import matplotlib.pyplot as plt

from statsmodels.tsa.statespace import tools

from statsmodels.tsa.statespace import representation

# Generar datos sintéticos

np.random.seed(0)

n = 200

t = np.arange(n)

y = np.random.normal(loc=0, scale=1, size=n)

for i in range(1, n):

y[i] = 0.8 \* y[i-1] + np.random.normal()

# Definir el modelo de espacio de estados lineal

class LinearStateSpaceModel(tools.FrozenRepresentation, representation.FrozenRepresentation):

import numpy as np

import matplotlib.pyplot as plt

from statsmodels.tsa.statespace import tools

from statsmodels.tsa.statespace import representation

# Generar datos sintéticos

np.random.seed(0)

n = 200

t = np.arange(n)

y = np.random.normal(loc=0, scale=1, size=n)

for i in range(1, n):

y[i] = 0.8 \* y[i-1] + np.random.normal()

# Definir el modelo de espacio de estados lineal

class LinearStateSpaceModel(tools.FrozenRepresentation, representation.FrozenRepresentation):

def \_\_init\_\_(self, endog):

super().\_\_init\_\_(endog, k\_states=1, k\_posdef=1, initialization='known')

def update(self, params, \*\*kwargs):

self['design', 0, 0] = params[0]

self['transition', 0, 0] = params[1]

self['selection', 0, 0] = params[2]

self['state\_cov', 0, 0] = params[3]

# Ajustar el modelo a los datos

model = LinearStateSpaceModel(y)

results = model.fit()

# Realizar predicciones

n\_forecast = 50

forecast = results.forecast(steps=n\_forecast)

# Plotear los datos y las predicciones

plt.figure(figsize=(10, 6))

plt.plot(t, y, label='Datos')

plt.plot(np.arange(n, n + n\_forecast), forecast, label='Predicción')

plt.legend()

plt.xlabel('Tiempo')

plt.ylabel('Valor')

plt.title('Modelo de Espacio de Estados Lineal')

plt.show()

En este ejemplo, se generan datos sintéticos utilizando un proceso AR(1). Luego, se define una clase LinearStateSpaceModel que hereda de las clases FrozenRepresentation y FrozenRepresentation de statsmodels. Esta clase representa el modelo de espacio de estados lineal y define los parámetros del modelo (diseño, transición, selección y covarianza del estado). A continuación, se ajusta el modelo a los datos utilizando el método fit() y se realizan predicciones para los próximos 50 pasos de tiempo utilizando el método forecast(). Por último, se plotean los datos y las predicciones.

El cual no se ha probado aún.

###### 1.1.2.4.2.- Modelo de Espacio de Estados con Cambio de Régimen:

El Modelo de Espacio de Estados con Cambio de Régimen (Switching State Space Model) es una técnica utilizada en el análisis de series temporales, incluyendo el criptotrading, para capturar cambios bruscos y no lineales en los datos. Este modelo permite identificar y modelar diferentes regímenes o estados en los que los datos pueden encontrarse, y ajustar los parámetros del modelo en consecuencia.

En este modelo, se asume que los datos observados están gobernados por un conjunto de ecuaciones de espacio de estados, donde el sistema se divide en múltiples regímenes. Cada régimen tiene su propio conjunto de ecuaciones y parámetros, lo que refleja diferentes dinámicas o comportamientos en los datos.

El modelo utiliza una variable latente o oculta llamada "estado de régimen" que indica en qué régimen se encuentra el sistema en cada período de tiempo. Esta variable puede tomar diferentes valores discretos que representan los diferentes regímenes. La transición entre los regímenes puede ser gobernada por un proceso estocástico, como una cadena de Markov, donde la probabilidad de cambio de régimen depende del régimen actual.

El Modelo de Espacio de Estados con Cambio de Régimen se estima mediante técnicas de estimación de máxima verosimilitud, donde se busca encontrar los parámetros óptimos del modelo que maximicen la probabilidad conjunta de los datos observados y los estados ocultos. La estimación de los parámetros se realiza a través de algoritmos de filtrado y suavizado, como el filtro de Kalman.

Una vez estimado el modelo, se puede utilizar para realizar predicciones y análisis. El modelo permite capturar cambios no lineales en los datos y adaptarse a diferentes condiciones del mercado. Por ejemplo, en el criptotrading, puede ayudar a identificar períodos de alta volatilidad o tendencias abruptas en los precios y ajustar las estrategias de trading en consecuencia.

Es importante destacar que la aplicación del Modelo de Espacio de Estados con Cambio de Régimen requiere un análisis cuidadoso de los datos y una selección adecuada de los regímenes y variables relevantes. Además, el rendimiento del modelo puede depender de la calidad de los datos y de las suposiciones realizadas. Por lo tanto, es recomendable realizar una evaluación rigurosa y considerar otras técnicas de modelado antes de utilizar este modelo en el criptotrading.

Un código sencillo, con la librería pykalman, sería el siguiente:

import numpy as np

import matplotlib.pyplot as plt

from pykalman import KalmanFilter

# Generar datos sintéticos

np.random.seed(0)

n = 200

t = np.arange(n)

y = np.random.normal(loc=0, scale=1, size=n)

for i in range(1, n):

y[i] = 0.8 \* y[i-1] + np.random.normal()

# Definir el modelo de espacio de estados con cambio de régimen

transition\_matrices = [[0.8, 0], [0, 0.9]]

transition\_covariance = np.eye(2)

observation\_matrices = np.array([[1, 0], [1, 0]])

observation\_covariance = np.eye(2)

initial\_state\_mean = [0, 0]

initial\_state\_covariance = np.eye(2)

kf = KalmanFilter(transition\_matrices=transition\_matrices,

transition\_covariance=transition\_covariance,

observation\_matrices=observation\_matrices,

observation\_covariance=observation\_covariance,

initial\_state\_mean=initial\_state\_mean,

initial\_state\_covariance=initial\_state\_covariance)

# Ajustar el modelo a los datos

filtered\_state\_means, filtered\_state\_covariances = kf.filter(y)

# Realizar predicciones

n\_forecast = 50

filtered\_state\_means\_forecast, filtered\_state\_covariances\_forecast = kf.filter(y, n\_steps\_ahead=n\_forecast)

# Obtener las predicciones y los intervalos de confianza

forecast\_means = filtered\_state\_means\_forecast[:, 0]

forecast\_std = np.sqrt(filtered\_state\_covariances\_forecast[:, 0, 0])

forecast\_ci\_upper = forecast\_means + 1.96 \* forecast\_std

forecast\_ci\_lower = forecast\_means - 1.96 \* forecast\_std

# Plotear los datos y las predicciones

plt.figure(figsize=(10, 6))

plt.plot(t, y, label='Datos')

plt.plot(np.arange(n, n + n\_forecast), forecast\_means, label='Predicción')

plt.fill\_between(np.arange(n, n + n\_forecast), forecast\_ci\_lower, forecast\_ci\_upper, color='gray', alpha=0.3, label='Intervalo de Confianza')

plt.legend()

plt.xlabel('Tiempo')

plt.ylabel('Valor')

plt.title('Modelo de Espacio de Estados con Cambio de Régimen')

plt.show()

En este ejemplo, se generan datos sintéticos utilizando un proceso AR(1). Luego, se define el modelo de espacio de estados con cambio de régimen utilizando los parámetros de transición y covarianza, matrices de observación y los valores iniciales. Se utiliza la clase KalmanFilter de la librería pykalman para construir el modelo. A continuación, se ajusta el modelo a los datos utilizando el método filter(), lo cual proporciona los estados filtrados (mean) y las covarianzas correspondientes. Se realizan predicciones utilizando el método filter() con el argumento n\_steps\_ahead para especificar la cantidad de pasos futuros a predecir. Finalmente, se plotean los datos originales, las predicciones y los intervalos de confianza.

Ahora veamos un código con la librería statsmodels:

import numpy as np

import matplotlib.pyplot as plt

import statsmodels.api as sm

# Generar datos sintéticos

np.random.seed(0)

n = 200

t = np.arange(n)

y = np.random.normal(loc=0, scale=1, size=n)

for i in range(1, n):

y[i] = 0.8 \* y[i-1] + np.random.normal()

# Definir el modelo de espacio de estados con cambio de régimen

model = sm.tsa.MarkovRegression(y, k\_regimes=2)

# Ajustar el modelo a los datos

result = model.fit()

# Obtener la estimación de los estados latentes

filtered\_states = result.smoothed\_state

# Plotear los datos y los estados latentes

plt.figure(figsize=(10, 6))

plt.plot(t, y, label='Datos')

plt.plot(t, filtered\_states[:, 0], label='Estado 1')

plt.plot(t, filtered\_states[:, 1], label='Estado 2')

plt.legend()

plt.xlabel('Tiempo')

plt.ylabel('Valor')

plt.title('Modelo de Espacio de Estados con Cambio de Régimen')

plt.show()

En este ejemplo, se generan datos sintéticos utilizando un proceso AR(1). Luego, se utiliza la clase MarkovRegression de statsmodels.tsa para definir el modelo de espacio de estados con cambio de régimen. El argumento k\_regimes=2 especifica que el modelo tiene dos regímenes. A continuación, se ajusta el modelo a los datos utilizando el método fit(). Se obtiene la estimación de los estados latentes utilizando el atributo smoothed\_state del resultado del ajuste. Finalmente, se plotean los datos originales y los estados latentes a lo largo del tiempo.

Los códigos no se han probado.

###### 1.1.2.4.3.- Modelo de Espacio de Estados con Volatilidad Estocástica:

El Modelo de Espacio de Estados con Volatilidad Estocástica (Stochastic Volatility State Space Model en inglés) es un enfoque utilizado para modelar y estimar la volatilidad de una serie de datos a lo largo del tiempo. Este modelo es particularmente útil en el análisis financiero, incluido el criptotrading, donde la volatilidad es una medida esencial para evaluar los riesgos y las oportunidades.

En este modelo, se asume que la volatilidad de la serie de datos cambia con el tiempo y está influenciada por un proceso estocástico. La volatilidad se modela como una serie no observable o latente, mientras que los datos observados se modelan como una función de la volatilidad latente.

El Modelo de Espacio de Estados con Volatilidad Estocástica se define por las siguientes ecuaciones:

1. Ecuación de Observación:

Y\_t = Z\_t \* alpha\_t + epsilon\_t

Donde Y\_t es la observación en el tiempo t, Z\_t es la matriz de diseño que relaciona la volatilidad latente con las observaciones, alpha\_t es el vector de estados latentes en el tiempo t y epsilon\_t es el término de error de observación.

2. Ecuación de Transición:

alpha\_t = T\_t \* alpha\_{t-1} + R\_t \* eta\_t

Donde T\_t es la matriz de transición que describe cómo evolucionan los estados latentes en el tiempo, R\_t es la matriz de selección que describe cómo la volatilidad latente se ve afectada por el proceso estocástico y eta\_t es el término de error de transición.

3. Ecuación de la Volatilidad:

sigma\_t^2 = exp(Q\_t)

Donde sigma\_t^2 es la varianza de la volatilidad latente en el tiempo t y Q\_t es el estado latente de la volatilidad.

El objetivo principal del modelo es estimar los estados latentes y la volatilidad no observada en función de los datos observados. Esto se puede lograr utilizando técnicas de estimación como el filtro de Kalman, el algoritmo de suavizado y el algoritmo de máxima verosimilitud.

Al estimar el Modelo de Espacio de Estados con Volatilidad Estocástica, se obtiene una estimación de la volatilidad latente a lo largo del tiempo, lo que permite realizar pronósticos más precisos y capturar mejor la naturaleza cambiante de la volatilidad en el criptotrading.

Es importante tener en cuenta que la implementación detallada del modelo puede variar según la biblioteca o el paquete que se utilice. Existen varias implementaciones en Python, como `statsmodels` y `pykalman`, que proporcionan funciones y clases específicas para estimar y analizar modelos de espacio de estados con volatilidad estocástica.

Un código sencillo, con la librería pykalman, sería el siguiente:

import numpy as np

import matplotlib.pyplot as plt

from pykalman import KalmanFilter

# Generar datos sintéticos

np.random.seed(0)

n = 200

t = np.arange(n)

y = np.random.normal(loc=0, scale=1, size=n)

for i in range(1, n):

y[i] = 0.8 \* y[i-1] + np.random.normal()

# Definir el modelo de espacio de estados con volatilidad estocástica

transition\_matrices = [[0.8]]

transition\_covariance = [[0.1]]

observation\_matrices = [[1]]

observation\_covariance = [[1]]

initial\_state\_mean = [0]

initial\_state\_covariance = [[1]]

n\_dim\_state = 1

kf = KalmanFilter(transition\_matrices=transition\_matrices,

observation\_matrices=observation\_matrices,

transition\_covariance=transition\_covariance,

observation\_covariance=observation\_covariance,

initial\_state\_mean=initial\_state\_mean,

initial\_state\_covariance=initial\_state\_covariance,

n\_dim\_state=n\_dim\_state)

# Ajustar el modelo a los datos

filtered\_state\_means, filtered\_state\_covariances = kf.filter(y)

# Obtener la volatilidad estimada

volatility = np.sqrt(filtered\_state\_covariances[:, 0, 0])

# Plotear los datos y la volatilidad estimada

plt.figure(figsize=(10, 6))

plt.plot(t, y, label='Datos')

plt.plot(t, volatility, label='Volatilidad Estimada')

plt.legend()

plt.xlabel('Tiempo')

plt.ylabel('Valor')

plt.title('Modelo de Espacio de Estados con Volatilidad Estocástica')

plt.show()

En este ejemplo, se generan datos sintéticos utilizando un proceso AR(1). Luego, se define el modelo de espacio de estados con volatilidad estocástica utilizando los parámetros de transición y covarianza, matrices de observación y los valores iniciales. Se utiliza la clase KalmanFilter de la biblioteca pykalman para construir el modelo. A continuación, se ajusta el modelo a los datos utilizando el método filter(), lo cual proporciona los estados filtrados (mean) y las covarianzas correspondientes. Se obtiene la volatilidad estimada extrayendo la desviación estándar de la covarianza del estado latente. Finalmente, se plotean los datos originales y la volatilidad estimada a lo largo del tiempo.

Ahora veamos un código con la librería statsmodels:

import numpy as np

import matplotlib.pyplot as plt

import statsmodels.api as sm

# Generar datos sintéticos

np.random.seed(0)

n = 200

t = np.arange(n)

y = np.random.normal(loc=0, scale=1, size=n)

for i in range(1, n):

y[i] = 0.8 \* y[i-1] + np.random.normal()

# Definir el modelo de espacio de estados con volatilidad estocástica

model = sm.tsa.UnobservedComponents(y, 'rwalk', stochastic\_level=True)

# Ajustar el modelo a los datos

result = model.fit()

# Obtener la volatilidad estimada

volatility = result.level.apply(np.sqrt)

# Plotear los datos y la volatilidad estimada

plt.figure(figsize=(10, 6))

plt.plot(t, y, label='Datos')

plt.plot(t, volatility, label='Volatilidad Estimada')

plt.legend()

plt.xlabel('Tiempo')

plt.ylabel('Valor')

plt.title('Modelo de Espacio de Estados con Volatilidad Estocástica')

plt.show()

En este ejemplo, también se generan datos sintéticos utilizando un proceso AR(1). Luego, se utiliza la clase UnobservedComponents de statsmodels.tsa para definir el modelo de espacio de estados con volatilidad estocástica. El argumento 'rwalk' especifica que la tendencia es un paseo aleatorio y se establece stochastic\_level=True para modelar la volatilidad como un componente estocástico. A continuación, se ajusta el modelo a los datos utilizando el método fit(). Se obtiene la volatilidad estimada a través del atributo level del resultado del ajuste y se calcula la raíz cuadrada para obtener la desviación estándar de la volatilidad. Finalmente, se plotean los datos originales y la volatilidad estimada a lo largo del tiempo.

Este método parece que no realiza predicciones de precios, sino más bien analiza la volatilidad del mercado. Habrá que ver si en realidad nos funciona el método.

Los códgos no se han probado.

###### 1.1.2.4.4.- Modelo de espacio de estados no lineal:

El Modelo de Espacio de Estados No Lineal (Nonlinear State Space Model en inglés) es una técnica de modelado que permite capturar relaciones no lineales entre las variables observadas y las variables latentes en un sistema. A diferencia de los modelos lineales de espacio de estados, que asumen relaciones lineales entre las variables, los modelos no lineales permiten representar de manera más flexible las interacciones complejas y no lineales presentes en muchos fenómenos.

En un Modelo de Espacio de Estados No Lineal, las variables observadas y las variables latentes se relacionan mediante funciones no lineales. Estas funciones no lineales pueden ser cualquier función matemática que refleje la relación entre las variables. Por ejemplo, se pueden utilizar funciones polinómicas, exponenciales, trigonométricas u otras funciones no lineales para modelar las relaciones.

La formulación general de un Modelo de Espacio de Estados No Lineal se puede expresar como:

- Equación de estado:

x(t) = f(x(t-1), u(t), θ) + ε(t)

- Equación de observación:

y(t) = g(x(t), θ) + δ(t)

Donde:

- x(t) es el vector de variables latentes en el tiempo t.

- f() es una función no lineal que describe la evolución de las variables latentes en el tiempo.

- u(t) es el vector de entradas externas en el tiempo t.

- θ es el vector de parámetros del modelo.

- ε(t) es el término de error en la ecuación de estado.

- y(t) es el vector de variables observadas en el tiempo t.

- g() es una función no lineal que relaciona las variables latentes con las variables observadas.

- δ(t) es el término de error en la ecuación de observación.

El objetivo en el Modelado de Espacio de Estados No Lineal es estimar los valores de las variables latentes x(t) y los parámetros θ a partir de las observaciones y. Esto se logra mediante técnicas de estimación y ajuste de los parámetros, como el método de máxima verosimilitud o el filtro de Kalman extendido.

Una vez que se ha estimado el Modelo de Espacio de Estados No Lineal, se pueden utilizar para predecir valores futuros de las variables observadas y analizar el comportamiento del sistema en función de los valores de las variables latentes.

Es importante tener en cuenta que la estimación y el ajuste de los Modelos de Espacio de Estados No Lineal pueden ser computacionalmente intensivos debido a la naturaleza no lineal de las funciones involucradas. Por lo tanto, es común utilizar métodos numéricos y algoritmos de optimización para encontrar los mejores valores de los parámetros y realizar las predicciones.

En resumen, el Modelo de Espacio de Estados No Lineal es una poderosa herramienta de modelado que permite capturar relaciones no lineales en sistemas dinámicos. Su flexibilidad y capacidad para modelar fenómenos complejos lo hacen adecuado para una amplia gama de aplicaciones, incluido el análisis y la predicción en el campo del criptotrading.

Vemos que este método puede ser costoso computacionalmente, sin embargo, también vemos que es una buena opción de predicción en sistemas no lineales, siendo nuestros datos de naturaleza no lineal. Sería buena opción a la hora de la predicción.

Un códifo básico, con la librería pykalman, sería como sigue:

import numpy as np

from pykalman import UnscentedKalmanFilter

# Definir la función no lineal para la evolución del estado

def state\_transition\_function(state, noise):

return state + 0.1 \* state\*\*2 + noise

# Definir la función no lineal para la observación del estado

def observation\_function(state, noise):

return state\*\*3 + noise

# Generar datos sintéticos

np.random.seed(0)

n = 100

state = np.zeros(n)

observations = np.zeros(n)

for i in range(1, n):

state[i] = state\_transition\_function(state[i-1], np.random.normal(0, 0.1))

observations[i] = observation\_function(state[i], np.random.normal(0, 1))

# Definir el modelo de espacio de estados no lineal

kf = UnscentedKalmanFilter(

state\_transition\_func=state\_transition\_function,

observation\_func=observation\_function

)

# Ajustar el modelo a los datos

filtered\_state\_estimates, \_ = kf.filter(observations)

# Imprimir los resultados

print("Estimaciones de estado filtradas:")

print(filtered\_state\_estimates)

En este ejemplo, se define una función no lineal para la evolución del estado (state\_transition\_function) y una función no lineal para la observación del estado (observation\_function). Estas funciones representan las relaciones no lineales entre las variables latentes y las variables observadas en el modelo.

Luego, se generan datos sintéticos utilizando estas funciones y ruido gaussiano. A continuación, se crea una instancia de la clase UnscentedKalmanFilter de pykalman, pasando las funciones de evolución del estado y observación del estado como argumentos.

Finalmente, se ajusta el modelo a los datos utilizando el método filter y se obtienen las estimaciones filtradas de los estados latentes.

Recuerda que este es solo un ejemplo básico y que la implementación del Modelo de Espacio de Estados No Lineal puede variar según tus necesidades y los datos específicos que estés utilizando. Asegúrate de consultar la documentación de pykalman para obtener más información sobre los parámetros y opciones disponibles.

El código no se ha probado.

###### 1.1.2.4.5.- Modelo de Espacio de Estados con Cambio de Nivel:

Este es el último ejemplo de los “state space models”

El Modelo de Espacio de Estados con Cambio de Nivel es una técnica utilizada para modelar series de tiempo que presentan cambios abruptos o cambios estructurales en su nivel a lo largo del tiempo. Este modelo permite capturar y estimar de manera flexible estos cambios, lo que lo hace útil en el análisis de datos de series temporales que exhiben comportamientos no estacionarios.

El modelo se compone de dos componentes principales: el componente de evolución del estado y el componente de observación. El componente de evolución del estado describe cómo evoluciona el estado latente a lo largo del tiempo, mientras que el componente de observación relaciona el estado latente con las observaciones disponibles.

El componente de evolución del estado se define mediante la siguiente ecuación:

S\_t = S\_{t-1} + \beta D\_t + \epsilon\_t

donde S\_t es el estado latente en el tiempo t, S\_{t-1} es el estado latente en el tiempo t-1, \beta es el cambio de nivel, D\_t es una variable binaria que indica si hay un cambio de nivel en el tiempo t, y \epsilon\_t es el ruido de evolución del estado.

El componente de observación se define mediante la siguiente ecuación:

Y\_t = S\_t + \eta\_t

donde Y\_t es la observación en el tiempo t y \eta\_t es el ruido de observación.

El modelo se estima utilizando métodos de estimación de máxima verosimilitud o métodos bayesianos. Los parámetros estimados incluyen el cambio de nivel (\beta), los estados latentes iniciales, los ruidos de evolución y de observación, y la distribución inicial de los estados latentes.

Una vez que el modelo ha sido ajustado a los datos, se pueden obtener las estimaciones de los estados latentes, lo que proporciona información sobre la tendencia y los cambios estructurales en la serie de tiempo. También se pueden realizar predicciones futuras utilizando el modelo ajustado.

En resumen, el Modelo de Espacio de Estados con Cambio de Nivel es una herramienta útil para modelar series de tiempo con cambios abruptos en su nivel. Permite capturar y estimar estos cambios, lo que proporciona información valiosa sobre la dinámica y las tendencias en los datos de series temporales.

Un código básico sería el siguiente:

import numpy as np

import matplotlib.pyplot as plt

import statsmodels.api as sm

# Generar datos sintéticos con cambio de nivel

np.random.seed(0)

n = 200

time = np.arange(n)

level\_change\_point = 100

level1 = 10

level2 = 15

noise = np.random.normal(0, 1, n)

observations = np.concatenate([np.repeat(level1, level\_change\_point), np.repeat(level2, n - level\_change\_point)]) + noise

# Definir el modelo de espacio de estados con cambio de nivel

model = sm.tsa.UnobservedComponents(observations, 'llevel')

# Ajustar el modelo a los datos

result = model.fit()

# Obtener las estimaciones de los estados latentes

filtered\_state = result.filtered\_state

# Plotear los datos y las estimaciones de nivel

plt.plot(time, observations, label='Observaciones')

plt.plot(time, filtered\_state.filtered\_state[0], label='Estimaciones de nivel')

plt.axvline(x=level\_change\_point, color='red', linestyle='--', label='Punto de cambio de nivel')

plt.xlabel('Tiempo')

plt.ylabel('Valor')

plt.legend()

plt.show()

En este ejemplo, se generan datos sintéticos que presentan un cambio de nivel en el punto de tiempo 100. Luego, se define el modelo de espacio de estados con cambio de nivel utilizando la clase UnobservedComponents de statsmodels. El argumento 'llevel' indica que se está modelando un cambio de nivel en el nivel de la serie de tiempo.

A continuación, el modelo se ajusta a los datos utilizando el método fit(). Las estimaciones de los estados latentes se obtienen a través del atributo filtered\_state.filtered\_state.

Finalmente, se traza el gráfico de las observaciones originales y las estimaciones del nivel, destacando el punto de cambio de nivel con una línea roja discontinua.

##### 1.1.2.5.- GARCH

El modelo GARCH (Generalized Autoregressive Conditional Heteroskedasticity) es una técnica utilizada en el análisis de series temporales para modelar y predecir la volatilidad de los datos. Fue propuesto por Robert Engle en 1982 como una extensión del modelo ARCH (Autoregressive Conditional Heteroskedasticity).

El modelo GARCH es especialmente útil cuando los datos presentan volatilidad heteroscedástica, es decir, cuando la varianza de los errores no es constante a lo largo del tiempo. Este modelo permite capturar y modelar esta variación en la volatilidad a través del tiempo.

La ecuación general del modelo GARCH(p, q) se puede expresar de la siguiente manera:

σ²\_t = ω + ∑(α\_i \* ε²\_{t-i}) + ∑(β\_j \* σ²\_{t-j})

Donde:

- σ²\_t es la varianza condicional en el tiempo t.

- ω es el parámetro de constante.

- α\_i son los parámetros de autorregresión de los errores al cuadrado.

- ε²\_{t-i} son los errores al cuadrado en el tiempo t-i.

- β\_j son los parámetros de autorregresión de las varianzas condicionales.

- σ²\_{t-j} son las varianzas condicionales en el tiempo t-j.

El modelo GARCH(p, q) se caracteriza por tener un componente autoregresivo (AR) para los errores al cuadrado y otro componente autoregresivo (AR) para las varianzas condicionales. Los parámetros α\_i y β\_j controlan la contribución de los errores pasados y las varianzas pasadas, respectivamente, en la volatilidad actual. El parámetro ω representa la varianza constante.

Para estimar los parámetros del modelo GARCH, se utiliza generalmente el método de máxima verosimilitud. Este método busca encontrar los valores de los parámetros que maximizan la probabilidad de obtener los datos observados dados los parámetros del modelo.

Una vez que el modelo GARCH ha sido estimado, se pueden obtener las estimaciones de la volatilidad condicional para cada período de tiempo. Estas estimaciones proporcionan información valiosa sobre la incertidumbre y la variabilidad en los datos. Además, se pueden realizar pronósticos de la volatilidad futura utilizando el modelo ajustado.

El modelo GARCH ha demostrado ser útil en una variedad de aplicaciones, incluyendo el análisis de mercados financieros, donde la volatilidad es un factor importante a tener en cuenta en la toma de decisiones de inversión. Al modelar y predecir la volatilidad, el modelo GARCH puede ayudar a los inversores a evaluar el riesgo y ajustar sus estrategias de inversión en consecuencia.

Como podemos ver, este modelo solo presenta un análisis sobre la volatilidad del mercado, lo cual puede ayudar a la gestión de riesgos, pero no parece ser un buen candidato para la predicción de precios, sino solo en la predicción de la volatilidad futura.

Un código sencillo de esto sería:

import pandas as pd

from arch import arch\_model

# Cargar los datos de volatilidad (por ejemplo, precios históricos)

data = pd.read\_csv('ruta\_del\_archivo.csv', index\_col='Fecha', parse\_dates=True)

# Crear el modelo GARCH(1, 1)

model = arch\_model(data['Volatilidad'], vol='Garch', p=1, q=1)

# Ajustar el modelo a los datos

model\_fit = model.fit()

# Obtener las predicciones de volatilidad condicional

predictions = model\_fit.conditional\_volatility

# Imprimir las predicciones de volatilidad

print(predictions)

El modelo GARCH se crea utilizando arch\_model de la librería arch. En este ejemplo, se especifica vol='Garch' para indicar que queremos un modelo GARCH. Los parámetros p=1 y q=1 indican que estamos utilizando un modelo GARCH(1, 1).

Después de ajustar el modelo a los datos utilizando fit(), podemos acceder a las predicciones de volatilidad condicional utilizando conditional\_volatility. Estas predicciones representan la volatilidad estimada para cada observación en los datos.

##### 1.1.2.6.- Prophet

Prophet no es un método de ML, sino una librería creada por facebook para el análisis de series temporales. Como tal, vamos a dar una descripción breve a continuación sobre esta librería. Posteriormente voy a dejar una ruta de trabajo recomendada para la librería.

Prophet fue diseñada específicamente para ser fácil de usar y brindar resultados rápidos y precisos en la predicción de datos de series de tiempo. Aunque Prophet no fue creado específicamente para el criptotrading, puede ser utilizado para hacer pronósticos de precios de criptomonedas.

A continuación, se describen algunos aspectos clave de Prophet y cómo se puede utilizar para el criptotrading:

Modelo aditivo: Prophet asume que la serie de tiempo es una combinación aditiva de diferentes componentes, incluyendo la tendencia, las estacionalidades y los efectos de vacaciones. Esta característica permite capturar diferentes patrones presentes en los datos de criptomonedas.

Componentes principales: Prophet descompone la serie de tiempo en sus componentes principales, es decir, la tendencia y las estacionalidades. Estos componentes se modelan por separado y luego se combinan para generar las predicciones.

Detección automática de cambios: Prophet tiene la capacidad de detectar automáticamente cambios en la tendencia y los efectos de estacionalidad, lo que es especialmente útil en el criptotrading, donde los precios pueden experimentar cambios bruscos y repentinos.

Flexibilidad en los datos de entrada: Prophet acepta como entrada un DataFrame de Pandas con dos columnas obligatorias: 'ds' (las fechas) y 'y' (los valores de la serie de tiempo). Esto facilita el proceso de preparación de los datos antes de aplicar el modelo.

Personalización de los modelos: Prophet proporciona la posibilidad de personalizar los modelos ajustando diferentes hiperparámetros, como la estacionalidad, la suavidad de la tendencia y la sensibilidad a los cambios. Esto permite adaptar el modelo a las características específicas de los datos de criptomonedas.

Visualización de resultados: Prophet ofrece herramientas para visualizar los resultados de las predicciones, incluyendo gráficos interactivos con los datos históricos y las predicciones futuras. Esto ayuda a los traders a comprender y evaluar los resultados de manera intuitiva.

En resumen, Prophet es una herramienta útil para el criptotrading que simplifica el proceso de análisis y predicción de series de tiempo. Puede ayudar a los traders a identificar patrones, tendencias y cambios en los precios de las criptomonedas, brindando información valiosa para la toma de decisiones de inversión. Sin embargo, es importante utilizar Prophet en conjunto con otras técnicas y análisis para obtener una visión más completa y precisa del mercado de criptomonedas.

Ahora veamos la ruta de trabajo...

1. Recopilación de datos: Obtén los datos históricos de precios de la criptomoneda que deseas analizar. Puedes obtener estos datos de diversas fuentes, como exchanges de criptomonedas o servicios de datos financieros.

2. Preparación de datos: Prepara los datos en un formato adecuado para Prophet. Debes tener una columna con las fechas ('ds') y otra columna con los precios ('y'). Asegúrate de que los datos estén en formato de serie de tiempo y que no haya valores faltantes.

3. División de datos: Divide los datos en conjuntos de entrenamiento y prueba. Puedes utilizar una proporción como 80% para entrenamiento y 20% para prueba. Esto te permitirá evaluar el rendimiento del modelo en datos no vistos.

4. Creación del modelo: Crea un objeto de modelo Prophet utilizando la clase `Prophet` de la librería. Puedes ajustar diversos parámetros para personalizar el modelo, como la configuración de la estacionalidad, la inclusión de días festivos, entre otros.

5. Ajuste del modelo: Utiliza el método `fit` para ajustar el modelo a los datos de entrenamiento. Esto calculará los parámetros del modelo basados en los datos históricos.

6. Generación de predicciones: Utiliza el método `predict` para generar predicciones futuras del precio de la criptomoneda. Puedes especificar el número de períodos hacia adelante que deseas pronosticar.

7. Evaluación del modelo: Evalúa el rendimiento del modelo utilizando los datos de prueba. Compara las predicciones generadas por el modelo con los valores reales y calcula métricas de evaluación, como el error cuadrado medio (RMSE) o el error absoluto medio (MAE).

8. Visualización de resultados: Utiliza las herramientas de visualización de Prophet para representar gráficamente los datos históricos, las predicciones y los intervalos de confianza. Esto te ayudará a comprender la tendencia y la incertidumbre en las predicciones.

9. Implementación de estrategias de trading: Utiliza las predicciones generadas por el modelo para implementar estrategias de trading. Por ejemplo, puedes generar señales de compra o venta basadas en la dirección de las predicciones y establecer reglas específicas para tomar decisiones comerciales.

Es recomendable utilizar Prophet junto con otras técnicas de análisis y considerar factores adicionales, como el análisis fundamental y las condiciones del mercado, al tomar decisiones de trading.

A mi parecer, esta librería podría llegar a ser muy rígida con respecto a lo que necesitemos, sin embargo, también podría tener herramientas útiles que pudiéramos utilizar en un futuro, por lo que sería buena idea preguntarnos si vale la pena investigarla y decidir si dejarla de lado o no.

#### 1.1.3.- Redes Neuronales:

Ahora vamos a ver a las redes neuronales aplicadas a la predicción de precios en el criptotrading.

Las redes neuronales son un enfoque popular y poderoso en el campo del machine learning, y se han utilizado ampliamente para la predicción de precios en el criptotrading. Estas redes están inspiradas en el funcionamiento del cerebro humano y pueden aprender patrones complejos y no lineales en los datos.

En el contexto de la predicción de precios en el criptotrading, existen varias arquitecturas de redes neuronales que se utilizan con frecuencia. A continuación, describiré algunas de las más comunes:

1.- Redes Neuronales Recurrentes (RNN): Las RNN son una elección popular para la predicción de series temporales, ya que pueden capturar la dependencia secuencial en los datos. La arquitectura de una RNN permite que la información fluya hacia adelante y hacia atrás a través de las capas, lo que las hace adecuadas para modelar secuencias de datos, como los precios históricos en el criptotrading.

2.- Redes Neuronales de Memoria a Corto y Largo Plazo (LSTM): Las LSTM son un tipo especial de RNN que se diseñaron para abordar el problema del desvanecimiento o desaparición del gradiente en las RNN estándar. Las LSTM tienen una arquitectura que permite retener información a largo plazo y seleccionar qué información recordar y qué olvidar. Esto las hace especialmente efectivas para capturar patrones a largo plazo en los datos de precios criptográficos.

3.- Redes Neuronales Convolucionales (CNN): Aunque las CNN son más comúnmente utilizadas en tareas de visión por computadora, también se han aplicado con éxito en la predicción de precios en el criptotrading. Las CNN son capaces de aprender características espaciales y temporales a través de capas convolucionales y de agrupación, lo que puede ser útil para identificar patrones en los datos de precios.

1. - Redes Neuronales Generativas Adversariales (GAN): Las GAN son una clase de redes neuronales que consisten en un generador y un discriminador que compiten entre sí. Estas redes se han utilizado para generar datos sintéticos realistas y también se han aplicado a la predicción de precios en el criptotrading. Las GAN pueden ser útiles para simular escenarios de precios futuros y generar señales de trading basadas en estos escenarios.

5.- Redes Neuronales de Onda Larga (WaveNet): WaveNet es una arquitectura de red neuronal que se utiliza principalmente en la generación de voz y audio, pero también se ha aplicado a la predicción de series temporales, incluidos los precios criptográficos. Esta arquitectura utiliza una estructura de red profunda y conexiones de retroalimentación para modelar dependencias a largo plazo en los datos.

6.- Redes Neuronales de Atención (Attention Networks): Las redes neuronales de atención se han utilizado en diversos problemas de procesamiento del lenguaje natural y también se han aplicado a la predicción de precios en el criptotrading. Estas redes permiten que el modelo se enfoque en partes específicas de la serie temporal relevante para la predicción, lo que puede mejorar el rendimiento y la interpretabilidad de las predicciones.

7.- Redes Neuronales Generativas Autoregresivas (Autoregressive Generative Networks): Estas redes neuronales se utilizan para modelar la distribución de los datos de series temporales y generar nuevas muestras. Se basan en la idea de que cada punto en la serie temporal depende de los puntos anteriores. Estas redes pueden ser utilizadas para generar escenarios futuros y evaluar el riesgo en el criptotrading.

8.- Redes Neuronales Transformers: Los Transformers son una arquitectura de red neuronal que se hizo popular en el campo del procesamiento del lenguaje natural, especialmente en tareas de traducción automática y modelado del lenguaje. También se han aplicado a la predicción de precios en el criptotrading, ya que pueden capturar relaciones de largo alcance en los datos de series temporales.

1. - GRU (Gated Recurrent Unit): Es una arquitectura de red neuronal recurrente utilizada en la predicción de series temporales, incluyendo el criptotrading. Es una variante de las redes LSTM con menos parámetros y una estructura más simple. Las GRU utilizan puertas de actualización y reinicio para controlar el flujo de información y capturar dependencias a largo plazo en los datos. Son capaces de modelar patrones complejos en series temporales y se pueden entrenar para predecir precios futuros en el criptotrading.

Vamos a ver cada una de estas arquitecturas un poco más a detalle en lo sucesivo, antes de pasar a la siguiente área del criptotrading, la cual es la gestión de riesgos. Vamos a dejar la última sección de la predicción de precios para más adelante con el fin de no hacer tan larga la sección sobre predicción de precios.

De hecho, aunque haya desarrollado ya un poco la investigación sobre redes neuronales, en la práctica, vamos a estar viendo un desarrollo que pase por estrategias de trading clásicas sin AI con su apropiado backtesting, primeramente y, posteriormente, por la implementación de técnicas de AI a los indicadores que hayamos encontrado útiles en el desarrollo de las estrategias clásicas, así como la aplicación de AI para encontrar nuevos indicadores clave y utilizar esos mismos indicadores.  
  
Evidentemente, tanto precio como volumen son indicadores importantes, pero puede haber más.

Ahora sigamos con las redes neuronales y sus arquitecturas. Me gustaría decir que no pondré código de las redes neuronales, este deberá introducirse en los documentos de investigación de cada arquitectura. Además, veremos las LSTM y las GRU como parte de la sección de RNN.

##### 1.1.3.1.- Redes Neuronales Recurrentes (RNN):

Las RNN (Redes Neuronales Recurrentes) son un tipo de arquitectura de redes neuronales que se utilizan ampliamente en la predicción de precios en el criptotrading. Estas redes son especialmente adecuadas para modelar datos secuenciales, como series de tiempo, debido a su capacidad para capturar dependencias temporales a largo plazo.

Las RNN funcionan procesando secuencialmente los datos de entrada a través de una serie de unidades recurrentes. Cada unidad recurrente mantiene un estado interno o memoria, que se actualiza en función de la entrada actual y el estado anterior. Esta capacidad de mantener una memoria interna permite a las RNN recordar información relevante de eventos pasados y utilizarla para hacer predicciones futuras.

En el contexto del criptotrading, las RNN se utilizan para predecir los precios futuros de las criptomonedas. Se alimenta a la red con una secuencia de datos históricos, como los precios anteriores, el volumen de negociación u otras características relevantes, y se entrena para predecir el siguiente valor en la secuencia. Estas predicciones pueden ayudar a los traders a tomar decisiones informadas sobre cuándo comprar o vender criptomonedas.

Una variante popular de las RNN en el criptotrading es la LSTM (Long Short-Term Memory), que utiliza unidades de memoria especializadas para capturar dependencias a largo plazo en los datos. Las LSTM han demostrado ser efectivas para modelar patrones complejos en series temporales y han sido ampliamente utilizadas en la predicción de precios en el criptotrading.

Otra variante común es el GRU (Gated Recurrent Unit), que es una versión simplificada de las LSTM con menos parámetros. Las GRU también son capaces de capturar dependencias temporales a largo plazo y han sido utilizadas con éxito en la predicción de precios en el criptotrading.

Es importante destacar que el rendimiento de las RNN en la predicción de precios en el criptotrading puede depender de varios factores, como la calidad y la cantidad de los datos disponibles, la elección adecuada de hiperparámetros, la arquitectura de la red y la estrategia de entrenamiento. Además, es recomendable utilizar técnicas de preprocesamiento de datos y validación cruzada para garantizar resultados confiables y evitar el sobreajuste de la red a los datos históricos.

Algunas redes de este tipo que se utilizan en el criptotrading son las siguientes:

1.- LSTM (Long Short-Term Memory): Las LSTM son ampliamente utilizadas en el criptotrading debido a su capacidad para capturar dependencias a largo plazo en las series de tiempo. Las LSTM pueden modelar patrones complejos y cambios en las tendencias de los precios de criptomonedas, lo que las hace adecuadas para la predicción y el análisis de mercado.

2.- GRU (Gated Recurrent Unit): Las GRU también son populares en el criptotrading. Son similares a las LSTM en términos de su capacidad para manejar dependencias a largo plazo, pero tienen una estructura más simple. Las GRU pueden ser más eficientes computacionalmente y ofrecen un rendimiento similar a las LSTM en muchos casos.

3.- RNN Bidireccionales: Las RNN bidireccionales son utilizadas en el criptotrading para capturar la información tanto del pasado como del futuro en un contexto temporal. Esto permite que el modelo tenga una visión más completa de la secuencia de precios y patrones en los datos, lo que puede mejorar la precisión de las predicciones.

4.- RNN Apiladas (Stacked RNNs): Las RNN apiladas son utilizadas para construir modelos más profundos y capturar representaciones de mayor nivel en las series de tiempo de criptotrading. Apilando múltiples capas de unidades recurrentes, se pueden aprender características más complejas y abstracciones de los datos, lo que puede ayudar en la identificación de patrones y tendencias en el mercado de criptomonedas.

###### 1.1.3.1.1.- Redes Neuronales de Memoria a Corto y Largo Plazo (LSTM):

Las Redes Neuronales de Memoria a Corto y Largo Plazo (LSTM, por sus siglas en inglés) son una variante popular de las redes neuronales recurrentes utilizadas en la predicción de precios en el criptotrading. Estas redes son especialmente adecuadas para modelar datos secuenciales, como las series de tiempo de precios de criptomonedas, debido a su capacidad para capturar dependencias temporales a largo plazo.

La principal característica distintiva de las LSTM es su capacidad de mantener y actualizar una "celda de memoria" interna, que les permite capturar dependencias a largo plazo en los datos. Esto es especialmente útil en el criptotrading, donde los precios de las criptomonedas pueden estar influenciados por eventos pasados que ocurrieron hace mucho tiempo.

La arquitectura de una LSTM consta de varias unidades de memoria llamadas "celdas LSTM". Cada celda LSTM tiene tres componentes principales: una "puerta de entrada" que controla qué información nueva se debe ingresar a la celda, una "puerta de olvido" que controla qué información antigua se debe descartar de la celda y una "puerta de salida" que controla qué información se debe transmitir desde la celda a la siguiente capa de la red.

Las LSTM son capaces de aprender y adaptarse a patrones complejos en los datos de series temporales. Pueden modelar relaciones no lineales y capturar dependencias a largo plazo, lo que las hace especialmente útiles para predecir precios en el criptotrading, donde los patrones y las tendencias pueden ser altamente no lineales y a largo plazo.

Para utilizar las LSTM en el criptotrading, se alimenta a la red con una secuencia de datos históricos, como los precios pasados de una criptomoneda, y se entrena para predecir el siguiente valor en la secuencia. El entrenamiento implica ajustar los pesos y los sesgos de las LSTM para minimizar el error entre las predicciones y los valores reales.

Es importante tener en cuenta que el rendimiento de las LSTM en la predicción de precios en el criptotrading puede depender de varios factores, como la calidad y la cantidad de los datos, la elección adecuada de hiperparámetros, la arquitectura de la red y la estrategia de entrenamiento. Además, se recomienda utilizar técnicas de preprocesamiento de datos, como la normalización, y técnicas de validación cruzada para evaluar y ajustar el rendimiento del modelo.

###### 1.1.3.1.2.- GRU (Gated Recurrent Unit):

Las Gated Recurrent Units (GRU) son un tipo de Redes Neuronales Recurrentes (RNN) que se utilizan en el criptotrading para el análisis y la predicción de precios de criptomonedas.

1. Arquitectura de las GRU: Las GRU son una variante de las RNN que incorporan mecanismos de puertas para controlar el flujo de información dentro de la red. Tienen una estructura interna más simple que las redes LSTM (Long Short-Term Memory), lo que las hace computacionalmente más eficientes. Una GRU consta de una puerta de actualización (update gate) y una puerta de reinicio (reset gate). Estas puertas permiten a la red decidir qué información debe ser recordada o olvidada en cada paso de tiempo.

2. Análisis de secuencias de tiempo: Las GRU son especialmente efectivas para el análisis de secuencias de tiempo, como los datos de precios de criptomonedas. Dado que los precios de las criptomonedas exhiben patrones y dependencias temporales, las GRU pueden capturar estas relaciones y aprender a predecir los movimientos futuros del precio.

3. Aprendizaje de dependencias a largo plazo: Una de las ventajas de las GRU es su capacidad para capturar dependencias a largo plazo en las secuencias de tiempo. Esto significa que la red puede recordar información relevante de eventos pasados y utilizarla para realizar predicciones precisas en el futuro. En el criptotrading, donde los precios están influenciados por factores históricos y tendencias de mercado, esta capacidad de las GRU es especialmente valiosa.

4. Entrenamiento y ajuste de las GRU: Al igual que otros modelos de aprendizaje automático, las GRU requieren de un proceso de entrenamiento utilizando datos históricos de precios de criptomonedas. Durante el entrenamiento, las GRU ajustan sus parámetros para minimizar la diferencia entre las predicciones realizadas y los valores reales de los precios. Esto se logra mediante técnicas de optimización como el descenso del gradiente estocástico (SGD) o algoritmos más avanzados como Adam.

1. Predicción de precios: Una vez que las GRU han sido entrenadas y ajustadas, se pueden utilizar para realizar predicciones de precios futuros. Se proporcionan los datos históricos más recientes como entrada a la red, y esta genera una predicción del precio futuro. Esta predicción se basa en los patrones y correlaciones aprendidos durante el entrenamiento.

###### 1.1.3.1.3.- Redes Neuronales Recurrentes Bidireccionales:

Las Redes Neuronales Recurrentes Bidireccionales (Bidirectional RNNs) son una extensión de las Redes Neuronales Recurrentes (RNNs) estándar que se utilizan en el criptotrading. A diferencia de las RNNs unidireccionales, las RNNs bidireccionales procesan la secuencia de datos en dos direcciones: hacia adelante y hacia atrás.

El objetivo principal de utilizar RNNs bidireccionales en el criptotrading es capturar tanto la información pasada como la futura de una secuencia de datos. Esto permite que el modelo tenga una comprensión más completa de la serie temporal y pueda hacer predicciones más precisas.

En el contexto del criptotrading, las RNNs bidireccionales pueden utilizarse para predecir los precios futuros de las criptomonedas. Al procesar la secuencia de precios hacia adelante y hacia atrás, el modelo puede capturar patrones y tendencias en ambos sentidos y utilizar esa información para realizar predicciones más confiables.

La arquitectura de una RNN bidireccional se compone de dos capas de unidades recurrentes: una capa que procesa la secuencia en orden cronológico (hacia adelante) y otra capa que procesa la secuencia en orden inverso (hacia atrás). Cada capa tiene su propio conjunto de parámetros y se entrena de manera independiente. Al finalizar, las salidas de ambas capas se combinan para obtener una representación final de la secuencia.

En resumen, las RNNs bidireccionales son una herramienta poderosa en el criptotrading, ya que permiten capturar tanto la información pasada como futura de una serie temporal. Esto puede mejorar la capacidad del modelo para predecir los precios de las criptomonedas y tomar decisiones más informadas en el trading.

###### 1.1.3.1.4.- Redes Neuronales Recurrentes Apiladas:

Las Redes Neuronales Recurrentes Apiladas (Stacked RNNs) son una técnica utilizada en el criptotrading que implica la superposición de múltiples capas de unidades recurrentes. Esta arquitectura permite que el modelo capture representaciones más complejas y de mayor nivel de abstracción de los datos secuenciales, lo que puede conducir a una mejor capacidad de predicción en el criptotrading.

En una RNN apilada, las salidas de las capas inferiores se utilizan como entradas para las capas superiores, creando así una jerarquía de representaciones en el modelo. Cada capa adicional de la RNN apilada tiene su propia memoria interna y realiza operaciones recurrentes en la secuencia de entrada. Esto permite que el modelo aprenda relaciones de dependencia a largo plazo en los datos y capture patrones más complejos.

En el contexto del criptotrading, las RNNs apiladas se utilizan para analizar y predecir las tendencias y los patrones de precios de las criptomonedas. Al tener múltiples capas de unidades recurrentes, el modelo puede aprender características más sofisticadas y detectar relaciones sutiles en los datos históricos de precios. Esto puede ayudar a identificar señales de compra o venta en el trading de criptomonedas.

Es importante tener en cuenta que el uso de RNNs apiladas en el criptotrading requiere un ajuste cuidadoso de los hiperparámetros y un conjunto de datos suficientemente grande y representativo. Además, el entrenamiento de RNNs apiladas puede ser computacionalmente más costoso que las RNNs estándar debido al mayor número de parámetros y la mayor complejidad del modelo.

En resumen, las RNNs apiladas son una técnica poderosa en el criptotrading que permite capturar relaciones complejas y de largo plazo en los datos secuenciales. Al utilizar múltiples capas de unidades recurrentes, el modelo puede aprender representaciones más sofisticadas y realizar predicciones más precisas en el trading de criptomonedas.

##### 1.1.3.2.- Redes Neuronales Convolucionales (CNN):

Las Redes Neuronales Convolucionales (CNN, por sus siglas en inglés) son un tipo de arquitectura de redes neuronales especialmente diseñada para procesar datos estructurados en forma de matrices, como imágenes. Sin embargo, también se han aplicado con éxito en otros dominios, incluido el análisis de series de tiempo, como los datos de precios en el criptotrading.

A diferencia de las redes neuronales tradicionales, las CNN están compuestas por capas convolucionales y capas de agrupación, que les permiten extraer características relevantes de los datos de entrada de manera eficiente y efectiva.

Las capas convolucionales en una CNN aplican filtros convolucionales a las entradas de datos, lo que permite detectar características locales en la imagen o serie de tiempo. Estos filtros se deslizan sobre los datos de entrada y realizan operaciones de convolución para calcular características como bordes, texturas o patrones específicos. Los filtros convolucionales son aprendidos durante el entrenamiento de la red y se pueden aplicar en varias capas para capturar características de diferentes niveles de abstracción.

Después de las capas convolucionales, se utilizan capas de agrupación para reducir la dimensionalidad de las características extraídas y preservar las características más relevantes. La operación de agrupación, comúnmente conocida como "pooling", reduce el tamaño de las características al seleccionar el valor máximo o promedio en una vecindad determinada.

En el contexto del criptotrading, las CNN pueden aplicarse directamente a los datos de series de tiempo de precios de criptomonedas. En lugar de tratar los datos como imágenes, se pueden considerar como secuencias unidimensionales y aplicar filtros convolucionales a lo largo del eje temporal. Esto permite a la red capturar patrones y características relevantes en la evolución de los precios.

Las CNN también se pueden combinar con otros componentes, como capas recurrentes, para capturar tanto las dependencias temporales a largo plazo como las características locales en los datos de series de tiempo. Esta combinación de características locales y dependencias temporales puede mejorar la capacidad de la red para predecir los precios futuros en el criptotrading.

Es importante tener en cuenta que el diseño y la arquitectura de una CNN pueden variar según el problema y los datos específicos. La elección adecuada de hiperparámetros, como el tamaño de los filtros convolucionales, las capas de agrupación y las funciones de activación, es crucial para obtener un rendimiento óptimo. Además, se requiere un conjunto de datos de entrenamiento adecuado y técnicas de preprocesamiento de datos, como la normalización, para mejorar el rendimiento del modelo y evitar el sobreajuste.

De aquí me parece interesante la idea de poder combinar la CNN con capas recurrentes, sería bueno intentarlo.

Las Redes Neuronales Convolucionales (CNN) son ampliamente utilizadas en el criptotrading para el análisis y predicción de precios de criptomonedas. Algunas opciones de CNN utilizadas en este contexto incluyen:

1.- CNN unidimensional: Se aplica convolución a lo largo de una dimensión (por ejemplo, el tiempo) de los datos secuenciales de precios. Puede capturar patrones y relaciones locales en los datos y se utiliza para predecir cambios de precios a corto plazo.

2.- CNN bidimensional: Se aplica convolución en dos dimensiones (por ejemplo, tiempo y características) a los datos de precios representados en forma de matriz. Puede extraer características espaciales y temporales más complejas y se utiliza para predecir cambios de precios a corto y mediano plazo.

3.- CNN con atención: Combina la arquitectura de CNN con mecanismos de atención para enfocarse en partes relevantes de los datos y asignar pesos a diferentes características. Puede mejorar la capacidad de capturar relaciones importantes en los datos de precios y se utiliza para predecir cambios de precios a corto y mediano plazo.

1. - CNN recurrente: Combina la arquitectura de CNN con unidades recurrentes, como las LSTM o GRU, para capturar tanto las características locales como las dependencias a largo plazo en los datos de precios. Puede predecir cambios de precios a corto, mediano y largo plazo.

###### 1.1.3.2.1.- CNN unidimensional:

Las Redes Neuronales Convolucionales (CNN) unidimensionales se utilizan en el criptotrading para el análisis y la predicción de precios de criptomonedas en función de datos secuenciales, como series temporales de precios. Estas CNN se caracterizan por aplicar la operación de convolución a lo largo de una dimensión, generalmente el tiempo, para extraer características locales y patrones relevantes en los datos.

En el contexto del criptotrading, las CNN unidimensionales se utilizan para capturar patrones y relaciones temporales en los precios de criptomonedas. Algunas características y consideraciones clave son las siguientes:

1. Capas de convolución: Las capas de convolución se utilizan para aplicar filtros a los datos secuenciales de precios. Cada filtro aprende a detectar patrones específicos en los datos, como tendencias, cambios bruscos, o patrones cíclicos. El tamaño y el número de filtros se determinan mediante experimentación y ajuste de hiperparámetros.

2. Capas de agrupación (pooling): Después de aplicar las capas de convolución, es común agregar la información relevante utilizando capas de agrupación, como MaxPooling o AveragePooling. Estas capas reducen la dimensionalidad de las características extraídas, preservando las características más importantes y mejorando la eficiencia computacional.

3. Capas completamente conectadas: Después de las capas de convolución y agrupación, se pueden agregar capas completamente conectadas para realizar la clasificación o predicción final. Estas capas toman como entrada las características extraídas y generan las salidas deseadas, como la predicción de los precios futuros.

4. Funciones de activación: Se aplican funciones de activación no lineales, como ReLU (Rectified Linear Unit), para introducir la no linealidad en la red y permitir la captura de relaciones más complejas en los datos.

5. Regularización: Para evitar el sobreajuste y mejorar la generalización del modelo, se pueden utilizar técnicas de regularización como la regularización L1 o L2, dropout o la normalización por lotes (batch normalization).

6. Ajuste de hiperparámetros: La elección de los hiperparámetros, como el número de capas, el tamaño de los filtros, la tasa de aprendizaje y el número de épocas de entrenamiento, es crucial para obtener un buen rendimiento del modelo. Se recomienda realizar ajustes y experimentar con diferentes configuraciones para optimizar el rendimiento en la tarea de predicción de precios en el criptotrading.

En resumen, las CNN unidimensionales son una poderosa herramienta para el análisis y la predicción de precios en el criptotrading. Permiten capturar patrones temporales en los datos de precios y pueden ser utilizadas para predecir cambios de precios a corto plazo. Sin embargo, es importante tener en cuenta que el desempeño de las CNN unidimensionales puede depender de la calidad de los datos, la cantidad de datos disponibles y otros factores específicos de cada criptomoneda o mercado.

Me parece importante destacar la utilidad de esta arquitectura en la predicción de precios a corto plazo. Lo cual se podría combinar con alguna que vaya bien en el largo plazo.

###### 1.1.3.2.2.- CNN bidimensional:

Las Redes Neuronales Convolucionales (CNN) bidimensionales se utilizan en el criptotrading para el análisis y la predicción de precios de criptomonedas en función de datos de imágenes, como gráficos de velas o imágenes de orden de libros de órdenes. Estas CNN se caracterizan por aplicar la operación de convolución en dos dimensiones (alto y ancho) para extraer características espaciales y patrones relevantes en los datos.

En el contexto del criptotrading, las CNN bidimensionales se utilizan para capturar patrones visuales y relaciones espaciales en los datos de precios representados como imágenes. Algunas características y consideraciones clave son las siguientes:

1. Capas de convolución: Las capas de convolución en una CNN bidimensional aplican filtros espaciales a la imagen de entrada para detectar características locales. Cada filtro aprende a reconocer patrones específicos, como líneas, bordes, formas o combinaciones de estos elementos. Estos filtros se aplican a través de la imagen para obtener mapas de características.

2. Capas de agrupación (pooling): Después de aplicar las capas de convolución, es común agregar la información relevante utilizando capas de agrupación, como MaxPooling o AveragePooling, que reducen la dimensionalidad de los mapas de características. Estas capas conservan las características más importantes y mejoran la eficiencia computacional.

3. Capas completamente conectadas: Después de las capas de convolución y agrupación, se pueden agregar capas completamente conectadas para realizar la clasificación o predicción final. Estas capas toman como entrada las características extraídas y generan las salidas deseadas, como la predicción de los precios futuros.

4. Funciones de activación: Se aplican funciones de activación no lineales, como ReLU (Rectified Linear Unit), para introducir la no linealidad en la red y permitir la captura de relaciones más complejas en los datos.

5. Regularización: Para evitar el sobreajuste y mejorar la generalización del modelo, se pueden utilizar técnicas de regularización como la regularización L1 o L2, dropout o la normalización por lotes (batch normalization).

6. Preprocesamiento de datos: Antes de aplicar las CNN bidimensionales, es importante realizar un adecuado preprocesamiento de los datos de imágenes. Esto puede incluir la normalización de los valores de los píxeles, la redimensión de las imágenes al mismo tamaño y la aplicación de técnicas de aumento de datos para aumentar la cantidad y variedad de ejemplos de entrenamiento.

7. Ajuste de hiperparámetros: Al igual que con otras redes neuronales, los hiperparámetros, como el número y tamaño de los filtros, la tasa de aprendizaje y el número de épocas de entrenamiento, deben ajustarse adecuadamente para obtener un buen rendimiento del modelo en la predicción de precios en el criptotrading.

Las CNN bidimensionales son especialmente útiles cuando los datos de precios se representan como imágenes, ya que permiten capturar patrones visuales y relaciones espaciales en los gráficos de velas u otras representaciones visuales. Sin embargo, es importante tener en cuenta que la calidad de los datos de entrada, la cantidad de datos disponibles y otros factores específicos del criptomercado pueden influir en el rendimiento de

la red. Además, se requiere un conjunto de datos etiquetado adecuado para el entrenamiento y la validación del modelo.

Realmente no me parece tan buena idea analizar el mercado y hacer las predicciones solo desde una perspectiva puramente visual, por lo que quizá esta arquitectura no sea la mejor para nuestros fines.

###### 1.1.3.2.3.- CNN con atención:

Las Redes Neuronales Convolucionales (CNN) con atención en el criptotrading combinan las capacidades de las CNN y las redes neuronales con atención para capturar características relevantes en los datos de precios de criptomonedas y enfocarse en las partes más importantes de la secuencia temporal.

La atención en las CNN se utiliza para asignar pesos o importancias a diferentes partes de la entrada, lo que permite a la red concentrarse en regiones específicas de interés en los datos. Esto es especialmente útil en el criptotrading, donde ciertos patrones o eventos pueden ser más relevantes para predecir los movimientos de precios.

El proceso básico de una CNN con atención en el criptotrading implica los siguientes pasos:

1. Capas de convolución: La red comienza con capas de convolución para extraer características espaciales de los datos de precios. Estas capas aplican filtros convolucionales para detectar patrones locales en la secuencia temporal.

2. Capas de agrupación: Después de las capas de convolución, se pueden utilizar capas de agrupación para reducir la dimensionalidad de las características extraídas y resumir la información relevante.

3. Capas de atención: En lugar de pasar directamente a las capas completamente conectadas, se introducen capas de atención. Estas capas generan un vector de atención que asigna pesos a diferentes partes de la secuencia temporal en función de su relevancia para la tarea de predicción. Los pesos de atención se calculan mediante una función de atención, que puede basarse en la similitud de características o en otros criterios definidos.

4. Combinación de características ponderadas: Los vectores de atención se utilizan para ponderar las características extraídas por las capas de convolución. Esto significa que las partes más importantes de la secuencia temporal tienen un mayor impacto en el resultado final de la red.

5. Capas completamente conectadas: Después de combinar las características ponderadas, se pueden utilizar capas completamente conectadas para realizar la predicción final de los precios en el criptotrading. Estas capas pueden incluir funciones de activación, como ReLU, y técnicas de regularización para mejorar el rendimiento y la generalización del modelo.

Las CNN con atención en el criptotrading permiten a la red enfocarse en los momentos críticos y las características clave en la secuencia temporal de precios. Esto ayuda a capturar patrones relevantes y a mejorar la capacidad del modelo para hacer predicciones precisas.

Es importante tener en cuenta que la implementación y el ajuste de una CNN con atención en el criptotrading requieren un conjunto de datos adecuado y una exploración cuidadosa de los hiperparámetros, como el número de capas de convolución, el tamaño de los filtros, la tasa de aprendizaje y la cantidad de atención a aplicar. Además, el preprocesamiento de los datos y el manejo de la serie temporal son consideraciones importantes para obtener resultados óptimos.

En lo personal, me parece que cualquier arquitectura con características de atención nos puede ser favorable puesto que en realidad sí se suelen repetir patrones con base en sucesos pasados, como el halving. Aunque el halving es mejor analizarlo con otro tipo de herramientas, como el analisis de sentimiento o simplemente la distancia en días al halving y comparar esa métrica con los halvings pasados. Pero como dije, es algo más complejo de procesar y analizar que la predicción de precios.

###### 1.1.3.2.4.- CNN recurrente:

Las Redes Neuronales Convolucionales Recurrentes (CNN recurrentes) en el criptotrading combinan las capacidades de las CNN y las redes neuronales recurrentes (RNN) para capturar tanto las características espaciales como las temporales de los datos de precios de criptomonedas.

A diferencia de las RNN tradicionales, que tienen una estructura de bucle en la que la salida de una unidad se alimenta como entrada a la siguiente unidad en la secuencia, las CNN recurrentes utilizan conexiones recurrentes dentro de las capas convolucionales. Esto permite a la red capturar características temporales en diferentes niveles de abstracción a medida que la información se propaga a través de las capas convolucionales.

El proceso básico de una CNN recurrente en el criptotrading implica los siguientes pasos:

1. Capas convolucionales: Al igual que en las CNN tradicionales, las capas convolucionales se utilizan para extraer características espaciales de los datos de precios. Estas capas aplican filtros convolucionales para detectar patrones locales en la secuencia temporal.

2. Capas recurrentes: Las capas recurrentes se agregan después de las capas convolucionales para capturar las dependencias temporales en los datos. Estas capas utilizan conexiones recurrentes, como las celdas LSTM o GRU, que permiten que la información fluya hacia atrás en el tiempo y se utilice para informar las predicciones futuras.

3. Capas de agrupación: Después de las capas recurrentes, se pueden utilizar capas de agrupación para reducir la dimensionalidad y resumir la información relevante. Esto puede ayudar a capturar características temporales a diferentes escalas y niveles de abstracción.

4. Capas completamente conectadas: Finalmente, se utilizan capas completamente conectadas para realizar la predicción final de los precios en el criptotrading. Estas capas pueden incluir funciones de activación, como ReLU, y técnicas de regularización para mejorar el rendimiento y la generalización del modelo.

Las CNN recurrentes en el criptotrading son especialmente útiles cuando las relaciones temporales son críticas para hacer predicciones precisas. Estas redes pueden capturar patrones temporales complejos en los datos de precios, lo que permite modelar tendencias, ciclos y otros fenómenos recurrentes en el mercado de criptomonedas.

Es importante tener en cuenta que la implementación y el ajuste de una CNN recurrente en el criptotrading requieren una cuidadosa selección de hiperparámetros, como el número de capas convolucionales y recurrentes, el tamaño de los filtros, la tasa de aprendizaje y la longitud de la secuencia de entrada. Además, el preprocesamiento adecuado de los datos, incluido el manejo de la serie temporal y la normalización de los precios, es esencial para obtener resultados óptimos.

##### 1.1.3.3.- Redes Neuronales Generativas Adversariales (GAN):

Las Redes Neuronales Generativas Adversariales (GAN, por sus siglas en inglés) son un tipo de modelo de aprendizaje automático que se utiliza en el criptotrading y otros campos para generar datos sintéticos que se asemejan a los datos de entrada reales. Las GAN se componen de dos componentes principales: el generador y el discriminador.

El generador es una red neuronal que toma una entrada aleatoria, conocida como ruido latente, y genera datos sintéticos que se parecen a los datos reales. En el contexto del criptotrading, esto puede implicar generar secuencias de precios de criptomonedas que se asemejan a las secuencias reales observadas en el mercado. El generador aprende a través del entrenamiento a producir datos sintéticos que engañan al discriminador y se asemejan lo más posible a los datos reales.

El discriminador, por otro lado, es una red neuronal que se entrena para distinguir entre datos reales y datos generados por el generador. Su objetivo es clasificar correctamente los datos de entrada como reales o sintéticos. A medida que el generador mejora su capacidad para generar datos más realistas, el discriminador también se entrena para ser más preciso en su clasificación.

El proceso de entrenamiento de una GAN implica la competencia entre el generador y el discriminador. El generador busca mejorar continuamente su capacidad para generar datos sintéticos que engañen al discriminador, mientras que el discriminador busca mejorar su capacidad para distinguir entre datos reales y sintéticos. Esta competencia entre ambos componentes conduce a la mejora y refinamiento del generador para producir datos sintéticos de alta calidad que se asemejen a los datos reales.

En el contexto del criptotrading, las GAN se pueden utilizar para generar datos sintéticos de precios de criptomonedas que se asemejen a las secuencias reales observadas en el mercado. Estos datos sintéticos pueden ser utilizados para realizar análisis, pruebas y experimentos sin la necesidad de utilizar datos reales o históricos. Además, las GAN también pueden aplicarse en otros aspectos del criptotrading, como la generación de estrategias de trading automatizadas o la detección de anomalías en los precios.

Es importante destacar que las GAN son modelos complejos y requieren un entrenamiento y ajuste adecuados para obtener resultados satisfactorios. Además, se necesita un conjunto de datos de entrenamiento representativo y técnicas de preprocesamiento para garantizar la calidad de los datos generados. Las GAN también pueden ser sensibles a problemas de estabilidad y convergencia, por lo que se requiere experiencia en su implementación y ajuste.

Algunos tipos de GAN utilizados en criptotrading son los siguiente:

En el criptotrading, se han explorado varias variantes de Redes Neuronales Generativas Adversariales (GAN) para diferentes aplicaciones. Algunos tipos comunes de GAN utilizados en el criptotrading incluyen:

1. GANs generativas de imágenes: Estas GAN se utilizan para generar imágenes sintéticas de criptomonedas o gráficos de precios. A través del entrenamiento adversarial, el generador aprende a producir imágenes realistas de criptomonedas, mientras que el discriminador se entrena para distinguir entre las imágenes generadas y las reales.

2. GANs para la generación de secuencias de precios: Estas GAN se utilizan para generar secuencias de precios de criptomonedas sintéticas que sean coherentes con los patrones y características observados en los datos históricos. Esto puede ser útil para la generación de datos de entrenamiento adicionales o para realizar simulaciones y análisis de escenarios.

3. GANs para la detección de anomalías: Estas GAN se utilizan para detectar anomalías o eventos inusuales en los datos de precios de criptomonedas. El generador se entrena para capturar la distribución normal de los datos, mientras que el discriminador se entrena para distinguir entre los datos normales y las anomalías. Esto permite identificar cambios o comportamientos anormales en los precios que pueden ser indicativos de oportunidades de trading o eventos inesperados en el mercado.

4. GANs para la mejora de señales de trading: Estas GAN se utilizan para mejorar las señales de trading mediante la generación de señales sintéticas que complementen o mejoren las señales existentes. Por ejemplo, se puede utilizar una GAN para generar señales de compra o venta adicionales basadas en patrones y características aprendidas de los datos históricos.

Sin embargo, solo vamos a ver el punto 2 y el 4, puesto que los otros 2 puntos no pienso que nos puedan ser realmente útiles en esta área de desarrollo, de hecho, de estos 2, el que pienso que podría funcionar en alguna otra área es el punto 4, ya que no me parece muy asertivo predecir precios mediante imágenes.  
  
Igual se puede hablar sobre esto.

###### 1.1.3.3.1.- GANs para la generación de secuencias de precios:

Las GANs (Redes Neuronales Generativas Adversariales) para la generación de secuencias de precios en el criptotrading se utilizan para generar datos sintéticos de precios de criptomonedas que sean coherentes con los patrones y características observados en los datos históricos. Estas GANs se componen de dos componentes principales: el generador y el discriminador.

El generador es una red neuronal que toma como entrada un vector de ruido aleatorio y genera una secuencia de precios sintéticos. A medida que se entrena, el generador aprende a mapear el espacio de ruido a secuencias de precios que se asemejen a los datos reales. Puede estar compuesto por capas LSTM, GRU o convolucionales, dependiendo de la arquitectura específica.

El discriminador es otra red neuronal que se entrena para distinguir entre las secuencias de precios generadas por el generador y las secuencias de precios reales del conjunto de datos históricos. Su objetivo es aprender a identificar las diferencias entre los datos generados y los datos reales. A medida que el discriminador mejora su capacidad para discernir entre los datos sintéticos y los reales, el generador se ajusta para generar secuencias de precios más realistas.

Durante el entrenamiento, el generador y el discriminador se actualizan de forma adversarial. El generador trata de engañar al discriminador generando secuencias de precios que sean cada vez más difíciles de distinguir de los datos reales, mientras que el discriminador se entrena para mejorar su capacidad de discernimiento. Esta competencia entre el generador y el discriminador impulsa la mejora continua del generador y la generación de secuencias de precios más realistas.

Una vez entrenada, la GAN puede generar secuencias de precios sintéticas que pueden utilizarse para diversos propósitos en el criptotrading, como la generación de datos adicionales para entrenar modelos predictivos, la realización de simulaciones y análisis de escenarios, o la exploración de posibles patrones y tendencias en los precios.

Es importante tener en cuenta que la generación de secuencias de precios con GANs implica una serie de desafíos, como la captura de la volatilidad y los patrones de fluctuación de los precios, así como la garantía de que los datos generados sean realistas y útiles para el análisis y la toma de decisiones en el criptotrading. Por lo tanto, se requiere un diseño cuidadoso de la arquitectura de la GAN, la selección de hiperparámetros y una evaluación rigurosa de los resultados para garantizar su utilidad y validez.

###### 1.1.3.3.2.- GANs para la mejora de señales de trading:

Las GANs (Redes Neuronales Generativas Adversariales) también se pueden utilizar en el criptotrading para la mejora de señales de trading. En este contexto, las GANs pueden emplearse para generar señales de trading más precisas, refinando las señales existentes o generando nuevas señales basadas en patrones aprendidos de los datos históricos.

El proceso comienza con la recopilación de datos históricos de precios de criptomonedas y las señales de trading correspondientes. Estos datos se utilizan para entrenar una GAN, que consta de un generador y un discriminador, de manera similar a la descripción anterior.

El generador de la GAN toma como entrada un vector de ruido y genera señales de trading sintéticas. Estas señales pueden representar puntos de entrada o salida en una operación, niveles de stop-loss o take-profit, u otros indicadores relevantes para el trading de criptomonedas. A medida que la GAN se entrena, el generador aprende a generar señales que se asemejen a las señales reales del conjunto de datos históricos.

El discriminador, por su parte, se entrena para distinguir entre las señales sintéticas generadas por el generador y las señales reales del conjunto de datos históricos. Su objetivo es aprender a identificar las diferencias y las características distintivas de las señales auténticas. Con el tiempo, el generador se ajusta para generar señales más cercanas a las señales reales, mientras que el discriminador se perfecciona en su capacidad para diferenciar entre las señales auténticas y las sintéticas.

Una vez entrenada la GAN, se pueden utilizar las señales mejoradas generadas por el generador para respaldar el proceso de toma de decisiones en el criptotrading. Estas señales pueden proporcionar una mayor precisión y confiabilidad en comparación con las señales originales o pueden ser utilizadas como complemento de las señales existentes.

Es importante tener en cuenta que, al igual que con otras aplicaciones de GANs, la generación de señales de trading mejoradas con GANs implica desafíos y consideraciones adicionales. Esto incluye la elección adecuada de la arquitectura de la GAN, la selección de hiperparámetros óptimos, la evaluación rigurosa de los resultados generados y la validación de la utilidad de las señales mejoradas en condiciones de mercado en tiempo real.

En resumen, las GANs aplicadas a la mejora de señales de trading en el criptotrading pueden ayudar a mejorar la precisión y confiabilidad de las señales utilizadas en la toma de decisiones. Sin embargo, es importante llevar a cabo una implementación cuidadosa y una evaluación continua para garantizar que las señales generadas sean realistas, útiles y apropiadas para su uso en el contexto del criptotrading.

Después de leer la descripción no me parece que esta opción sea una buena forma de proceder puesto que se basa en señales de trading que realmente no tenemos y que habría que conseguir primero, lo cual es un proceso más complejo que el solo conseguir los datos temporales, incluso puede ser costoso.

Aún así, se puede repensar en un futuro si se llega a tener éxito.

##### 1.1.3.4.- Redes Neuronales de Onda Larga (WaveNet):

Las Redes Neuronales de Onda Larga, también conocidas como WaveNet, son un tipo de modelo de redes neuronales profundas que se utilizan en el criptotrading y otros campos para modelar y generar secuencias de datos, como los precios de criptomonedas a lo largo del tiempo. WaveNet se basa en la arquitectura de redes neuronales recurrentes, pero a diferencia de las RNN tradicionales, utiliza una estructura de conexiones en forma de onda larga, lo que le permite capturar patrones temporales a diferentes escalas.

Una característica distintiva de WaveNet es su capacidad para generar datos secuenciales de forma autoregresiva. En lugar de predecir directamente el siguiente valor en la secuencia, WaveNet se entrena para predecir la distribución de probabilidad condicional del siguiente valor dado los valores anteriores. Esto permite que el modelo genere muestras más realistas y variadas, capturando la estructura de la secuencia en mayor detalle.

En el contexto del criptotrading, WaveNet se puede utilizar para modelar y predecir los precios de criptomonedas en función de su histórico. El modelo aprende a capturar patrones y dependencias complejas en los datos, lo que le permite generar predicciones más precisas y realistas. Además, WaveNet también puede ser utilizado para la generación de datos sintéticos, lo que puede ser útil para la realización de pruebas y experimentos en el trading al simular diferentes escenarios.

Una de las ventajas de WaveNet es su capacidad para capturar relaciones temporales a largo plazo en los datos. Esto significa que el modelo puede identificar patrones y tendencias que se extienden a lo largo de períodos de tiempo más largos, lo que es especialmente relevante en el criptotrading, donde la volatilidad y las tendencias a largo plazo pueden influir en las decisiones comerciales.

Sin embargo, es importante tener en cuenta que WaveNet es un modelo complejo y requiere una cantidad significativa de datos de entrenamiento y capacidad computacional para su implementación. Además, el proceso de entrenamiento puede ser lento y requerir ajustes cuidadosos de los hiperparámetros para obtener buenos resultados. En general, WaveNet es una herramienta poderosa en la predicción de precios en el criptotrading, pero su implementación exitosa requiere experiencia en el uso de redes neuronales y el manejo de secuencias de datos.

Algunas de sus variantes son:

En el criptotrading, se pueden utilizar diferentes tipos de Redes Neuronales de Onda Larga (WaveNet) para modelar y predecir los precios de criptomonedas u otros aspectos relacionados con el mercado. A continuación, mencionaré algunos de los tipos de WaveNet utilizados en el criptotrading:

1.- WaveNet convolucional: Esta es la forma más común de WaveNet utilizada en el criptotrading. Se basa en una arquitectura de red neuronal convolucional profunda que utiliza dilataciones exponenciales para capturar patrones a diferentes escalas temporales. Esta arquitectura permite modelar relaciones de largo alcance en los datos de precios y puede utilizarse para predecir futuros movimientos de precios.

2.- WaveNet autoregresivo: Este tipo de WaveNet se basa en la generación autoregresiva de los precios de criptomonedas. Utiliza una arquitectura recurrente que toma como entrada las observaciones pasadas y genera una distribución de probabilidad sobre los posibles valores futuros. La generación autoregresiva permite capturar la dependencia temporal en los datos y es útil para generar secuencias de precios sintéticos o realizar pronósticos a largo plazo.

3.- WaveNet condicional: En el criptotrading, a menudo es útil tener en cuenta información adicional, como indicadores técnicos, noticias o sentimiento del mercado, al modelar y predecir los precios. Las WaveNet condicionales se diseñan para incorporar esta información adicional como una condición adicional en la red neuronal. Esto permite una mayor flexibilidad y precisión al realizar predicciones basadas en múltiples fuentes de información.

4.- WaveNet variacional: Esta variante de WaveNet se basa en técnicas de inferencia y generación de modelos generativos mediante el uso de inferencia variacional. Permite modelar la incertidumbre en las predicciones y proporciona una distribución de probabilidad sobre los posibles valores futuros. Esto es útil para el criptotrading, ya que proporciona una estimación de la confianza en las predicciones y ayuda a tomar decisiones informadas.

Es importante tener en cuenta que la elección de la arquitectura de WaveNet depende de los requisitos específicos del problema y de la naturaleza de los datos de criptomonedas. Cada tipo de WaveNet tiene sus ventajas y desafíos asociados, y es necesario realizar experimentos y ajustes para obtener el mejor rendimiento en el contexto del criptotrading.

###### 1.1.3.4.1.- WaveNet convolucional:

La WaveNet convolucional es una arquitectura de red neuronal profunda utilizada en el criptotrading para modelar y predecir los precios de criptomonedas. Se basa en el concepto de dilatación exponencial para capturar patrones a diferentes escalas temporales.

En la WaveNet convolucional, las capas convolucionales utilizan filtros de convolución con dilataciones exponenciales. Estas dilataciones permiten que las capas convolucionales tengan un campo receptivo creciente a medida que se profundiza en la red neuronal. Esto significa que las capas posteriores pueden capturar relaciones de largo alcance en los datos de precios, lo que resulta especialmente útil en el criptotrading, donde las dependencias temporales pueden extenderse a lo largo de múltiples períodos de tiempo.

En el contexto del criptotrading, la WaveNet convolucional se entrena utilizando datos históricos de precios y puede utilizarse para predecir los movimientos futuros de los precios de las criptomonedas. La arquitectura convolucional permite aprender patrones complejos en los datos, como tendencias, patrones estacionales y cambios bruscos, lo que ayuda a capturar las características importantes para la predicción de precios.

Una de las ventajas de la WaveNet convolucional en el criptotrading es su capacidad para modelar dependencias temporales a diferentes escalas. Esto significa que la red neuronal puede capturar tanto patrones a corto plazo como a largo plazo en los datos de precios, lo que permite realizar predicciones más precisas y capturar cambios significativos en la tendencia.

Además, la WaveNet convolucional puede adaptarse a diferentes frecuencias de datos, lo que es beneficioso en el criptotrading, ya que las criptomonedas pueden tener diferentes frecuencias de actualización (por ejemplo, minutos, horas o días). Esto permite utilizar la arquitectura para predecir precios en diferentes horizontes de tiempo y adaptarse a diferentes estrategias de trading.

Sin embargo, es importante destacar que la WaveNet convolucional también puede requerir grandes cantidades de datos para un entrenamiento efectivo y puede ser computacionalmente intensiva debido a su profundidad y número de parámetros. Se requiere un cuidadoso ajuste de hiperparámetros y experimentación para obtener el mejor rendimiento de la arquitectura en el contexto del criptotrading.

En resumen, la WaveNet convolucional es una poderosa arquitectura de red neuronal utilizada en el criptotrading para modelar y predecir los precios de criptomonedas. Su capacidad para capturar patrones a diferentes escalas temporales y adaptarse a diferentes frecuencias de datos la convierte en una opción atractiva para el análisis y la predicción de precios en el mercado de criptomonedas.

###### 1.1.3.4.2.- WaveNet autoregresivo:

Las WaveNet autoregresivas son una variante de las redes neuronales de onda larga (WaveNet) que se utilizan en el criptotrading para modelar y generar secuencias de precios de criptomonedas de manera autónoma.

A diferencia de la WaveNet convolucional que se utiliza para la predicción de precios, las WaveNet autoregresivas se centran en la generación de nuevas muestras de precios basadas en un conjunto de datos de entrenamiento existente. Estas redes neuronales autoregresivas se diseñan para aprender las dependencias temporales y las estructuras de patrones en los datos de precios, de modo que puedan generar nuevas secuencias de precios que sean coherentes con las características del mercado observadas en los datos de entrenamiento.

El proceso de generación de secuencias de precios con WaveNet autoregresivo es iterativo. En cada paso de tiempo, la red neuronal toma una ventana de tiempo anterior como entrada y genera una nueva muestra de precio. Esta muestra se agrega a la secuencia generada y se utiliza como entrada para generar la siguiente muestra, y así sucesivamente hasta que se complete la secuencia deseada.

Para entrenar una WaveNet autoregresiva en el criptotrading, se utiliza un conjunto de datos históricos de precios como datos de entrada. La red neuronal se entrena para aprender las distribuciones condicionales de las muestras de precios dados los valores anteriores en la secuencia. Esto permite que la red capture las relaciones y patrones complejos presentes en los datos y genere secuencias de precios que sean realistas y coherentes con el comportamiento observado.

La generación de secuencias de precios con WaveNet autoregresivo puede ser útil en el criptotrading para simular escenarios hipotéticos, realizar pruebas de estrategias de trading, generar datos sintéticos para el desarrollo y prueba de algoritmos de trading, o incluso como una fuente de señales adicionales para la toma de decisiones en el trading.

Es importante destacar que, si bien las WaveNet autoregresivas pueden generar secuencias de precios realistas, la predicción de precios en el criptotrading sigue siendo un desafío debido a la naturaleza volátil y no lineal de los mercados de criptomonedas. Por lo tanto, la generación de secuencias de precios con WaveNet autoregresivo se utiliza principalmente con fines exploratorios y de simulación, y se requiere un análisis adicional y la consideración de otros factores para la toma de decisiones de trading.

En resumen, las WaveNet autoregresivas son redes neuronales utilizadas en el criptotrading para generar secuencias de precios de criptomonedas. Estas redes aprenden las dependencias temporales y las estructuras de patrones en los datos de precios existentes y generan nuevas secuencias de precios que son coherentes con las características observadas en los datos de entrenamiento. La generación de secuencias de precios con WaveNet autoregresivo es útil para la simulación, el desarrollo y la prueba de estrategias

###### 1.1.3.4.3.- WaveNet condicional:

Las WaveNet condicionales son una variante de las redes neuronales de onda larga (WaveNet) que se utilizan en el criptotrading para modelar y generar secuencias de precios de criptomonedas teniendo en cuenta información adicional o condicional.

A diferencia de las WaveNet convolucionales o autoregresivas que generan secuencias de precios de forma independiente, las WaveNet condicionales toman en consideración características adicionales o condicionales que pueden influir en la generación de precios. Estas características pueden incluir indicadores técnicos, datos macroeconómicos, noticias relevantes u otra información relevante para el mercado de criptomonedas.

El objetivo de las WaveNet condicionales es capturar la relación entre las características condicionales y la generación de precios. La red neuronal se entrena utilizando un conjunto de datos históricos de precios junto con las características condicionales correspondientes. La red aprende a modelar la dependencia entre las características condicionales y la generación de precios, lo que permite generar secuencias de precios que sean coherentes con las características condicionales dadas.

Al utilizar las WaveNet condicionales en el criptotrading, se puede explorar cómo las diferentes características condicionales afectan la generación de precios y cómo se pueden utilizar como señales adicionales para la toma de decisiones en el trading. Por ejemplo, se puede evaluar cómo los cambios en los indicadores técnicos o en los datos macroeconómicos influyen en la dirección y la volatilidad de los precios generados.

Es importante destacar que las WaveNet condicionales requieren un conjunto de datos históricos completo y confiable, así como características condicionales relevantes y precisas para lograr resultados significativos. Además, es importante realizar un análisis adicional y considerar otros factores antes de tomar decisiones de trading basadas únicamente en las secuencias de precios generadas por las WaveNet condicionales.

En resumen, las WaveNet condicionales son redes neuronales de onda larga utilizadas en el criptotrading para generar secuencias de precios de criptomonedas teniendo en cuenta características adicionales o condicionales. Estas redes aprenden la relación entre las características condicionales y la generación de precios, lo que permite generar secuencias de precios que sean coherentes con las características condicionales dadas. Las WaveNet condicionales pueden utilizarse como herramientas para explorar el impacto de diferentes características en la generación de precios y como señales adicionales en la toma de decisiones de trading.

Esta herramienta me parece adecuada para nuestros fines, puesto que podemos agregar datos adicionales a la secuencia temporal, los cuales pueden ser noticias sobre los criptoactivos en cuestión, lo que mejoraría las predicciones, ya que sabemos que las noticias cambian el precio de los criptoactivos de manera considerable.

La dificultad de esto está en el web scrapping de las noticias relacionadas con cada criptomoneda, sin embargo, si ponemos en función esta arquitectura, el web scrapping nos va a funcionar más adelante en el análisis de sentimiento.

###### 1.1.3.4.4.- WaveNet variacional:

Las WaveNet Variacionales (Variational WaveNet) es una variante de la arquitectura WaveNet que combina las redes neuronales generativas autoregresivas con técnicas de inferencia variacional. Esta combinación permite que el modelo genere muestras de alta calidad y, al mismo tiempo, capture la incertidumbre asociada a cada predicción.

En el contexto del criptotrading, las WaveNet Variacionales pueden utilizarse para modelar y generar secuencias de precios de criptomonedas. Estas redes neuronales son capaces de capturar patrones complejos y dependencias temporales en los datos históricos, lo que les permite generar muestras realistas y diversificadas del comportamiento de los precios.

La clave de las WaveNet Variacionales radica en su capacidad para modelar la distribución de probabilidad condicional de cada valor en la secuencia, dado su contexto previo. En lugar de simplemente generar una única predicción determinista, estas redes generan una distribución de probabilidad completa para cada valor en la secuencia. Esto permite tener una noción de la incertidumbre asociada a cada predicción y brinda la posibilidad de muestrear diferentes posibles trayectorias futuras.

El entrenamiento de una WaveNet Variacional implica maximizar la probabilidad conjunta de los datos observados y la probabilidad de los valores ocultos que generan estos datos. Esto se logra mediante la combinación de técnicas de inferencia variacional y aprendizaje autoregresivo. Durante el entrenamiento, la red aprende a generar muestras coherentes y realistas al tiempo que captura la variabilidad y la incertidumbre en los datos.

Al utilizar las WaveNet Variacionales en el criptotrading, se pueden obtener predicciones no solo en forma de valores deterministas, sino también en forma de distribuciones de probabilidad. Esto permite evaluar el riesgo y la incertidumbre asociados a cada predicción y, por lo tanto, tomar decisiones de trading más informadas.

Es importante tener en cuenta que la implementación y el uso de las WaveNet Variacionales en el criptotrading pueden ser complejos y requerir un conjunto de datos adecuado y un entrenamiento riguroso. Además, es necesario considerar otras señales y factores adicionales al tomar decisiones de trading basadas en las predicciones generadas por las WaveNet Variacionales.

En resumen, las WaveNet Variacionales son una variante de la arquitectura WaveNet que combina redes neuronales generativas autoregresivas con técnicas de inferencia variacional. En el criptotrading, estas redes pueden utilizarse para modelar y generar secuencias de precios de criptomonedas, capturando la incertidumbre y la variabilidad asociada a cada predicción. Al proporcionar distribuciones de probabilidad en lugar de predicciones deterministas, las WaveNet Variacionales permiten una evaluación más completa del riesgo y la incertidumbre en las predicciones, lo que puede ayudar en la toma de decisiones de trading.

También esta herramienta me parece bastante adecuada a nuestros intereses puesto que nos brinda la oportunidad de obtener resultados probabilistas, lo que nos permite tomar decisiones mejor informadas.  
  
Podría ser una buena combinación el uso de esta arquitectura con la arquitectura de wavenet condicional.

##### 1.1.3.5.- Redes Neuronales de Atención (Attention Networks):

Las Redes Neuronales de Atención, también conocidas como Attention Networks, son un tipo de arquitectura de redes neuronales que se utiliza en el criptotrading y otros campos para procesar secuencias de datos y enfocarse en partes relevantes de la información. Estas redes se destacan por su capacidad para asignar pesos o importancias diferentes a diferentes elementos de la secuencia de entrada, lo que les permite enfocarse en los aspectos más relevantes y realizar predicciones más precisas.

En el contexto del criptotrading, las Redes Neuronales de Atención pueden utilizarse para analizar patrones en los datos históricos de precios de criptomonedas y realizar predicciones sobre su comportamiento futuro. Estas redes son especialmente útiles cuando hay relaciones y dependencias complejas entre los datos, ya que pueden enfocarse en los momentos y características más relevantes en la secuencia temporal.

La idea principal detrás de las Redes Neuronales de Atención es que no todos los elementos de una secuencia son igualmente importantes para hacer una predicción. Al asignar pesos diferentes a los elementos, el modelo puede prestar más atención a las partes más informativas y hacer un mejor uso de la información disponible.

La arquitectura de las Redes Neuronales de Atención generalmente consta de tres componentes principales: la capa de consulta (query), la capa de clave (key) y la capa de valor (value). Estas capas se utilizan para calcular los pesos de atención para cada elemento de la secuencia, y luego se combinan ponderadamente para obtener una representación final que se utiliza para realizar la predicción.

Una ventaja de las Redes Neuronales de Atención es que son capaces de modelar relaciones a largo plazo en los datos, lo que es especialmente útil en el criptotrading, donde las tendencias y patrones pueden extenderse a lo largo de períodos de tiempo más largos. Además, estas redes pueden capturar relaciones complejas y no lineales en los datos, lo que las hace adecuadas para tratar con la naturaleza volátil y no lineal de los precios de criptomonedas.

Sin embargo, es importante tener en cuenta que la implementación de las Redes Neuronales de Atención puede requerir una cantidad significativa de datos de entrenamiento y poder computacional. Además, es necesario ajustar cuidadosamente los hiperparámetros y estructuras de la red para obtener buenos resultados. En general, las Redes Neuronales de Atención son una herramienta poderosa en el análisis y predicción de precios en el criptotrading, ya que permiten capturar relaciones y patrones relevantes en los datos históricos.

Algunos tipos de esta arquitectura utilizados en el criptotrading son:

En el contexto del criptotrading, las Redes Neuronales de Atención (Attention Networks) se utilizan para capturar las dependencias y relaciones importantes dentro de secuencias de datos, como los precios históricos de criptomonedas. A continuación, se mencionan algunos tipos de Redes Neuronales de Atención que se utilizan en el criptotrading:

1. Attention-Based RNN: Esta arquitectura combina una Red Neuronal Recurrente (RNN) con mecanismos de atención para enfocarse en partes específicas de una secuencia de precios. La atención permite que el modelo se centre en los momentos más relevantes o influyentes dentro de la secuencia.

2. Transformer: El modelo Transformer es una arquitectura de Red Neuronal de Atención ampliamente utilizada en el procesamiento del lenguaje natural (NLP) pero también aplicable al criptotrading. Esta arquitectura se basa en bloques de atención para capturar las relaciones entre los diferentes elementos de una secuencia.

3. Self-Attention: Es un tipo de Red Neuronal de Atención en la que una secuencia de entrada se compara consigo misma para capturar las dependencias entre diferentes elementos. Esto permite que el modelo identifique automáticamente los momentos más influyentes o significativos dentro de la secuencia de precios.

4. Multi-Head Attention: Esta variante de las Redes Neuronales de Atención utiliza múltiples cabezas o subconjuntos de atención para capturar diferentes tipos de relaciones en una secuencia. Cada cabeza se enfoca en aspectos diferentes de la secuencia, lo que permite una representación más rica y compleja de los datos.

5. Bidirectional Attention: Esta técnica combina la atención bidireccional con las redes neuronales recurrentes para capturar tanto las dependencias hacia el pasado como hacia el futuro en una secuencia. Esto permite que el modelo tome en cuenta la información contextual de ambos lados de cada punto en la secuencia de precios.

Estos son solo algunos ejemplos de las Redes Neuronales de Atención utilizadas en el criptotrading. Cada una de estas técnicas tiene sus propias ventajas y características, y su elección dependerá de la naturaleza de los datos y las necesidades específicas del problema de predicción de precios de criptomonedas.

Ahora vamos a dar una descripción breve de cada arquitectura:

###### 1.1.3.5.1.- Attention-Based RNN:

En el criptotrading, el Attention-Based RNN (Red Neuronal Recurrente con Atención) es una arquitectura que combina los beneficios de las Redes Neuronales Recurrentes (RNN) con el mecanismo de atención. Esta combinación permite al modelo enfocarse en partes específicas y relevantes de una secuencia de precios de criptomonedas, lo que ayuda a capturar patrones y relaciones importantes en los datos.

La atención en el Attention-Based RNN se utiliza para asignar ponderaciones a diferentes pasos de tiempo dentro de la secuencia de precios. Estas ponderaciones reflejan la importancia relativa de cada paso de tiempo en relación con la predicción que se va a realizar. Al asignar mayor atención a los momentos más relevantes, el modelo puede tomar decisiones más informadas y precisas sobre los precios futuros de las criptomonedas.

El proceso de atención en el Attention-Based RNN se realiza en varias etapas:

1. Cálculo de los pesos de atención: Se calcula un peso de atención para cada paso de tiempo en la secuencia de precios. Los pesos de atención se calculan mediante una función de similitud que compara el estado oculto de la RNN en cada paso de tiempo con un vector de contexto.

2. Aplicación de los pesos de atención: Los pesos de atención se aplican a los estados ocultos de la RNN, ponderando la importancia de cada estado en la predicción final. Esto permite que el modelo se enfoque en los pasos de tiempo más relevantes y descarte aquellos menos informativos.

3. Combinación de los estados ponderados: Los estados ocultos de la RNN se combinan mediante una suma ponderada para obtener una representación final que captura las características más importantes de la secuencia de precios.

4. Generación de la predicción: Utilizando la representación final combinada, se realiza la predicción de precios futuros de las criptomonedas.

La utilización del Attention-Based RNN en el criptotrading proporciona varias ventajas. Permite al modelo aprender de forma más eficiente patrones complejos y dependencias a largo plazo en los datos de precios de criptomonedas. Además, al enfocarse en partes relevantes de la secuencia, el modelo puede adaptarse a cambios y eventos importantes que pueden influir en los precios.

Sin embargo, es importante tener en cuenta que la elección de la arquitectura y los parámetros específicos del Attention-Based RNN dependerá del problema de predicción y de las características de los datos de criptomonedas. Es recomendable realizar experimentos y ajustes para obtener el mejor rendimiento posible en cada caso.

###### 1.1.3.5.2.- Redes Neuronales Transformers:

Las Redes Neuronales Transformers son un tipo de arquitectura de redes neuronales que se ha vuelto muy popular en el campo del procesamiento del lenguaje natural (NLP) debido a su capacidad para capturar relaciones a largo plazo en secuencias de datos. Aunque su aplicación en el criptotrading es menos común, todavía pueden proporcionar beneficios significativos.

A diferencia de las Redes Neuronales Recurrentes (RNN), que procesan secuencias de manera secuencial, las Redes Neuronales Transformers pueden procesar simultáneamente todos los elementos de una secuencia utilizando mecanismos de atención. Esta capacidad de atención permite que las redes Transformers se enfoquen en relaciones importantes y capturen dependencias a largo plazo entre los elementos de la secuencia.

En el contexto del criptotrading, las Redes Neuronales Transformers pueden utilizarse para realizar predicciones de precios o tomar decisiones comerciales. Pueden capturar patrones complejos en los datos de precios históricos y aprovechar esa información para realizar pronósticos futuros.

Una de las principales ventajas de las Redes Neuronales Transformers es su capacidad para modelar relaciones no lineales en los datos, lo que les permite capturar patrones complejos y no lineales en las series de tiempo de los precios de criptomonedas. Además, su capacidad para procesar simultáneamente toda la secuencia de precios históricos puede ayudar a capturar correlaciones a largo plazo entre diferentes monedas o activos criptográficos.

En la práctica, las Redes Neuronales Transformers para el criptotrading se pueden entrenar utilizando datos históricos de precios de criptomonedas y otros datos relevantes, como volúmenes de operaciones, indicadores técnicos u otras características relevantes. Estos modelos pueden ser utilizados para realizar predicciones de precios futuros, identificar oportunidades comerciales o generar señales de compra/venta.

Es importante destacar que, al igual que con cualquier modelo de predicción en el criptotrading, las Redes Neuronales Transformers no son infalibles y están sujetas a las limitaciones y riesgos asociados con la naturaleza volátil del mercado de criptomonedas. La calidad y relevancia de los datos utilizados para entrenar el modelo, así como la adecuada configuración y optimización del mismo, son factores cruciales para obtener resultados precisos y útiles.

En resumen, las Redes Neuronales Transformers son una arquitectura de redes neuronales que pueden capturar relaciones a largo plazo en secuencias de datos. En el criptotrading, estas redes pueden utilizarse para realizar predicciones de precios, identificar oportunidades comerciales y generar señales de compra/venta. Su capacidad para capturar patrones complejos y no lineales en los datos de precios históricos las hace relevantes para el análisis y la toma de decisiones en el contexto de las criptomonedas.

###### 1.1.3.5.3.- Self-Attention:

En el criptotrading, el Attention-Based RNN (Red Neuronal Recurrente con Atención) es una arquitectura que combina los beneficios de las Redes Neuronales Recurrentes (RNN) con el mecanismo de atención. Esta combinación permite al modelo enfocarse en partes específicas y relevantes de una secuencia de precios de criptomonedas, lo que ayuda a capturar patrones y relaciones importantes en los datos.

La atención en el Attention-Based RNN se utiliza para asignar ponderaciones a diferentes pasos de tiempo dentro de la secuencia de precios. Estas ponderaciones reflejan la importancia relativa de cada paso de tiempo en relación con la predicción que se va a realizar. Al asignar mayor atención a los momentos más relevantes, el modelo puede tomar decisiones más informadas y precisas sobre los precios futuros de las criptomonedas.

El proceso de atención en el Attention-Based RNN se realiza en varias etapas:

1. Cálculo de los pesos de atención: Se calcula un peso de atención para cada paso de tiempo en la secuencia de precios. Los pesos de atención se calculan mediante una función de similitud que compara el estado oculto de la RNN en cada paso de tiempo con un vector de contexto.

2. Aplicación de los pesos de atención: Los pesos de atención se aplican a los estados ocultos de la RNN, ponderando la importancia de cada estado en la predicción final. Esto permite que el modelo se enfoque en los pasos de tiempo más relevantes y descarte aquellos menos informativos.

3. Combinación de los estados ponderados: Los estados ocultos de la RNN se combinan mediante una suma ponderada para obtener una representación final que captura las características más importantes de la secuencia de precios.

4. Generación de la predicción: Utilizando la representación final combinada, se realiza la predicción de precios futuros de las criptomonedas.

La utilización del Attention-Based RNN en el criptotrading proporciona varias ventajas. Permite al modelo aprender de forma más eficiente patrones complejos y dependencias a largo plazo en los datos de precios de criptomonedas. Además, al enfocarse en partes relevantes de la secuencia, el modelo puede adaptarse a cambios y eventos importantes que pueden influir en los precios.

Sin embargo, es importante tener en cuenta que la elección de la arquitectura y los parámetros específicos del Attention-Based RNN dependerá del problema de predicción y de las características de los datos de criptomonedas. Es recomendable realizar experimentos y ajustes para obtener el mejor rendimiento posible en cada caso.

###### 1.1.3.5.4.- Multi-Head Attention:

En el criptotrading, las Multi-Head Attention (Atención Multi-Cabeza) son una variante de las redes neuronales de atención que se utilizan para capturar múltiples relaciones entre elementos en una secuencia de precios de criptomonedas. Esta técnica mejora la capacidad de las Self-Attention Networks al permitir que la red focalice la atención en diferentes aspectos y perspectivas de los datos.

La idea principal de las Multi-Head Attention es realizar múltiples transformaciones de atención en paralelo y luego combinar los resultados de estas transformaciones. Cada transformación de atención, conocida como "cabeza de atención", se enfoca en aprender y capturar diferentes patrones o características de la secuencia.

El proceso de las Multi-Head Attention se puede resumir en los siguientes pasos:

1. Representación de entrada: Al igual que en las Self-Attention Networks, la secuencia de precios de criptomonedas se transforma en una representación de entrada, como matrices de embeddings.

2. Cálculo de la atención por cabeza: Para cada cabeza de atención, se calculan los pesos de atención entre los pares de elementos de la secuencia. Cada cabeza de atención tiene sus propios parámetros y se enfoca en aprender diferentes relaciones y características.

3. Aplicación de los pesos de atención por cabeza: Los pesos de atención obtenidos en cada cabeza se utilizan para ponderar las representaciones de los elementos de la secuencia correspondientes a esa cabeza.

4. Concatenación y transformación lineal: Las representaciones ponderadas de cada cabeza de atención se concatenan y se transforman linealmente para obtener una representación final combinada.

5. Generación de la predicción: Utilizando la representación final combinada, se realiza la predicción de los precios futuros de las criptomonedas.

Las Multi-Head Attention son especialmente útiles en el criptotrading, ya que permiten capturar diferentes aspectos y relaciones en los datos de precios, como tendencias a corto y largo plazo, cambios repentinos, patrones estacionales, entre otros. Al tener múltiples cabezas de atención, la red puede aprender representaciones más ricas y capturar de manera más efectiva las complejidades del mercado criptográfico.

Es importante destacar que la implementación y el ajuste de las Multi-Head Attention pueden variar según el problema y los datos específicos del criptotrading. Es recomendable experimentar con diferentes configuraciones de cabezas de atención y parámetros para obtener el mejor rendimiento en cada caso.

###### 1.1.3.5.5.- Bidirectional Attention:

Las Bidirectional Attention Networks (Redes de Atención Bidireccionales) son una variante de las redes neuronales de atención que se utilizan en el criptotrading para capturar tanto la información pasada como futura de una secuencia de precios de criptomonedas y enfocarse en las partes más relevantes de la secuencia.

La idea principal de las Bidirectional Attention Networks es combinar dos componentes clave: la atención bidireccional y la atención ponderada. Estos componentes permiten que la red pueda tomar en consideración tanto la información anterior como la posterior a cada punto en la secuencia de precios.

El proceso de las Bidirectional Attention Networks se puede resumir en los siguientes pasos:

1. Representación de entrada: La secuencia de precios de criptomonedas se transforma en una representación de entrada, como matrices de embeddings o características numéricas.

2. Cálculo de la atención bidireccional: Se aplican dos redes de atención unidireccionales a la secuencia de precios en direcciones opuestas: una en sentido directo (forward) y otra en sentido inverso (backward). Cada red de atención calcula los pesos de atención entre los elementos de la secuencia y captura la información contextual tanto anterior como posterior.

3. Cálculo de la atención ponderada: Se calculan los pesos de atención ponderados para cada elemento de la secuencia utilizando la información obtenida de las redes de atención bidireccionales. Estos pesos reflejan la importancia relativa de cada elemento en el contexto de la secuencia completa.

4. Generación de la representación final: Utilizando los pesos de atención ponderados, se genera una representación final de la secuencia de precios que captura las características más relevantes y distintivas.

5. Aplicación de técnicas de predicción: La representación final se utiliza como entrada para técnicas de predicción, como redes densas o recurrentes, para predecir los precios futuros de las criptomonedas.

Las Bidirectional Attention Networks son particularmente efectivas en el criptotrading porque permiten que el modelo capture tanto la información pasada como futura de los precios, lo que ayuda a identificar patrones y tendencias importantes en el mercado. Además, la atención ponderada permite enfocarse en los puntos más relevantes de la secuencia, mejorando así la capacidad de predicción.

Cabe destacar que la implementación y el ajuste de las Bidirectional Attention Networks pueden variar dependiendo del problema y los datos específicos del criptotrading. Es recomendable experimentar con diferentes arquitecturas y parámetros para optimizar el rendimiento en cada caso.

Cabe aclarar que no se hacen predicciones directas con este modelo, sino que es necesario utilizar sus resultados en modelos que hagan predicciones directas.

##### 1.1.3.6.- Redes Neuronales Generativas Autoregresivas (Auto regressive Generative Networks):

Las Redes Neuronales Generativas Autoregresivas son un tipo de modelo generativo que se centra en la generación de datos secuenciales o multidimensionales. A diferencia de las GAN, las Redes Neuronales Generativas Autoregresivas no involucran una competencia entre un generador y un discriminador.

En lugar de eso, estas redes se basan en la idea de modelar la distribución de probabilidad condicional de los datos, dado el contexto o las observaciones anteriores. Es decir, dado un conjunto de datos previos, la red autoregresiva genera la siguiente muestra condicionada a los datos anteriores.

El proceso de generación comienza con una entrada inicial (como un vector de ruido aleatorio) y luego se utiliza un modelo autoregresivo para generar cada elemento de la secuencia uno a la vez, teniendo en cuenta las observaciones anteriores generadas. Esto implica que la generación es secuencial y depende de las predicciones anteriores.

Las Redes Neuronales Recurrentes (RNN) son un tipo común de arquitectura utilizada para implementar las Redes Neuronales Generativas Autoregresivas. Las RNN tienen la capacidad de capturar dependencias temporales en los datos y se pueden utilizar para modelar secuencias y generar nuevos ejemplos secuenciales.

Un ejemplo popular de Red Neuronal Generativa Autoregresiva es la Red Neuronal de Onda Larga (WaveNet). WaveNet es un modelo de generación de audio desarrollado por DeepMind que utiliza una arquitectura autoregresiva basada en RNN para generar formas de onda de audio de alta calidad. WaveNet ha demostrado ser eficaz en la generación de audio realista y se ha utilizado en diversas aplicaciones, como la síntesis de voz y la música generativa.

En resumen, las Redes Neuronales Generativas Autoregresivas se enfocan en la generación de datos secuenciales o multidimensionales mediante la modelización de la distribución condicional basada en observaciones anteriores. Utilizan una arquitectura autoregresiva, como las RNN, para generar nuevas muestras una a la vez, teniendo en cuenta las predicciones anteriores.

La variante que encontré de esta arquitectura es:

1.- Variational Autoencoders (VAE): Los VAE son una clase de modelos generativos que utilizan redes neuronales autoregresivas para generar datos, en este caso, secuencias de precios de criptomonedas. Los VAE aprenden una representación latente de los datos y utilizan técnicas de muestreo para generar nuevas muestras que sigan la distribución de los datos originales.

Veamos más a detalle este tipo de red neuronal:

###### 1.1.3.6.1.- Variational Autoencoders:

Los Variational Autoencoders (VAE) son una clase de modelos generativos que utilizan redes neuronales para aprender una representación latente de los datos y generar nuevas muestras similares a los datos de entrenamiento. En el contexto del criptotrading, los VAE se utilizan para generar secuencias de precios de criptomonedas basadas en datos históricos.

La estructura básica de un VAE consta de dos componentes principales: el codificador y el decodificador. El codificador es una red neuronal que mapea los datos de entrada (en este caso, los precios de criptomonedas) a una distribución en el espacio latente, normalmente una distribución gaussiana multivariada. El objetivo del codificador es capturar las características relevantes de los datos y comprimirlas en una representación más compacta.

El decodificador, por otro lado, es una red neuronal que mapea las muestras del espacio latente de vuelta al espacio original de los datos. En el caso de los precios de criptomonedas, el decodificador genera nuevas secuencias de precios basadas en las muestras del espacio latente. Estas secuencias generadas pueden considerarse predicciones de los futuros precios de criptomonedas.

Un aspecto clave de los VAE es que están entrenados utilizando el principio de máxima verosimilitud, pero también incorporan una regularización que promueve una distribución latente suave y continua. Esta regularización se realiza a través de la divergencia de Kullback-Leibler (KL), que mide la diferencia entre la distribución latente aprendida y una distribución prior predefinida, como la gaussiana estándar.

En el criptotrading, los VAE se utilizan para generar muestras realistas de precios de criptomonedas y explorar diferentes escenarios futuros. Por ejemplo, se pueden utilizar para generar muestras de precios para realizar análisis de riesgo, probar estrategias de inversión o simular escenarios de trading. Además, los VAE también se pueden combinar con otros modelos o técnicas de predicción para mejorar la precisión y robustez de las predicciones en el criptotrading.

En resumen, los Variational Autoencoders son modelos generativos que utilizan redes neuronales para aprender una representación latente de los datos y generar nuevas muestras similares a los datos de entrenamiento. En el criptotrading, los VAE se utilizan para generar secuencias de precios de criptomonedas y explorar diferentes escenarios futuros.

Ahora hemos terminado de momento con esta sección de predicción de precios, recordando que aún hace falta desarrollar la subsección de Aprendizaje Reforzado aplicado a la Predicción de Precios.

## 2.- ML en la Gestión de Riesgos para Criptotrading:

Antes de hablar específicamente del ML en la Gestión de Riesgos, hablemos más a detalle del concepto en sí mismo.

### 2.1.- Descripción del Concepto “Gestión de Riesgos”:

La gestión de riesgos en el criptotrading es una parte fundamental para proteger y maximizar los activos en el mercado de criptomonedas. Implica la identificación, evaluación y mitigación de los riesgos asociados con las operaciones y las inversiones en criptomonedas. Aquí tienes una descripción más detallada de los aspectos clave de la gestión de riesgos en el criptotrading:

1. Identificación de riesgos: El primer paso en la gestión de riesgos es identificar los posibles riesgos que pueden afectar tus operaciones de criptotrading. Estos riesgos pueden incluir la volatilidad del mercado, la liquidez, la seguridad de las plataformas de intercambio, el riesgo regulatorio, el riesgo operacional, entre otros.

2. Evaluación de riesgos: Una vez identificados los riesgos, es importante evaluar su impacto potencial en tus operaciones y determinar la probabilidad de que ocurran. Esto implica analizar los datos históricos, las tendencias del mercado, los eventos macroeconómicos y otros factores relevantes para evaluar el riesgo asociado a tus inversiones en criptomonedas.

3. Establecimiento de límites de riesgo: Con base en la evaluación de riesgos, es necesario establecer límites de riesgo adecuados para tus operaciones. Estos límites pueden incluir límites de pérdida máxima por operación, límites de exposición al mercado y límites de apalancamiento. Establecer límites de riesgo te ayuda a controlar y limitar las pérdidas en caso de movimientos desfavorables del mercado.

4. Diversificación de la cartera: La diversificación es una estrategia clave para gestionar el riesgo en el criptotrading. Implica invertir en diferentes criptomonedas y activos digitales para reducir la exposición a un solo activo o mercado. Al diversificar tu cartera, puedes mitigar el impacto negativo de eventos específicos en una criptomoneda o sector del mercado.

5. Uso de órdenes stop-loss: Las órdenes stop-loss son herramientas importantes en la gestión de riesgos. Te permiten establecer un nivel de precio específico en el que se ejecutará una venta automáticamente si el precio de mercado alcanza ese nivel. Esto te ayuda a limitar las pérdidas en caso de movimientos adversos del mercado.

6. Monitoreo continuo y ajuste: La gestión de riesgos en el criptotrading es un proceso continuo. Es importante monitorear constantemente tus operaciones, el mercado y los eventos relevantes para realizar ajustes y tomar decisiones informadas. Esto incluye el seguimiento de las noticias y actualizaciones del mercado, el análisis técnico y fundamental, y el ajuste de tus estrategias de inversión en función de las condiciones cambiantes del mercado.

7. Uso de herramientas y análisis: En el criptotrading, existen diversas herramientas y técnicas de análisis que puedes utilizar para gestionar el riesgo. Estas incluyen el análisis técnico, el análisis fundamental, el uso de indicadores técnicos, el análisis de sentimiento del mercado y el uso de modelos predictivos para evaluar las tendencias y los riesgos potenciales.

En resumen, la gestión de riesgos en el criptotrading implica la identificación, evaluación y mitigación de los riesgos asociados con las operaciones en criptomonedas. Al establecer límites de riesgo, diversificar la cartera, utilizar órdenes stop-loss y monitorear constantemente el mercado, puedes reducir los riesgos y proteger tus activos en el emocionante pero volátil mercado de criptomonedas.

Veamos algunos de los puntos que se mencionaron anteriormente sobre la gestión de riesgos.

### 2.2.- Identificación de Riesgos:

La identificación de riesgos en el criptotrading es un proceso clave para gestionar de manera efectiva las operaciones y las inversiones en criptomonedas. Aquí tienes una descripción más detallada de cómo se lleva a cabo la identificación de riesgos en el criptotrading:

1. Volatilidad del mercado: El mercado de criptomonedas es conocido por su alta volatilidad, lo que significa que los precios pueden fluctuar rápidamente en cortos períodos de tiempo. La volatilidad puede ser tanto una oportunidad como un riesgo, ya que las ganancias pueden ser significativas, pero también existe el riesgo de pérdidas sustanciales.

2. Riesgo de liquidez: La liquidez se refiere a la facilidad con la que se pueden comprar o vender criptomonedas en el mercado. Los activos con baja liquidez pueden presentar dificultades para ejecutar operaciones rápidamente o a precios desfavorables, lo que aumenta el riesgo de operar en esos activos.

3. Riesgo de seguridad: La seguridad es un riesgo importante en el criptotrading, ya que los activos digitales pueden ser vulnerables a hackeos y robos. La elección de una plataforma de intercambio segura y el uso de prácticas de seguridad, como el almacenamiento en carteras frías (cold wallets), son fundamentales para mitigar este riesgo.

4. Riesgo regulatorio: El criptotrading está sujeto a regulaciones en diferentes jurisdicciones. Los cambios en las regulaciones pueden tener un impacto significativo en el valor y la operabilidad de las criptomonedas. Es importante tener en cuenta las regulaciones actuales y futuras para evaluar el riesgo regulatorio asociado con el criptotrading.

5. Riesgo operativo: El riesgo operativo se refiere a los riesgos relacionados con la infraestructura tecnológica y los procesos operativos utilizados en el criptotrading. Esto incluye problemas técnicos, interrupciones en los servicios de la plataforma de intercambio, fallas en la ejecución de órdenes, entre otros. Identificar y mitigar estos riesgos es fundamental para garantizar una operación fluida y segura.

6. Riesgo de mercado: El riesgo de mercado se relaciona con los movimientos generales del mercado de criptomonedas. Esto incluye factores macroeconómicos, eventos geopolíticos, cambios en la demanda y oferta, y otros factores que pueden afectar el valor de las criptomonedas en general. Es importante estar al tanto de estos factores y evaluar su impacto potencial en las operaciones.

7. Riesgo de contraparte: El riesgo de contraparte se refiere a la posibilidad de que la contraparte en una transacción no cumpla con sus obligaciones. Esto puede ocurrir en transacciones de intercambio directo entre individuos o en transacciones a través de plataformas de intercambio centralizadas. Es fundamental evaluar la reputación y la confiabilidad de las contrapartes antes de realizar transacciones.

Estos son algunos de los riesgos más comunes en el criptotrading, pero la lista no es exhaustiva. Cada trader debe realizar su propia evaluación de riesgos y adaptar sus estrategias y prácticas de gestión de riesgos según sus necesidades y objetivos

### 2.3.- Evaluación de Riesgos:

La evaluación de riesgos en el criptotrading es un proceso fundamental para identificar, analizar y cuantificar los riesgos asociados con las operaciones y las inversiones en criptomonedas. Aquí tienes una descripción más detallada de cómo se lleva a cabo la evaluación de riesgos en el criptotrading:

1. Identificación de riesgos: El primer paso en la evaluación de riesgos es identificar los posibles riesgos a los que se enfrenta un trader en el criptotrading. Esto implica analizar factores como la volatilidad del mercado, el riesgo de liquidez, el riesgo de seguridad, el riesgo regulatorio, el riesgo operativo y otros riesgos relevantes.

2. Análisis de riesgos: Una vez que los riesgos han sido identificados, se realiza un análisis para comprender su naturaleza, impacto y probabilidad de ocurrencia. Se pueden utilizar técnicas como el análisis histórico de datos, el análisis de tendencias y la investigación de eventos pasados para evaluar los riesgos de manera más precisa.

3. Cuantificación de riesgos: En esta etapa, se asigna una medida numérica a los riesgos identificados y analizados. Esto implica calcular la probabilidad de ocurrencia de cada riesgo y estimar el impacto financiero o comercial asociado. Se pueden utilizar métodos estadísticos y modelos matemáticos para cuantificar los riesgos de manera más precisa.

4. Priorización de riesgos: Una vez que los riesgos han sido identificados, analizados y cuantificados, se deben priorizar en función de su importancia y nivel de impacto. Esto permite al trader enfocar sus esfuerzos de gestión de riesgos en los riesgos más significativos y tomar medidas para mitigarlos de manera efectiva.

5. Desarrollo de estrategias de gestión de riesgos: Con base en la evaluación de riesgos, se desarrollan estrategias y medidas de gestión de riesgos para mitigar los riesgos identificados. Estas estrategias pueden incluir diversificación de cartera, uso de órdenes de stop-loss, establecimiento de límites de exposición, implementación de medidas de seguridad robustas y otras acciones para reducir la exposición a los riesgos.

6. Monitoreo y revisión continua: La evaluación de riesgos es un proceso continuo y dinámico. Es importante monitorear y revisar regularmente los riesgos identificados, así como las estrategias de gestión de riesgos implementadas. Esto permite ajustar las estrategias en función de cambios en el entorno del criptotrading y garantizar que la gestión de riesgos esté actualizada y efectiva.

La evaluación de riesgos en el criptotrading es esencial para proteger los activos y maximizar las oportunidades de ganancias. Cada trader debe realizar su propia evaluación de riesgos en función de sus circunstancias particulares y adaptar las estrategias de gestión de riesgos según sus necesidades y objetivos.

### 2.4.- Establecimiento de Límites de Riesgo:

Establecer límites de riesgo es una parte crucial de la gestión de riesgos en el criptotrading. Los límites de riesgo son niveles predefinidos de exposición al riesgo que un trader está dispuesto a asumir en sus operaciones con criptomonedas. Aquí tienes más detalles sobre el establecimiento de límites de riesgo en el criptotrading:

1. Determinación de la tolerancia al riesgo: El primer paso para establecer límites de riesgo es determinar la tolerancia al riesgo del trader. Esto implica evaluar su aversión al riesgo, su capacidad financiera y sus objetivos de inversión. Al comprender su tolerancia al riesgo, el trader puede establecer límites de riesgo que estén en línea con sus preferencias y metas.

2. Establecimiento de límites de pérdida: Los límites de pérdida son niveles predefinidos de pérdida máxima que el trader está dispuesto a tolerar en una operación o en un período de tiempo determinado. Estos límites se establecen en función del capital disponible, la volatilidad del mercado y la estrategia de trading utilizada. Los límites de pérdida ayudan a limitar las pérdidas y proteger el capital de operación.

3. Establecimiento de límites de exposición: Los límites de exposición se refieren a la cantidad máxima de capital que el trader está dispuesto a invertir en una sola criptomoneda o en un conjunto de criptomonedas. Estos límites se establecen para evitar una sobreexposición a un activo particular o a un sector específico del mercado. Establecer límites de exposición ayuda a diversificar el riesgo y reducir la dependencia de una sola criptomoneda.

4. Definición de límites de apalancamiento: Si el trader utiliza apalancamiento en sus operaciones, es importante establecer límites de apalancamiento. El apalancamiento amplifica tanto las ganancias como las pérdidas, por lo que establecer límites de apalancamiento ayuda a controlar el riesgo y evitar pérdidas excesivas. Los límites de apalancamiento se basan en la capacidad financiera del trader y su tolerancia al riesgo.

5. Uso de órdenes de stop-loss: Las órdenes de stop-loss son herramientas clave para establecer límites de riesgo en el criptotrading. Estas órdenes se colocan a un precio predeterminado y se activan cuando el precio de la criptomoneda alcanza ese nivel. Las órdenes de stop-loss permiten al trader limitar las pérdidas y cerrar automáticamente una posición si el mercado se mueve en su contra.

6. Monitoreo y ajuste continuo: Es importante monitorear regularmente los límites de riesgo establecidos y ajustarlos según sea necesario. Los mercados de criptomonedas son volátiles y pueden experimentar cambios rápidos, por lo que es necesario adaptar los límites de riesgo para reflejar las condiciones actuales del mercado y los cambios en la tolerancia al riesgo del trader.

Establecer límites de riesgo en el criptotrading es esencial para proteger el capital y evitar pérdidas significativas. Cada trader debe establecer límites de riesgo personalizados que se ajusten a sus circunstancias y objetivos.

Ahora sí vamos a ver herramientas de ML aplicadas a la gestión de riesgos.

### 2.5.- Herramientas de ML en la Gestión de Riesgos:

Las siguientes son solo algunas herramientas de ML en la gestión de riesgo, veremos cada una de ellas más a detalle, así como también buscaremos herramientas específicas en cada punto de los que se mencionan a continuación.

En la gestión de riesgos del criptotrading, se utilizan diversas herramientas de Machine Learning (ML) para ayudar en el análisis y la toma de decisiones. Aquí tienes algunas herramientas de ML comúnmente utilizadas en la gestión de riesgos del criptotrading:

1. Modelos de predicción de volatilidad: Los modelos de predicción de volatilidad utilizan algoritmos de ML para predecir la volatilidad futura de los precios de las criptomonedas. Estos modelos pueden ayudar a identificar y evaluar el riesgo de fluctuaciones extremas en los precios, lo que permite a los traders tomar decisiones informadas sobre la gestión de riesgos.

2. Modelos de clasificación de riesgos: Los modelos de clasificación de riesgos utilizan algoritmos de ML para categorizar diferentes niveles de riesgo en base a diferentes variables y características del mercado. Estos modelos pueden ayudar a identificar las operaciones de mayor riesgo y tomar decisiones basadas en esa clasificación, como establecer límites de pérdida o reducir la exposición a ciertos activos.

3. Análisis de sentimiento: El análisis de sentimiento utiliza técnicas de procesamiento de lenguaje natural (NLP) y algoritmos de ML para analizar y clasificar la opinión y el sentimiento en los medios de comunicación y las redes sociales en relación con las criptomonedas. Esto puede ayudar a evaluar el sentimiento del mercado y anticipar posibles cambios en la tendencia, lo que a su vez puede influir en las decisiones de gestión de riesgos.

4. Análisis de correlación: El análisis de correlación utiliza algoritmos de ML para identificar y medir las correlaciones entre diferentes variables financieras y de mercado. Esto puede ayudar a comprender las relaciones entre los precios de las criptomonedas y otros factores, como noticias económicas o indicadores macroeconómicos. El análisis de correlación puede ser útil para identificar riesgos relacionados con la dependencia de ciertos factores y tomar decisiones de gestión de riesgos adecuadas.

5. Modelos de aprendizaje por refuerzo: Los modelos de aprendizaje por refuerzo utilizan algoritmos de ML para aprender y optimizar las decisiones de trading en función de las recompensas y penalizaciones recibidas. Estos modelos pueden ayudar a automatizar el proceso de gestión de riesgos al ajustar automáticamente las estrategias de trading en función de las condiciones del mercado y el rendimiento histórico.

Estas son solo algunas de las herramientas de ML utilizadas en la gestión de riesgos del criptotrading. La elección de la herramienta adecuada depende de los objetivos y las necesidades específicas de cada trader o empresa. Es importante tener en cuenta que el uso de estas herramientas debe complementarse con un análisis exhaustivo y la experiencia en el mercado financiero para una toma de decisiones sólida y precisa.

#### 2.5.1.- Modelos de predicción de volatilidad:

#### 2.5.2.- Modelos de clasificación de riesgos:

Los modelos de predicción de volatilidad son herramientas utilizadas en la gestión de riesgos del criptotrading para estimar la volatilidad futura de los precios de las criptomonedas. Estos modelos se basan en algoritmos de Machine Learning (ML) y estadísticas para analizar el comportamiento histórico de los precios y realizar proyecciones sobre la volatilidad futura. Aquí tienes más detalles sobre los modelos de predicción de volatilidad y algunos ejemplos:

1. Modelos de volatilidad estocástica: Estos modelos asumen que la volatilidad de los precios es una variable aleatoria que sigue ciertos patrones y procesos estocásticos. Ejemplos de modelos de volatilidad estocástica incluyen el modelo GARCH (Generalized Autoregressive Conditional Heteroskedasticity) y sus variantes como EGARCH, TGARCH, entre otros. Estos modelos utilizan el análisis de la serie temporal para estimar la volatilidad futura en función de la información histórica.

2. Modelos basados en redes neuronales: Las redes neuronales, como las Redes Neuronales Recurrentes (RNN) y las Redes Neuronales Convolucionales (CNN), se utilizan para capturar patrones complejos en los datos históricos de precios y volatilidad. Estas redes pueden aprender relaciones no lineales y realizar predicciones de volatilidad basadas en patrones identificados en los datos.

3. Modelos de volatilidad implícita: La volatilidad implícita se refiere a la volatilidad esperada por el mercado y se deriva de los precios de las opciones. Los modelos de volatilidad implícita, como el modelo Black-Scholes, se utilizan para inferir la volatilidad futura en función de los precios de las opciones y otros factores relacionados. Estos modelos son especialmente útiles para la gestión de riesgos en derivados financieros.

Es importante destacar que la predicción de volatilidad en el criptotrading es un desafío debido a la naturaleza altamente volátil y no lineal de los precios de las criptomonedas. Por lo tanto, es recomendable combinar múltiples enfoques y utilizar técnicas adicionales, como la gestión de riesgos basada en límites de pérdida y diversificación de la cartera, para una gestión de riesgos efectiva.

#### 2.5.3.- Análisis de sentimiento:

El análisis de sentimiento es un factor importante también en la predicción de precios, sin embargo, y aunque lo vayamos a describir brevemente en esta sección, será de las últimas técnicas que utilizaremos en nuestro proyecto.

El análisis de sentimiento aplicado a la gestión de riesgo en el criptotrading es una técnica que utiliza técnicas de procesamiento del lenguaje natural (NLP) y aprendizaje automático para evaluar las emociones y opiniones expresadas en textos relacionados con las criptomonedas, como noticias, publicaciones en redes sociales y comentarios de usuarios. El objetivo es capturar el sentimiento general del mercado y utilizar esta información para mejorar la toma de decisiones y gestionar los riesgos asociados al trading de criptomonedas. A continuación, se detalla más a fondo cómo funciona este enfoque:

1. Recopilación de datos: Se recopilan datos de diversas fuentes, como redes sociales, foros de discusión, noticias financieras y blogs, que contengan información relevante sobre las criptomonedas y su mercado. Estos datos pueden ser obtenidos a través de web scraping, APIs o servicios de terceros.

2. Preprocesamiento de texto: Antes de realizar el análisis de sentimiento, los datos textuales deben ser procesados para eliminar ruido y estandarizar el formato. Esto incluye eliminar palabras irrelevantes (stopwords), puntuación, símbolos especiales y realizar tareas como tokenización, lematización y stemming para normalizar el texto.

3. Análisis de sentimiento: Una vez que los datos están preparados, se aplica el análisis de sentimiento para determinar la polaridad (positiva, negativa o neutral) de cada texto. Este análisis puede ser basado en reglas o utilizar modelos de aprendizaje automático, como clasificadores o modelos basados en redes neuronales, entrenados en conjuntos de datos etiquetados.

4. Agregación y seguimiento del sentimiento: El sentimiento de cada texto se agrega y se realiza un seguimiento a lo largo del tiempo. Esto permite identificar tendencias y cambios en el sentimiento del mercado. Se pueden utilizar técnicas de suavizado o modelos de series temporales para obtener una visión más clara de la evolución del sentimiento.

5. Integración con la gestión de riesgos: El análisis de sentimiento se combina con otras herramientas de gestión de riesgos, como modelos de predicción de volatilidad y análisis técnico, para obtener una visión más completa y precisa del riesgo asociado al trading de criptomonedas. Se pueden establecer estrategias de gestión de riesgos en función de las señales de sentimiento, como ajustar el tamaño de la posición o activar alertas de riesgo.

Algunas técnicas y herramientas utilizadas en el análisis de sentimiento aplicado a la gestión de riesgo en el criptotrading incluyen:

- Análisis de polaridad basado en léxicos: Se utilizan listas de palabras con polaridad asignada para clasificar los textos según su sentimiento.

- Clasificación supervisada: Se entrenan modelos de clasificación, como clasificadores bayesianos ingenuos o máquinas de vectores de soporte (SVM), utilizando conjuntos de datos etiquetados para predecir el sentimiento de nuevos textos.

- Modelos basados en redes neuronales: Se utilizan modelos como las redes neuronales convolucionales (CNN) o las redes neuronales recurrentes (RNN) para capturar características más complejas y contextuales en el texto.

- Modelos de aprendizaje no supervisado: Se utiliz

an algoritmos de clustering o técnicas de análisis de tópicos para agrupar textos con sentimientos similares y descubrir patrones en los datos.

- Herramientas y bibliotecas de NLP: Se pueden emplear bibliotecas como NLTK, spaCy o Stanford NLP para realizar tareas de procesamiento de texto y análisis de sentimiento.

Es importante tener en cuenta que el análisis de sentimiento no es una técnica infalible y está sujeto a ciertas limitaciones, como la ambigüedad del lenguaje y la dificultad para capturar el contexto completo. Por lo tanto, es recomendable combinar el análisis de sentimiento con otras herramientas y enfoques de gestión de riesgos para tomar decisiones más informadas en el criptotrading.

Ahora que veo un poco más sobre el enfoque del análisis de sentimiento, me queda claro que es mejor enfocarse en las noticias que en redes sociales o foros, puesto que las noticias son esporádicas y no requieren tanto poder computacional, por lo que se puede aplicar el análisis de sentimiento solo a las noticias referentes al criptomercado.

#### 2.5.4.- Análisis de correlación:

#### 2.5.5.- Modelos de aprendizaje por refuerzo:

#### Análisis de Hiperparámetros:

Un hiperparámetro es un parámetro externo a un modelo de aprendizaje automático que se elige antes de iniciar el proceso de entrenamiento. A diferencia de los parámetros del modelo, que se aprenden durante el entrenamiento, los hiperparámetros son configurables y se establecen antes de que comience el proceso de optimización.

Los hiperparámetros definen aspectos del algoritmo de aprendizaje automático y afectan el rendimiento y el comportamiento del modelo. Son decisiones que toma el desarrollador o el científico de datos para configurar el algoritmo y personalizarlo para un problema y conjunto de datos específicos.

Algunos ejemplos comunes de hiperparámetros incluyen:

1. Tasa de aprendizaje: Es un hiperparámetro que determina la cantidad de ajuste que se realiza durante cada iteración del algoritmo de optimización. Controla la velocidad a la que el modelo aprende de los datos.

2. Número de capas ocultas: En las redes neuronales, el número de capas ocultas es un hiperparámetro que define la cantidad de capas intermedias entre la capa de entrada y la capa de salida. Este hiperparámetro afecta la complejidad y capacidad del modelo para aprender representaciones más abstractas.

3. Número de árboles en un bosque aleatorio: En algoritmos de ensamble como Random Forest, el número de árboles es un hiperparámetro que determina la cantidad de árboles de decisión utilizados en el modelo.

4. Tamaño del lote (batch size): En el entrenamiento de redes neuronales, el tamaño del lote es el número de muestras de entrenamiento que se utilizan en cada iteración del algoritmo de optimización. Es un hiperparámetro que puede afectar la estabilidad y velocidad del entrenamiento.

5. Regularización: La regularización es una técnica utilizada para evitar el sobreajuste en los modelos de aprendizaje automático. Los hiperparámetros de regularización, como la fuerza de la regularización L1 o L2, controlan la cantidad de penalización aplicada a los coeficientes del modelo.

La selección adecuada de los hiperparámetros es esencial para obtener un modelo de aprendizaje automático bien ajustado y con un buen rendimiento. Por lo general, se realizan experimentos y validaciones cruzadas con diferentes combinaciones de valores de hiperparámetros para encontrar la configuración óptima. Existen técnicas de optimización de hiperparámetros, como la búsqueda de cuadrícula (grid search) y la optimización bayesiana, que ayudan a encontrar los mejores valores de hiperparámetros de manera más eficiente..