TREBALL FINAL DE GRAU

APLICACIÓ AUTOMÀTICA D'ESTRATÈGIES POTENCIADES AMB INTEL·LIGÈNCIA ARTIFICIAL EN EL TRADING DE CRIPTOMONEDES

JESÚS ROJAS MORALES

31/01/2023

COGNOMS:	ROJAS MORALES	**NOM:**	JESÚS
TITULACIÓ:	GRAU EN ENGINYERIA INFORMÀTICA		
PLA:	2018		
DIRECTOR:	GODOY BALIL, GUILLERMO		
DEPARTAMENT:	CIÈNCIES DE LA COMPUTACIÓ		

QUALIFICACIÓ DEL TFG

TRIBUNAL

PRESIDENT	**SECRETARI**	**VOCAL**
ROMAN JIMENEZ, JOSÉ ANTONIO	ESTEVE CUSINE, JORDI	MASSANA HUGAS, IMMACULADA

DATA DE LECTURA: 07/02/2023

Aquest Projecte té en compte aspectes mediambientals: ☐ **Si** ☑ **No**

Taula de continguts

Glossari 7
1. Introducció 10
2. Estudi sobre la idoneïtat de models de regressió 12
3. Estudi sobre la idoneïtat de models de classificació 17
4. Estudi sobre la idoneïtat dels indicadors tècnics 24
    4.1. Implementació de la classe “tester” 25
    4.2. Estratègia basada en triple SMA 33
        4.2.1. Càlcul i exploració visual dels indicadors SMA 33
        4.2.2. Backtesting de l'estratègia basada en triple SMA 34
        4.2.3. Optimització de l'estratègia basada en triple SMA 36
        4.2.4. Rendiment mensual mitjà de l'estratègia basada en triple SMA 37
    4.3. Estratègia basada en RSI i Bandes de Bollinger 38
        4.3.1. Càlcul i exploració visual dels indicadors RSI i Bandes de Bollinger 38
        4.3.2. Backtesting de l'estratègia basada en RSI i Bandes de Bollinger 39
        4.3.3. Optimització de l'estratègia basada en RSI i Bandes de Bollinger 41
        4.3.4. Rendiment mensual mitjà de l'estratègia basada en RSI i Bandes de Bollinger 42
    4.4. Estratègia basada en MACD i DEMA 43
        4.4.1. Càlcul i exploració visual dels indicadors MACD i DEMA 44
        4.4.2. Backtesting de l'estratègia basada en MACD i DEMA 45
        4.4.3. Optimització de l'estratègia basada MACD i DEMA 47
        4.4.4. Rendiment mensual mitjà de l'estratègia basada en MACD I DEMA 48
    4.5. Estratègia basada en SQUEEZM i ADX 49
        4.5.1. Càlcul i exploració visual dels indicadors SQUEEZM i ADX 50
        4.5.2. Backtesting de l'estratègia basada en SQUEEZM i ADX 51
        4.5.3. Optimització de l'estratègia basada SQUEEZM i ADX 53
        4.5.4. Rendiment mensual mitjà de l'estratègia basada en SQUEEZM i ADX 54
5. Desenvolupament dels robots 55
    5.1. Implementació de la classe “trader” 56
    5.2 Especificació d’una prova en temps real pels robots finals 61
6. Conclusions 63
7. Bibliografia 64

Resum

En aquest treball ens aproparem al món de la inversió, implementant algoritmes per fer **trading** amb **criptomonedes**. El trading consisteix en la compravenda estratègica d'un actiu, en aquest cas una criptomoneda, amb l'objectiu de guanyar diners.

La motivació principal del projecte és automatitzar aquesta tasca. Per tal d'aconseguir-ho, desenvolupem **robots** de trading, que són executables que reben actualitzacions del preu cada pocs segons i han de comprar o vendre automàticament mirant de generar beneficis.

Els robots plantejats pel projecte generen un **senyal de trading** que pren valors del conjunt [0,1,-1] on cadascun dels símbols correspon a una acció: 0 vol dir no fer res, 1 vol dir comprar i -1 vol dir vendre. Un bon senyal guanyarà diners i un senyal dolent perdrà diners.

Al treball s'estudien diverses aproximacions a aquest problema. Mirem de valorar la idoneïtat de cada mètode a priori, i prenem la decisió de si fer-los servir o no per la implementació final dels robots.

Un dels mètodes considerats es basa en fer prediccions del preu amb un **model de regressió**. Determinem fins a quin punt el preu d'una criptomoneda té autocorrelació lineal i generem models per tal d'avaluar-ne l'efectivitat.

També estudiem un mètode basat en crear el senyal amb un **model de classificació**. En aquest apartat mirem de categoritzar les dades del preu en moments de compra, venda o neutralitat.

L'últim mètode que estudiem es basa en **indicadors tècnics**, que son càlculs derivats del preu, i que poden ajudar a detectar oportunitats de compravenda. Cada estratègia és tunejada i testejada amb dades històriques. És el que anomenem **backtesting**.

Per últim, prenem una decisió formada de quin mètode utilitzar i portem a terme la implementació final dels robots. Descrivim el test final al **cloud** i en temps real, per a posar a prova el seu funcionament.

Paraules clau:

trading	criptomonedes	robots	senyal de trading
model de regressió	model de classificació	indicadors tècnics	backtesting
cloud			

Abstract

In this project, we get into the world of investment by implementing algorithms for **trading** with **cryptocurrencies**. Trading consists of the strategic buying and selling of an asset, in this case a cryptocurrency, with the aim to make money.

The main motivation of the project is to automate this task. With this goal in mind, we develop trading **robots**, which are executables that receive price updates every few seconds and automatically buy or sell trying to generate profits.

The robots proposed in this project generate a **trading signal** from the set [0,1,-1], where each symbol corresponds to a doable action: 0 means do nothing, 1 means buy and -1 means sell. A good signal will make money and a bad signal will lose money.

In this project, we analyze several approaches and assess their suitability for generating a good signal, and make the decision of whether to use them or not for the final implementation of the robots.

One of the methods consists of predicting the price evolution by means of a **regression model**. We determine to what extent a cryptocurrency price has linear autocorrelation and generate models to evaluate its effectiveness.

The approach of creating the signal with a **classification model** is also studied. In this case, we try to categorize the price data into buying, selling or neutrality times.

The last studied method is based on technical indicators, which are price derived calculations used to detect buying and selling opportunities. Each strategy is tuned and tested with historical data, by simulating how they would have performed in the past. This is what we call **backtesting**.

Finally, we make a well-informed decision on which method to use for the final implementation of the robots. We describe the final test, which is carried out in the **cloud** and in real-time, to evaluate the functioning of the robots.

Key words:

trading	cryptocurrencies	robots	trading signal
regression model	classification model	technical indicators	backtesting
cloud			

Glossari

Amb l'objectiu d'aclarir el significat d'alguns termes que són utilitzats en les pròximes pàgines definim:

- **Actiu/Instrument financer:** és un producte digital que pots comprar i vendre per guanyar diners.
- **Trading:** activitat financera que consisteix a comprar i vendre estratègicament instruments financers amb l'objectiu de generar beneficis.
- **Criptomonedes:** un tipus d'instrument financer. Són monedes digitals intercanviables per diners mitjançant transaccions. El seu nom prové del fet que les transaccions estan assegurades i verificades amb criptografia.
- **Robot de trading:** programa executable que fa compres i vendes d'un actiu de forma automàtica amb l'objectiu de generar beneficis.
- **Posició a curt:** adoptar una posició a curt, significa que estàs comprant una quantitat determinada d'un instrument financer amb la intenció de vendre'l a un preu més alt en el futur. La idea és guanyar diners amb la diferència entre el preu que vas pagar i el preu al qual ho vendràs.
- **Posició a llarg:** adoptar una posició a llarg, significa que estàs venent una quantitat determinada d'un instrument financer amb la intenció de comprar-lo a un preu més baix en el futur. La idea és guanyar diners amb la diferència entre el preu al qual ho vas vendre i el preu al qual el compraràs.
- **Gràfica d'espelmes japoneses**: és un tipus de gràfic utilitzat per representar el preu i la volatilitat d'un actiu financer en un període determinat. Cada espelma representa un període de temps, i la seva mida i forma indiquen el preu d'obertura, de tancament, el màxim i el mínim durant aquest període.
- **Retorn positiu:** és una variació del preu a l'alça. Les espelmes amb un preu d'obertura inferior al preu de tancament indiquen que el retorn del període és positiu. Els retorns positius, quan estem en una posició a curt signifiquen beneficis. Per contra, si estem en una posició a llarg, signifiquen pèrdues.
- **Retorn negatiu:** és una variació del preu a la baixa. Les espelmes amb un preu d'obertura superior al preu de tancament indiquen que el retorn del període és negatiu. Els retorns negatius, quan estem en una posició a llarg, signifiquen beneficis. Per contra, si estem en una posició a curt, signifiquen pèrdues.
- **Dades OCHL:** és un conjunt de dades format per registres datats del preu d'un actiu. Cada registre correspon a un període de temps representable per una espelma i conté preus d'obertura, clausura, màxim i mínim del període.
- **Senyal de trading:** és un senyal continu que es genera a partir de les dades OCHL i que pren valors del conjunt [0,1,-1] on cadascun dels símbols correspon a una acció: 0 vol dir no fer res, 1 vol dir comprar alguna quantitat d'un actiu i -1 vol dir vendre alguna quantitat d'un actiu.

- **Model de regressió**: és un model de IA que fa prediccions d'una variable tractant d'establir una relació lineal entre valors passats i futurs de la variable.
- **Autocorrelació**: és un càlcul estadístic que indica el grau de relació lineal existent entre dos valors diferents que pertanyen a una sèrie.
- **Model de classificació:** és un tipus de model de IA que aprèn amb un conjunt de dades d'entrenament a categoritzar noves mostres del conjunt.
- **Etiquetació:** és el procés d'assignar les categories o classes a cada mostra del conjunt d'entrenament perquè el model de classificació prengui l'exemple de quines són les respostes correctes i aprengui a categoritzar noves mostres.
- **Estratègia de trading:** és un conjunt de regles a seguir per generar un senyal de trading a partir de les dades OCHL d'un actiu. Cada estratègia genera un senyal de trading únic.
- **Rendiment financer:** és una mesura dels beneficis o pèrdues que produeix una activitat financera com l'aplicació d'una estratègia de trading.
- **Backtesting:** El backtesting és el procés de simular accions de compravenda utilitzant dades històriques del preu, per avaluar la viabilitat d'una estratègia de trading.
- **Múltiple d'inversió:** càlcul que mesura la rendibilitat d'una inversió indicant factor pel qual s'ha multiplicat la inversió inicial. Per exemple en una inversió on en la inversió inicial és 1$ i el capital final són 2$, el múltiple d'inversió els 2.
- **CAGR:** càlcul que mesura el creixement mitjà d'una inversió al cap d'un any assumint la reinversió continua dels beneficis.
- **Sharpe:** és un càlcul que posa en relació tant la rendibilitat, amb el retorn positiu mitjà, com el risc, amb la volatilitat que mostren els retorns, d'una inversió.
- **Indicador tècnic**: és una eina utilitzada per prendre decisions de trading. Les dades OCHL són processades per calcular un indicador tècnic que ens permet identificar tendències i patrons en el mercat.
- **SMA**: també coneguda com a Simple Moving Average o Mitjana Mòbil Simple, és un indicador tècnic que elimina les fluctuacions del preu per donar la tendència general del preu.
- **MACD:** també conegut com a Moving Average Convergence Divergence o Convergència i Divergència de Mitjanes Mòbils, és un indicador que combina dues mitjanes mòbils diferents per detectar els canvis de tendència.
- **DEMA:** també conegut com a Double Exponential Moving Average és un indicador molt semblant a la SMA, però en el seu càlcul es dona més pes als valors més recents del preu. Així és més sensible als canvis del preu.
- **Bollinger Bands:** són un indicador tècnic format per tres bandes. Una intermèdia que és una SMA i dues bandes, superior i inferior que s'allunyen o s'aproximen a la intermèdia en funció de la volatilitat que presenti el mercat

- **SQUEEZM**: també conegut com a Squeeze Momentum és un indicador tècnic que permet diferenciar els moments en els quals el mercat es mou amb una tendència clara o quan està en un període de consolidació.
- **ADX:** també conegut com a Average Directional Index, és un indicador tècnic que mesura la força de la tendència actual en el mercat.
- **Bid-Ask Spread**: quan fas una operació, és un cost afegit producte de la diferència mitjana entre el preu al qual vols fer l'operació i el millor preu que et pot oferir la plataforma de trading on operes. És interpretable com costos addicionals derivats del trading que s'han de tenir en compte.

1. Introducció

Durant l'any 2008, un grup d'experts en criptografia començaren a teoritzar sobre la descentralització de les finances. Van desenvolupar un sistema que no depenia d'intermediaris confiables com bancs o governs. Aquest sistema va passar a ser conegut com a blockchain, una tecnologia que permet registrar transaccions en una base de dades distribuïda i segura, que és actualitzada constantment.

Actualment, la blockchain està a l'ordre del dia i des que va sorgir ha generat debat i expectació al món tecnològic. En gran part, això és degut al fet que una de les seves aplicacions, les criptomonedes, s'han consolidat fermament com una extensió dels mercats financers tradicionals.

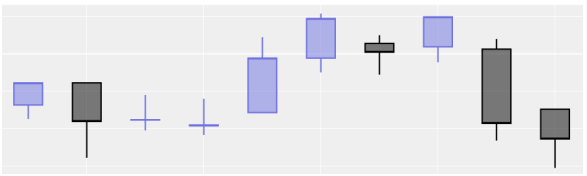
Les criptomonedes o criptos són bàsicament divises digitals que utilitzen mètodes de criptografia per assegurar les transaccions. En aquest treball ens aproparem al món de la inversió, implementant algoritmes per fer trading amb criptomonedes.

El trading consisteix en la compravenda estratègica d'un actiu, en aquest cas una criptomoneda. Es mira de comprar criptomonedes a un preu baix i vendre-les quan estan a un preu alt. Així es guanyarien diners. És una activitat que requereix atenció contínua, ja que segons com evoluciona el preu de la criptomoneda cal prendre diferents decisions de compravenda.

La motivació principal d'aquest treball és automatitzar aquesta presa de decisions. Per aconseguir-ho desenvoluparem robots de trading, que són executables que realitzen de forma automàtica tota aquesta tasca. Aquests robots rebran recurrentment actualitzacions del preu de la criptomoneda. Cada actualització forma part d'una sèrie que tindrà la següent forma:

Date	Open	High	Low	Close
2017-08-17 04:00:00	4261.4800	4313.6200	4261.3200	4308.8300
2017-08-17 05:00:00	4308.8300	4328.6900	4291.3700	4315.3200
2017-08-17 06:00:00	4330.2900	4345.4500	4309.3700	4324.3500
2017-08-17 07:00:00	4316.6200	4349.9900	4287.4100	4349.9900
2017-08-17 08:00:00	4333.3200	4377.8500	4333.3200	4360.6900

Aquestes dades les anomenarem dades OCHL. Gràcies a rebre actualitzacions d'aquestes dades els robots coneixeran el preu d'obertura, clausura, màxim i mínim de cada període o temporalitat. Temporalitats que són representables per una espelma en un gràfic d'espelmes japoneses com el següent:



El que miraran de fer els robots és predir si el preu pujarà o baixarà. Una forma simple de veure-ho és que els robots han de generar un senyal continu que pren valors del conjunt [0, 1, -1], on cadascun dels símbols correspon a una acció realitzable quan s'opera en el mercat:

- **0**: vol dir que el robot ha de tancar operacions obertes i mantenir-se sense fer res.
- **1**: vol dir que el robot ha de comprar alguna quantitat de l'actiu, és a dir, adoptar el que anomenem una posició a llarg.
- **-1**: vol dir que el robot ha de vendre alguna quantitat de l'actiu, és a dir, adoptar el que anomenem una posició a curt.

El senyal determina la capacitat del robot per guanyar diners. Això és degut al fet que en el càlcul dels beneficis el senyal fa de signe per les variacions en el preu que presenta el mercat.

Plantegem un exemple per fer-ho més entenedor. Imaginem que el robot compra perquè ha generat un 1. Llavors cada nova espelma positiva o increment de preu significarà que guanyem diners. Per contra, cada nova espelma negativa o decrement del preu significaran pèrdues. Per tant, com més s'adapti el senyal generat als canvis del mercat més beneficis guanyarà el robot.

Com a passes prèvies a la implementació dels robots, en aquest projecte estudiarem la idoneïtat de diversos mètodes per generar aquests senyals de compravenda. Valorarem la idoneïtat i el rendiment que ens ofereix cada mètode i decidirem si han de ser descartats o seleccionats per la implementació final dels robots.

Respecte al llenguatge de programació escollit pel projecte, s'han considerat diverses opcions com R, C++ o Java. Finalment ens hem decantat per Python. Python és un llenguatge versàtil que té una gran quantitat de paquets de ciència de dades i intel·ligència artificial disponibles per a la manipulació, neteja i anàlisi de grans conjunts de dades. A més, té una sintaxi senzilla i una comunitat molt activa que contribueix en l'evolució i millora d'aquestes eines. Això fa que sigui fàcil d'aprendre i implementar per a molts usuaris, incloent-hi data scientists, analistes de dades i desenvolupadors.

Per últim, cal aclarir que es vol és testejar la bondat de sistemes concrets pel trading de criptomonedes. El fet que aquests es comportin millor o pitjor no serà el criteri que farem servir per avaluar la qualitat del projecte. És a dir, interpretarem l'èxit financer com un símptoma del bon treball fet, però en cap cas com un objectiu directe.

2. Estudi sobre la idoneïtat de models de regressió

El primer tipus de models que tractarem són els models de regressió. Aquests es fonamenten en el fet que existeix una relació lineal entre el valor a predir i els valors coneguts del passat de la variable. La qüestió llavors és, en quin grau això és cert pel preu d'una criptomoneda. Només ho podem saber investigant l'autocorrelació de la variable, mètrica que mesura la relació lineal.

En el nostre cas, un alt grau d'autocorrelació en el preu indicaria que els preus futurs estan linealment relacionats amb els preus passats, i per tant un model de regressió entrenat amb els preus passats podria ser efectiu per predir els preus futurs. Altrament, si el preu no presenta autocorrelació, el model podria no ser efectiu perquè no hi ha una relació lineal clara entre els valors passats i futurs.

En resum, l'autocorrelació és un factor important a tenir en compte abans de construir un model de regressió, ja que pot indicar-nos quina serà la seva efectivitat de les seves prediccions. Definim la següent hipòtesi per veure si és refutada:

> *El preu del Bitcoin presenta una autocorrelació apta pels models de regressió, per tant, els models de regressió són aptes per donar solució al nostre problema.*

Ara explorarem l'autocorrelació de la variable a predir, és a dir, el preu d'una criptomoneda. Per aquest projecte hem escollit el Bitcoin perquè és la criptomoneda més coneguda. Els resultats poden variar depenent de la criptomoneda, però el Bitcoin ens servirà per fer-nos una idea de quin és el potencial de cada sistema que plantegem.

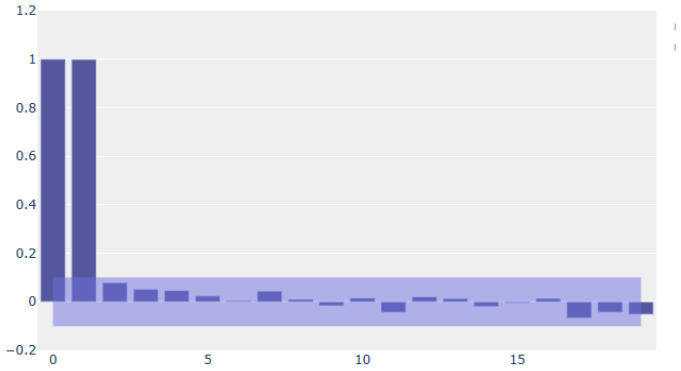
A continuació mostrem el càlcul i l'exploració visual de l'autocorrelació parcial dels preus de clausura per un conjunt de dades OCHL, que recordem que són les dades inicials del problema:

```
# Gráfic d'autocorrelació parcial
fig = make_subplots(rows = 1, cols = 1)
x=list(range(0, 100, 1))
ypos=np.full(100, 0.1)
yneg=np.full(100, -0.1)
fig.add_trace(go.Scatter(
    mode = 'lines', x=x[0:20], y=ypos, fill='tozeroy', line=dict(color=colors['soft'], width=0.1)), row=1, col=1)
fig.add_trace(go.Scatter(
    mode = 'lines', x=x[0:20], y=yneg, fill='tozeroy', line=dict(color=colors['soft'], width=0.1)), row=1, col=1)

# Càlcul
df_pacf = pacf(data.loc[0:, 'Close'], nlags=20, method = 'ols')

fig.append_trace(go.Bar(
    x=np.arange(len(df_pacf)-1), y= df_pacf, marker_color=colors['base'], name= 'PACF lag{}'.format(lag)), row=1, col=1
)

layout = go.Layout(plot_bgcolor=colors['bg'], height=500, width=800)
fig.update_layout(layout)
fig.update_yaxes(range = [-0.2, 1.2])
fig.show()
```



D'aquesta forma mesurem i visualitzem la relació lineal que hi ha entre dos valors separats en el temps sense tenir en compte les relacions intermèdies. Observem que només tenim valors d'autocorrelació significativa pels instants de temps 0 i 1. Per tant, del gràfic podem extreure que cada valor del preu només té relació amb el valor immediatament anterior. Això ens porta a pensar que els models de regressió no són aptes pel nostre problema. De totes maneres, crearem alguns models per mesurar el seu rendiment.

Com a pas previ a la creació dels models de regressió, generarem un conjunt de prediccions quasi aleatòries. Aquestes prediccions de referència les generarem fonamentant-nos en la teoria del *Random Walk* sobre els mercats borsaris, que estableix que la millor predicció possible del preu és l'últim valor conegut de la sèrie més un error impredictible.

D'aquesta forma tindrem una referència amb què comprar els nostres resultats. Hem convertit el propòsit d'aquest estudi en crear un model de regressió que sigui capaç de preveure els preus futurs amb una millor precisió que simplement utilitzar el darrer preu conegut més un error impredictible. A continuació mostrem el codi amb el qual generem les prediccions*:*

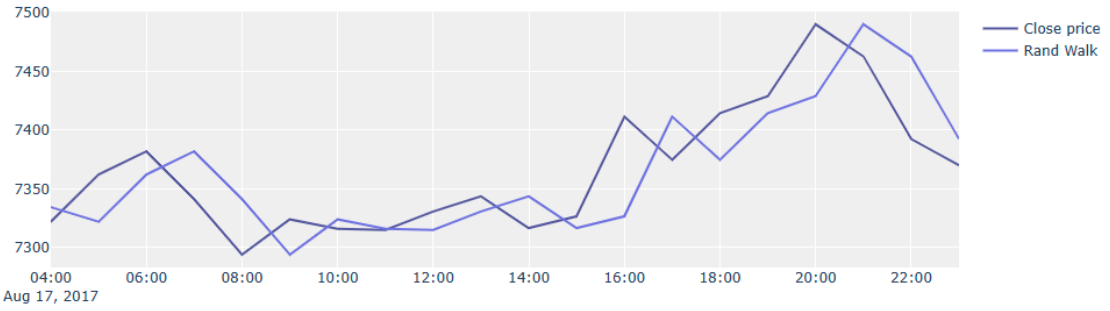
```
# Generem una simulació Random Walk
data_rw = data[['Date', 'Close']].copy()
data_rw['Prediction'] = data_rw['Close'].shift(1)

# Error de test
y_verdadera = data_rw.loc[fin_train:, 'Close']
y_prediccio = data_rw.loc[fin_train:, 'Prediction']
metrica_rw = mean_absolute_percentage_error(y_verdadera, y_prediccio)*100
print(f'Error del test: {metrica_rw}')
display(data_rw.loc[fin_train:,].head(4))
```

```
Error del test: 0.36460550603114894
```

	Date	Close	Prediction
7999	2018-07-18 21:00:00	7321.9500	7334.4200
8000	2018-07-18 22:00:00	7362.1900	7321.9500
8001	2018-07-18 23:00:00	7381.9000	7362.1900
8002	2018-07-19 00:00:00	7341.2500	7381.9000

En el codi la funció *shift* de la llibreria *pandas* desplaça els preus una posició endavant i així les prediccions de referència satisfan la base plantejada: *la millor predicció possible del preu és l'últim valor conegut de la sèrie més un error impredictible*. Després calculem l'error mitjà de les prediccions del model i ens dona un error aproximat del 0,36% pel conjunt de dades seleccionat. Aquest és l'error que haurien de millorar els models de regressió per considerar que són capaços de predir els preus en algun grau. Observem les prediccions de referència comparades gràficament amb els valors reals del preu:



Ara procedim a la creació dels models. Hem utilitzat un regressor anomenat LightGBM (Light Gradient Boosting Machine) que ofereix un bon rendiment a un cost temporal menor que altres opcions investigades per aquest projecte.

Del primer aspecte que ens encarreguem és del que anomenem optimització d'hiperparàmetres. Els hiperparàmetres són valors configurables en un model de IA que afecten el seu comportament i rendiment. Són específics per a cada model i han de ser seleccionats manualment abans d'entrenar el model. La selecció adequada dels hiperparàmetres és crucial per al rendiment del model. Observem a continuació la cerca dels hiperparàmetres:

```
# Optimització dels hiperparametres déntrada
grid_params = {
    'learning_rate': [0.1, 0.01, 0.2, 0.02, 0.3, 0.03],
    'num_leaves': [6, 8, 12, 16, 32, 64],
    'boosting_type' : ['gbdt'],
    'objective' : ['regression'],
    'max_bin':[256, 512],
    'random_state' : [123],
    'colsample_bytree' : [0.64, 0.65, 0.66],
    'reg_alpha' : [1,1.2],
    'reg_lambda' : [1,1.2,1.4],
}

grid = GridSearchCV(lgbm.LGBMRegressor(random_state=123), grid_params, cv=5)
grid.fit(X_train, y_train)
model_params = {
    'learning_rate': grid.best_params_['learning_rate'],
    'num_leaves': grid.best_params_['num_leaves'],
    'boosting_type' : grid.best_params_['boosting_type'],
    'objective' : grid.best_params_['objective'],
    'max_bin': grid.best_params_['max_bin'],
    'random_state' : grid.best_params_['random_state'],
    'colsample_bytree' : grid.best_params_['colsample_bytree'],
    'reg_alpha' : grid.best_params_['reg_alpha'],
    'reg_lambda' : grid.best_params_['reg_lambda']
}
model_params
```

Ara només ens queda procedir a l'entrenament i el test dels models. Per dur a terme aquests creem una instància de model, definim els conjunts d'entrenament i de test i procedim a executar tant l'entrenament com el test, mesurant l'error mitjà que presenta cada iteració de model de regressió en les seves prediccions. Parlem d'iteracions perquè hem creat models per fer prediccions de t1 (valor immediatament següent), t5 i t10 (valors a més llarg termini). Observem el codi amb els passos esmentats:

```
for lag in [1, 5, 10]:

    df = data.copy()
    for i in list(range(1, lag+1, 1)):
        df['Lag{}'.format(i)] = df['Close'].shift(i)

    X_train, X_test, y_train, y_test = train_test_split(
        df.drop(['Date', 'High', 'Low', 'Close', 'Volume'], axis=1),
        df.Close, test_size=0.2
    )

    train_data = LGBM.Dataset(X_train, y_train)
    test_data = LGBM.Dataset(X_test, y_test)

    # Entrenament del model de regressio
    model = LGBM.train(model_params, train_data, valid_sets=test_data, num_boost_round=100)

    # Testing i puntuació del model
    y_pred = model.predict(X_test)
    metrica = mean_absolute_percentage_error(y_test, y_pred)*100
    modelo = 'LGBMRegressor{}'.format(lag)

    df_errores = pd.concat(
        [df_errores, pd.DataFrame([{'Modelo': modelo, 'Lags': lag, 'Error': metrica}])],
        axis=0, ignore_index=True
    )
```

Una vegada mesurats els rendiments recullim la informació de tots els models de regressió creats en una taula per procedir a l'anàlisi dels resultats. A continuació mostrem la taula esmentada:

```
df_errores
```

	Modelo	Lags	Error
0	Random Walk	1	0.3646
1	LGBMRegressor1	1	0.8026
2	LGBMRegressor5	5	0.8141
3	LGBMRegressor10	10	0.8568

Com podem observar cap model millora les prediccions basades en la teoria del *Random Walk* que es fonamentava en prediccions pràcticament aleatòries, fet que ens porta a assumir que aquests tipus de models no serveixen per donar solució al nostre problema. Consegüentment, i tenint també en compte l'anàlisi de l'autocorrelació de la variable realitzada, refutem la nostra hipòtesi inicial:

> *REFUTAT: El preu del Bitcoin presenta una autocorrelació apta pels models de regressió, per tant, els models de regressió són aptes per donar solució al nostre problema.*

3. Estudi sobre la idoneïtat de models de classificació

Un altre enfocament que explorarem serà deixar de banda les prediccions directes sobre el preu, per realitzar prediccions sobre el signe del mercat, és a dir, tractarem d'endevinar si el preu pujarà o baixarà. D'aquesta forma reduïm la complexitat de les prediccions.

Per dur a terme aquest enfocament treballarem en la creació d'un model de classificació. Aquest tipus de models necessiten ser entrenats amb un conjunt de dades etiquetades. Això vol dir que cada mostra té associada una resposta. A partir d'aquest conjunt de dades d'entrenament, el model aprèn a detectar patrons i característiques que associen les mostres amb les seves respostes.

Pel que sembla, aquests tipus de models encaixen molt bé amb el problema. Si definim tres etiquetes [-1, 0, 1] podríem crear un model que ens doni com a resultat un senyal amb valors del conjunt C = [-1, 0, 1] i recordem que aquest tipus de senyals són els que han de calcular els robots finals del treball.

A més, que el resultat del model sigui un directament senyal de compravenda ens permet calcular el rendiment financer que ens oferiria. Tenint tot això en compte, donem cert crèdit previ al model i formulem la següent hipòtesi:

> *Un model de classificació aplicat al nostre problema pot oferir un percentatge d'encerts major al 50%, fet que el faria apte com a solució pels bots finals, ja que a la llarga ens faria guanyar diners.*

Comencem doncs a crear un model. El primer que hem de saber és que cal dotar al model de sensibilitat a certs aspectes del mercat. Això ho aconseguirem afegint indicadors tècnics com a predictors del model.

Els predictors són dades afegides al conjunt d'entrenament que ajuden a explicar el comportament del fenomen a modelar. És important seleccionar els predictors adequats, ja que poden tenir un gran impacte en la precisió del model. Una selecció inadequada de predictors pot portar a un model amb un rendiment baix. El que hem d'intentar és trobar els predictors que millor expliquin el fenomen a modelar i tinguin un major impacte en la predicció.

Per a un model de classificació com el nostre hem considerat afegir aquests indicadors tècnics com a principals predictors que complementin les dades OCHL:

- **Mitjana mòbil simple (SMA)**: és un indicador tècnic que s'utilitza per suavitzar les fluctuacions del preu d'un actiu i per identificar tendències a llarg termini.

Fórmula:
SMA (i) = (Preu (i) + Preu (i-1) + ... + Preu (i-n)) / n
On:
SMA (i) = Valor de la mitjana mòbil simple en el període i. Preu (i) = Preu de l'actiu en el període i. n = Nombre de períodes en el càlcul de la mitjana mòbil simple.

- **Índex de força relativa (RSI)**: és un indicador tècnic que mesura la força o debilitat del preu d'un actiu.

Fórmula:
RSI = 100 - (100 / (1 + RS)) RS = Mitjana dels guanys / Mitjana de les pèrdues
On:
RSI = Valor de l'índex de força relativa en el període i RS = Relació de força Mitjana dels guanys = Mitjana de les variacions positives del preu Mitjana de les pèrdues = Mitjana de les variacions negatives del preu

- **Stop and Reverse (SAR)**: és un indicador tècnic que s'utilitza per identificar tendències en els preus dels actius i per determinar punts d'entrada i sortida del mercat.

Fórmula:
SAR (i) = SAR (i-1) + AF (i-1) * (EP (i-1) - SAR (i-1))
On:
SAR (i) = Valor del SAR en el període i SAR (i-1) = Valor del SAR en el període anterior EP (i-1) = Punt més alt en una tendència alcista o punt més baix en una tendència baixista

- **ADX:** és un indicador tècnic que mesura la força de la tendència d'un actiu.

Fórmula:
ADX = (DX / n) * 100 DX = ((+DI) - (-DI)) / (+DI) + (-DI)
On:
ADX = Valor de l'índex de direcció mitjana en el període i DX = Indicador de força de tendència +DI = Indicador de tendència alcista -DI = Indicador de tendència baixista n = Nombre de períodes en el càlcul de l'ADX

- **Convergència i divergència de mitjanes mòbils (MACD)**: és un indicador tècnic que mesura la relació entre dues mitjanes mòbils d'un actiu i indica possibles canvis en la tendència.

Fórmula:
MACD = SMA (x) - SMA (y) Línia de seguiment = SMA (z) de MACD x > y
On:
MACD = Valor de la mitjana mòbil convergència/divergència en el període i SMA (x) = Mitjana mòbil simple de x períodes del preu de l'actiu SMA (y) = Mitjana mòbil simple de y períodes del preu de l'actiu Línia de seguiment = Valor de la línia de seguiment en el període i SMA (z) = Mitjana mòbil simple de z períodes del MACD

- **Desviació Típica (Standard Deviation, STDEV)** mesura la dispersió d'un conjunt de dades en relació amb la seva mitjana. És una mesura de la variabilitat d'un conjunt de dades.

Fórmula:
STDEV = sqrt (suma (xi - xbar)^2 / n)
On:
STDEV = Valor de la desviació típica xi = Valor individual en el conjunt de dades xbar = Mitjana del conjunt de dades n = Nombre de dades en el conjunt

Observem a continuació la selecció de predictors realitzada per preparar l'entrenament del model, integrant-los amb les dades base OCHL:

```
n=12
df['SMA'] = df['Close'].shift(1).rolling(window=n).mean()
df['RSI'] = ta.RSI(np.array(df['Close'].shift(1)),timeperiod= n)
df['MACD'], df['MACD_SIGNAL'], df['MACD_HIST'] = ta.MACD(np.array(df['Close'].shift(1)), n, 2*n)
df['STDDEV'] = ta.STDDEV(np.array(df['Close'].shift(1)), timeperiod=n, nbdev=1)
df['SAR'] = ta.SAR(np.array(df['High'].shift(1)), np.array(df['Low'].shift(1)), 0.2, 0.2)
df['ADX'] = ta.ADX(np.array(df['High'].shift(1)), np.array(df['Low'].shift(1)), np.array(df['Open']), timeperiod=n)
```

```
# Donem informacio clau del periode Tn-1 (abans del actual) al model
df['Prev_High'] = df['High'].shift(1)
df['Prev_Volume'] = df['Volume'].shift(1)
df['Prev_Low'] = df['Low'].shift(1)
df['Prev_Close'] = df['Close'].shift(1)
df['Prev_Open'] = df['Open'].shift(1)
df['OO']= df['Prev_Open'] - df['Open']
df['OC']= df['Prev_Open'] - df['Prev_Close'].shift(1)
```

```
# Generem els els n+1 ultims retorns
df['Ret'] = (df['Close']-df['Prev_Close'])/df['Close']
for i in range(1, n):
    df['Prev_Volume%i' % i] = df['Prev_Volume'].shift(i)
    df['Prev_Close%i' % i] = df['Prev_Close'].shift(i)
    df['Ret%i' % i] = df['Ret'].shift(i)
```

Una vegada afegits els predictors desitjats tenim el conjunt d'entrenament preparat i només cal realitzar l'etiquetació. El procés d'etiquetació consisteix a proporcionar al model l'exemple de quina seria la resposta correcta per cada mostra. En el nostre cas, l'etiquetació es força senzilla gràcies a la columna Ret calculada, que ens diu quant ha variat el preu des de la mostra anterior.

- Assignarem un 0 per defecte a totes les mostres com a etiquetació inicial.
- Després assignarem un 1 a un terç de les mostres, les de majors retorns positius. Així, quan el preu puja significativament l'etiqueta indica una posició a llarg de compra.
- Per últim, assignarem un -1 a un terç de les mostres, les de majors retorns negatius. Així, quan el preu baixa significativament l'etiqueta indica una posició a curt de venda.

```
split = int(0.8*len(df))
# Generem una senyal de compravenda ideal que el model haura de imitar
df['Signal'] = 0
# Pel terç dels majors retorns positius la posicio correcta es "hem comprat"
df.loc[df['Ret'] > df['Ret'].quantile(0.67), 'Signal'] = 1
# Pel terç dels majors retorns positius la posicio correcta es "hem venut"
df.loc[df['Ret'] < df['Ret'].quantile(0.33), 'Signal'] = -1
```

Una vegada tenim enllestida l'etiquetació procedim a l'entrenament del model i l'anàlisi dels resultats.

```
model = SVC(C=1000, gamma='auto', kernel='rbf', probability=True)
```

```
ss = StandardScaler()
model.fit(ss.fit_transform(X.iloc[:split]), y.iloc[:split])
```

```
SVC(C=1000, gamma='auto', probability=True)
```

```
predictions = model.predict(ss.transform(X.iloc[split:]))
predictions
```

```
array([ 1, -1,  1,  0, -1,  0,  0,  1,  1, -1, -1, -1, -1, -1,  1, -1, -1,
        1, -1, -1,  1,  0,  1,  1,  0,  1,  1,  0,  0,  0,  0, -1,  0,  1,
        1,  1,  1,  1,  1,  0,  1, -1,  1, -1, -1, -1, -1, -1, -1, -1, -1,
        1,  1, -1,  1, -1, -1, -1,  1,  0,  1, -1,  0, -1,  1, -1,  0, -1,
        1, -1, -1,  0,  0,  0,  1,  1,  0,  0,  0,  0,  1,  1,  0, -1, -1,
        0,  0,  0,  0,  1, -1,  0,  1,  0,  0,  0,  0,  0,  1,  1,  0,  0,
        0,  0, -1,  0,  0,  0,  0, -1,  0, -1,  0, -1,  0, -1, -1,  0, -1,
        1,  1,  0,  1, -1, -1, -1,  0, -1,  1, -1, -1, -1, -1, -1, -1, -1,
        1,  1,  1,  1, -1, -1, -1,  1,  1,  1,  1,  1,  0,  0, -1, -1,  0,
       -1,  1,  0, -1, -1,  0,  1, -1, -1, -1,  1,  1, -1, -1,  1,  0,  0,
        0,  1,  0,  0,  0,  0, -1, -1, -1,  1, -1,  1, -1,  0, -1, -1,  1,
       -1, -1,  0,  0,  1,  1,  0,  0,  1, -1,  0, -1,  0], dtype=int64)
```

```
# Estudi sobre la precisió del model
cr = classification_report(y[split:], predictions)
print(cr)
```

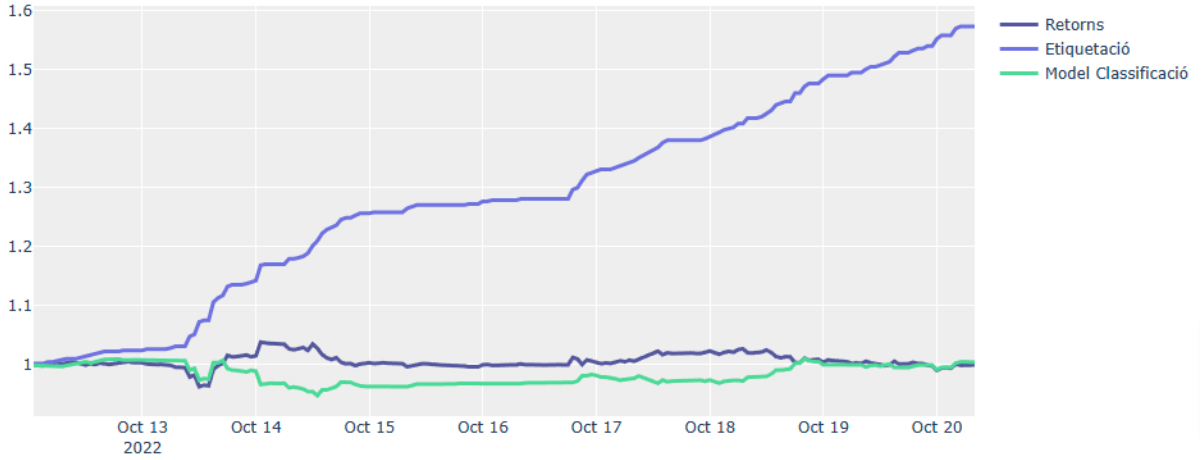
```
              precision    recall  f1-score   support

          -1       0.30      0.43      0.35        54
           0       0.60      0.41      0.49        95
           1       0.34      0.39      0.36        51

    accuracy                           0.41       200
   macro avg       0.41      0.41      0.40       200
weighted avg       0.45      0.41      0.42       200
```

La precisió general reportada del 41% suggereix que el model està classificant incorrectament el 59% de les entrades. Això significa que el model no està sent capaç de categoritzar de manera adequada les noves mostres.

Per últim, prendrem els resultats del model per simular els guanys i les pèrdues que produiria seguir les prediccions del model. Farem una simulació en la qual la inversió inicial és 1$ i assumim la reinversió continua dels diners generats per la compravenda.



- En blau fosc observem els **retorns** del preu. Ens mostra com varia el preu en el temps.
- En blau clar els guanys produïts per l'**etiquetació**. De la gràfica podem extreure que ens produiria entre un 50% i un 60% de beneficis respecte a la inversió inicial, en poc més d'una setmana.
- En verd clar els guanys produïts per les prediccions del **model.** Com podem observar la línia dels guanys produïts no creix en el temps i els beneficis generats són aproximadament del 0%.

Si acceptéssim el model de classificació com a solució pel nostre problema les compres i les vendes realitzades no ens donarien beneficis. Ara bé, independentment del baix rendiment financer observat, la precisió general que presenta el model està per sota del 50%, per tant, refutem la hipòtesi inicial sobre aquest tipus de models:

> *REFUTAT: Un model de classificació aplicat al nostre problema pot oferir un percentatge d'encerts major al 50%, fet que el faria apte com a solució pels bots finals.*

4. Estudi sobre la idoneïtat dels indicadors tècnics

En aquest punt del treball considerem comprovada la no idoneïtat dels sistemes basats en models de regressió i classificació com a solucions pel trading de criptomonedes. Ara l'enfocament recaurà sobre els indicadors tècnics, que són càlculs estadístics que ens donen informació clau del mercat. Ja vam utilitzar alguns com a predictors pels models de classificació.

Els indicadors tècnics analitzen les dades històriques del preu d'un actiu per ajudar a determinar aspectes com la tendència, moment i volatilitat actuals del mercat. Són una excel·lent opció com a base per crear estratègies de trading perquè permeten identificar oportunitats de compra o venda en diferents marcs temporals. A més, es poden combinar diferents indicadors per a crear estratègies de trading més completes i eficaces.

Per testejar el seu potencial, generarem estratègies amb la combinació de diferents indicadors. Les estratègies es creen fent una bona lectura de la informació que ens donen els indicadors. Quan de la lectura entenem que estem en un moment amb condicions de mercat favorables per una compra, l'estratègia generarà un senyal de compra (1). Si entenem que estem en un moment amb condicions favorables per una venda, l'estratègia generarà un senyal de venda (-1). Altrament, si els indicadors no ens fan entendre que les condicions del mercat són favorables per cap acció de compravenda, l'estratègia generarà un senyal de neutralitat (0).

Per cada estratègia creada simularem la seva execució amb dades històriques del preu de la criptomoneda. Just com vam fer amb el senyal del model de classificació. Aquest tipus de proves s'anomenen backtesting. El backtesting consisteix, per tant, a simular amb dades històriques del preu les accions de compravenda que planteja l'estratègia i prendre mesures del rendiment financer que hagués oferit en el passat.

4.1. Implementació de la classe “tester”

Per realitzar el backtesting d'estratègies hem creat un framework de treball, és a dir, un sistema que agilitzarà el procés de testatge i optimització de les estratègies. Hem materialitzat aquest sistema escrivint una classe que permet l'execució d'una estratègia de trading per un conjunt de dades històriques i la mesura de mètriques de rendiment financer per avaluar-ne els resultats. Aquesta classe inclou les funcions següents:

- **__init__**: és el constructor de la classe, on es defineixen atributs generals com la ruta al fitxer CSV (filepath) amb l’històric de preus, els costos del trading o comissions (tc) i un atribut buit on es guardaran els resultats de les proves posteriorment (results).

```
def __init__(self, filepath, tc):

    self.filepath = filepath
    self.tc = tc
    self.results = None
    self.get_data()
```

- **get_data**: serveix per carregar les dades històriques de preus de la criptomoneda. A més, s'aprofita per fer un càlcul dels retorns del preu. En aquest cas hem fet servir retorns logarítmics, és a dir retorns calculats en proporció a la variació percentual del preu. Això els fa més útils per comparar rendiments a llarg termini, ja que tenen en compte el creixement acumulat del preu i no distorsionen la comparació de rendiments a causa grans variacions de preu.

```
def get_data(self):
    ''' Importa les dades.
    '''
    data = pd.read_csv(self.filepath, parse_dates = ["Date"], index_col = "Date")
    data['returns'] = np.log(data.Close / data.Close.shift(1))
    self.data = data
```

- **test_strategy**: podríem dir que és la funció orquestradora de la classe que pren els paràmetres d'entrada: dates d'inici i final del test i els inputs de l'estratègia. S'encarrega de cridar a les funcions que apliquen l'estratègia, calculen el rendiment financer i imprimeixen un informe dels resultats.

```
def test_strategy(self, start, end, inputs):
    '''
    Prepara les dades i testeja la estrategia. Inclou informe dels resultats.

    Parametres
    ============
    start: str
        es la data inicial de les dades que s'utilitzaran per el testing.

    end: str
        es la data inicial de les dades que s'utilitzaran per el testing.

    inputs: tuple
        son els valors d'entrada de la estrategia.

    '''

    self.prepare_data(start, end, inputs)
    self.run_backtest()

    data = self.results.copy()
    data["creturns"] = data["returns"].cumsum().apply(np.exp)
    data["cstrategy"] = data["strategy"].cumsum().apply(np.exp)
    self.results = data

    self.print_performance()
```

- **prepare_data**: té la responsabilitat de fer el càlcul dels indicadors que fa servir l'estratègia i generar els senyals de compravenda. El codi d'aquesta funció és específic de cada estratègia i, per tant, serà una de les funcions que haurem d'adaptar per cada estratègia.

```
def prepare_data(self, start, end, inputs):
    ''' Prepara les dades pel testing.
    '''

    data = self.data.loc[start:end, ["Close", 'returns']].copy()
    self.tp_year = (data.Close.count() / ((data.index[-1] - data.index[0]).days / 365.25))
    self.inputs = inputs

    ########################## Strategy-Specific ############################

    sma_s_length=int(inputs[0])
    sma_m_length=int(inputs[0]*inputs[1])
    sma_l_length=int(inputs[0]*inputs[1]*inputs[2])

    # Claculant les tres mitjes movils
    data["sma_s"] = data['Close'].rolling(sma_s_length).mean()
    data["sma_m"] = data['Close'].rolling(sma_m_length).mean()
    data["sma_l"] = data['Close'].rolling(sma_l_length).mean()

    # Definint les condicions de compravenda
    cond_long = (data['sma_s'] > data['sma_m']) & (data['sma_m'] > data['sma_l'])
    cond_short = (data['sma_s'] < data['sma_m']) & (data['sma_m'] < data['sma_l'])

    # Generació del senyal de l'estratègia
    data["position"] = 0
    data.loc[cond_long, "position"] = 1
    data.loc[cond_short, "position"] = -1

    #########################################################################

    data.dropna(inplace = True)
    self.results = data
```

- **run_backtest**: té la responsabilitat de calcular els guanys i les pèrdues que produiria l'estratègia si fos executada pel conjunt de dades seleccionat. En el càlcul es tenen en compte els costos addicionals derivats del trading com les comissions que cobren les plataformes de trading per cada operació o l'anomenat Bid-Ask Spread, és a dir, la diferència mitjana entre el preu al qual vols fer una operació i el millor preu que et pot oferir la plataforma en un moment concret. És important tenir en compte aquests costos addicionals per tenir una mesura acurada del veritable rendiment de l'estratègia. A vegades, una estratègia massa proactiva, que fa moltes operacions en poc temps, es menja els mateixos beneficis que genera en els costos operacionals.

```
def run_backtest(self):
    ''' Testeja la estrategia.
    '''

    data = self.results.copy()
    data["strategy"] = data["position"].shift(1) * data["returns"]
    data["trades"] = data.position.diff().fillna(0).abs()
    data.strategy = data.strategy + data.trades * self.tc

    self.results = data
```

- **optimize_strategy**: és una funció que realitza una optimització de l'estratègia cercant els millors paràmetres d'entrada per aquesta. Pren com a paràmetres les dates d'inici i final per les quals optimitzar l'estratègia, la mètrica de rendiment financer que es vol optimitzar i rangs de valors que s'utilitzaran per a la cerca. Un rang per cada input de l'estratègia a optimitzar. El que fa és generar totes les combinacions possibles de valors amb els rangs donats i cridar a les funcions de backtesting per cada combinació. Guarda el rendiment associat a cada combinació i al final es queda amb la combinació que millor rendiment ha donat pel període de temps seleccionat.

```
def optimize_strategy(self, ranges, start, end,  metric = "Multiple"):
    '''
    Testeja la estrategia per diferents parametres d'entrada. Inclou optimització e informe de resultats.

    Parametres
    ============
    ranges: tuple
        son els rangs de valors d'entrada a combinar per la optimització.

    start: str
        es la data inicial de les dades que s'utilitzaran per l'optimització.

    end: str
        es la data final de les dades que s'utilitzaran per l'optimització.

    metric: str
        la metrica de rendiment que volem optimitzar: "Multiple" o "Sharpe".
    '''

    self.metric = metric

    if metric == "Multiple":
        performance_function = self.calculate_multiple
    elif metric == "Sharpe":
        performance_function = self.calculate_sharpe

    ########################## Strategy-Specific #############################

    SMA_S_range = np.arange(*ranges[0])
    SMA_Mmul_range = np.arange(*ranges[1])
    SMA_Lmul_range = np.arange(*ranges[2])

    combinations = list(product(SMA_S_range, SMA_Mmul_range, SMA_Lmul_range))

    performance = []
    for comb in combinations:
        self.prepare_data(inputs = comb, start = start, end = end)
        self.run_backtest()
        performance.append(performance_function(self.results.strategy))

    self.results_overview =  pd.DataFrame(data = np.array(combinations),
                                          columns = ["SMA_S", "SMA_Mmul", "SMA_Lmul"])
    self.results_overview["performance"] = performance

    ##########################################################################

    self.find_best_strategy()
    self.test_strategy(inputs = self.best_inputs, start = start, end = end)
```

- **find_best_strategy**: té la responsabilitat de trobar la combinació guanyadora derivada de l'optimització realitzada pel mètode **optimize_strategy**. A més, imprimeix els inputs que formen la combinació guanyadora i el rendiment que ofereixen per donar feedback del resultat de l'optimització.

```
def find_best_strategy(self):
    ''' Troba la estrategia optima.
    '''

    best = self.results_overview.nlargest(1, "performance")
    SMA_S = best.SMA_S.iloc[0]
    SMA_M = int(SMA_S*best.SMA_Mmul.iloc[0])
    SMA_L = int(SMA_S*best.SMA_Mmul.iloc[0]*best.SMA_Lmul.iloc[0])
    perf = best.performance.iloc[0]
    print("SMA_S: {} | SMA_M: {} | SMA_L: {} | {}: {}".format(SMA_S, SMA_M, SMA_L, self.metric, round(perf, 4)))
    self.best_inputs = (SMA_S, best.SMA_Mmul.iloc[0], best.SMA_Lmul.iloc[0])
```

- **calculate_multiple**: donada la sèrie de guanys i pèrdues que ofereix una estratègia, calcula el seu múltiple d'inversió, una mètrica que indica per quina quantitat multiplica l'estratègia la inversió inicial. És una mètrica de la recompensa oferida per l'estratègia.

```
def calculate_multiple(self, series):
    return np.exp(series.sum())
```

- **calculate_cagr**: donada la sèrie de guanys i pèrdues que ofereix una estratègia, calcula quin seria el creixement anual de la inversió inicial, assumint la reinversió continua dels beneficis generats per l'estratègia. És una mètrica de la recompensa oferida per l'estratègia.

```
def calculate_cagr(self, series):
    return np.exp(series.sum())**(1/((series.index[-1] - series.index[0]).days / 365.25)) - 1
```

- **calculate_annualized_mean**: donada la sèrie de guanys i pèrdues que ofereix una estratègia, calcula els beneficis o pèrdues mitjans al cap d'un any. És una mètrica de la recompensa oferida per l'estratègia.

```
def calculate_annualized_mean(self, series):
    return series.mean() * self.tp_year
```

- **calculate_annualized_std**: donada la sèrie de guanys i pèrdues que ofereix una estratègia, calcula la variació mitjana al cap d'un any. Com més variabilitat es considera que l'estratègia assumeix més risc en les seves operacions. És una mètrica del risc assumit per l'estratègia.

```
def calculate_annualized_std(self, series):
    return series.std() * np.sqrt(self.tp_year)
```

- **calculate_sharpe**: donada la sèrie de guanys i pèrdues que ofereix una estratègia, calcula la relació entre els beneficis acumulats anual i la variabilitat mitjana al cap d'un any. És una mètrica que mesura la relació recompensa/risc oferida per l'estratègia.

```
def calculate_sharpe(self, series):
    if series.std() == 0:
        return np.nan
    else:
        return self.calculate_cagr(series) / self.calculate_annualized_std(series)
```

- **print_performance**: genera un informe del rendiment financer de l'estratègia després de realitzar la simulació. El rendiment és comparat amb el d'una estratègia passiva de Buy & Hold. Aquesta estratègia consisteix a comprar al començament i vendre al final. La comparació ens permet determinar si la proactivitat de la nostra estratègia té un impacte positiu sobre el rendiment, millorant el que ens ofereix només mantenir l'actiu a llarg termini de forma passiva.

```
def print_performance(self):
    ''' Calcula i mostra varies metriques de rendiment.
    '''

    data = self.results.copy()
    strategy_multiple = round(self.calculate_multiple(data.strategy), 6)
    bh_multiple =       round(self.calculate_multiple(data.returns), 6)
    outperf =           round(strategy_multiple - bh_multiple, 6)
    cagr =              round(self.calculate_cagr(data.strategy), 6)
    ann_mean =          round(self.calculate_annualized_mean(data.strategy), 6)
    ann_std =           round(self.calculate_annualized_std(data.strategy), 6)
    sharpe =            round(self.calculate_sharpe(data.strategy), 6)

    self.strategy_multiple=strategy_multiple
    self.outperf=outperf

    print(100 * "=")
    print("SIMPLE PRICE & VOLUME STRATEGY | Inputs = {}".format(self.inputs))
    print(100 * "-")
    print("PERFORMANCE MEASURES:")
    print("\n")
    print("Multiple (Strategy):         {}".format(strategy_multiple))
    print("Multiple (Buy-and-Hold):     {}".format(bh_multiple))
    print(38 * "-")
    print("Out-/Underperformance:       {}".format(outperf))
    print("\n")
    print("CAGR:                        {}".format(cagr))
    print("Annualized Mean:             {}".format(ann_mean))
    print("Annualized Std:              {}".format(ann_std))
    print("Sharpe Ratio:                {}".format(sharpe))

    print(100 * "=")
```

- **plot_results**: crea gràfics per visualitzar el benefici que generaria l'estratègia durant el temps comparat amb el benefici que generaria la Buy & Hold.

```
def plot_results(self):
    ''' Mostra el rendiment acumulat de la estrategia comparada amb una estrategia passiva de Buy & Hold.
    '''
    if self.results is None:
        print("Run test_strategy() first.")
    else:
        title = "Estrategia vs Buy & Hold | TC = {}".format(self.tc)
        self.results[["creturns", "cstrategy"]].plot(title=title, figsize=(16, 8))
```

Amb aquesta explicació coneixem millor el framework creat per fer el backtesting d'estratègies de forma més àgil. Una vegada creat només cal afegir algunes línies de codi addicionals per adaptar la classe a cada estratègia i realitzar les proves que considerem necessàries.

Ara parlarem de les proves realitzades per diverses estratègies que són molt conegudes de les quals esperem que donin millors resultats que els models de IA. Per cada estratègia:

1. Calcularem els seus indicadors i els explorarem visualment.
2. Definirem les regles per generar un senyal de trading.
3. Adaptarem la classe “tester” a l'estratègia.
4. Testejarem l'estratègia utilitzant la classe “tester”.
5. Optimitzarem l’estratègia utilitzant la classe “tester”.
6. Testejarem el rendiment mensual de l’estratègia en el 2022.

4.2. Estratègia basada en triple SMA

La primera estratègia de la qual parlarem utilitza un sol indicador anomenat SMA o Simple Moving Average. És el que anomenem una mitjana mòbil simple que suavitza les fluctuacions del preu dibuixant-nos la tendència general del preu.Per aquesta estratègia utilitzarem tres mitjanes mòbils simples amb diferents n:

- **Mitjana mòbil curta**: una SMA amb un període de temps curt (amb una n petita) que és utilitzada per detectar les tendències a curt termini.
- **Mitjana mòbil mediana**: una SMA amb un període de temps mitjà (amb una n mediana) que és utilitzada per detectar les tendències a mitjà termini.
- **Mitjana mòbil llarga**: una SMA amb un període de temps llarg (amb una n gran) que és utilitzada per detectar les tendències a llarg termini.

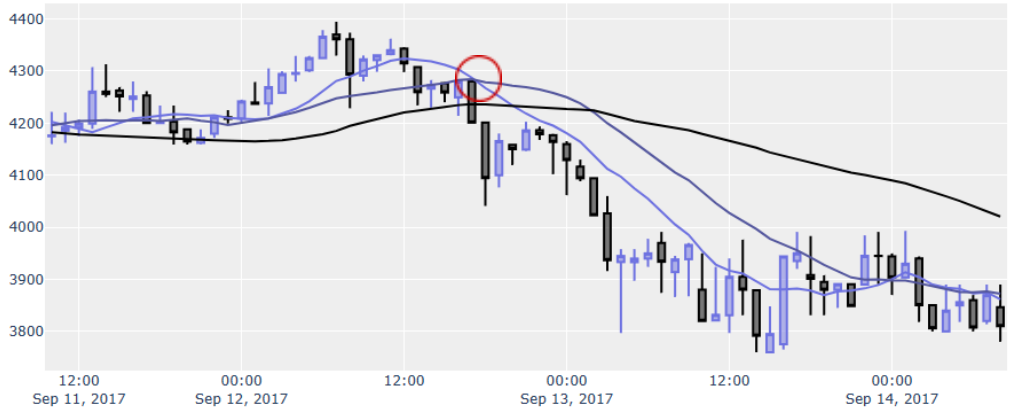
La idea principal d'aquesta estratègia és que, si la SMA curta creua per sobre a la SMA mitjana, es considera un senyal de compra, mentre que si la SMA curta creua per sota a la SMA mitjana, es considera un senyal de venda.

La SMA a llarg termini ens servirà per filtrar els senyals de compravenda. Només farem compres quan la SMA curta està per sobre d'aquesta i només farem vendes quan està per sota.

4.2.1. Càlcul i exploració visual dels indicadors SMA

```
SMA_S = 10
SMA_M = 20
SMA_L = 50 # Definicio de les finestres de cada mitja
```

```
data["sma_s"] = data.Close.rolling(SMA_S).mean()
data["sma_m"] = data.Close.rolling(SMA_M).mean()
data["sma_l"] = data.Close.rolling(SMA_L).mean() # Claculant les mitjes tres mitjes movils
```



Observem en el gràfic com les mitjanes mòbils ens ajuden a identificar el començament un moviment baixista quan la mitjana curta (blau clar) creua per sota la mitjana intermèdia (blau fosc). Aquí l'estratègia ens generaria un senyal de venda i guanyaríem diners.

4.2.2. Backtesting de l'estratègia basada en triple SMA

Per realitzar les proves de backtesting cal adaptar la classe "tester" per l'estratègia actual. Mostrem a continuació la funció **prepare_data** que conté les principals modificacions de la classe, el càlcul dels indicadors i la generació del senyal de compravenda:

```
def prepare_data(self, start, end, inputs):
    ''' Prepara les dades pel testing.
    '''

    data = self.data.loc[start:end, ["Close", 'returns']].copy()
    self.tp_year = (data.Close.count() / ((data.index[-1] - data.index[0]).days / 365.25))
    self.inputs = inputs

    ########################## Strategy-Specific ##############################

    sma_s_length=int(inputs[0])
    sma_m_length=int(inputs[0]*inputs[1])
    sma_l_length=int(inputs[0]*inputs[1]*inputs[2])

    # Claculant les tres mitjes movils
    data["sma_s"] = data['Close'].rolling(sma_s_length).mean()
    data["sma_m"] = data['Close'].rolling(sma_m_length).mean()
    data["sma_l"] = data['Close'].rolling(sma_l_length).mean()

    # Definint les condicions de compravenda
    cond_long = (data['sma_s'] > data['sma_m']) & (data['sma_m'] > data['sma_l'])
    cond_short = (data['sma_s'] < data['sma_m']) & (data['sma_m'] < data['sma_l'])

    # Generació del senyal de l'estratègia
    data["position"] = 0
    data.loc[cond_long, "position"] = 1
    data.loc[cond_short, "position"] = -1

    ###########################################################################
```

Després de realitzar les adaptacions a la classe procedim a l'execució de la funció **test_strategy** passant com a paràmetres els inputs de l'estratègia. Després utilitzem la funció **plot_results** per tenir un gràfic dels retorns de l'estratègia. Observem a continuació un primer output que ens informa del rendiment que ha oferit l'estratègia així com una exploració gràfica dels guanys i les pèrdues que produeix amb els inputs donats i pel període de temps seleccionat:

```
filepath = "../csv/BTCUSDT_15m.csv"
tc = -0.00015
```

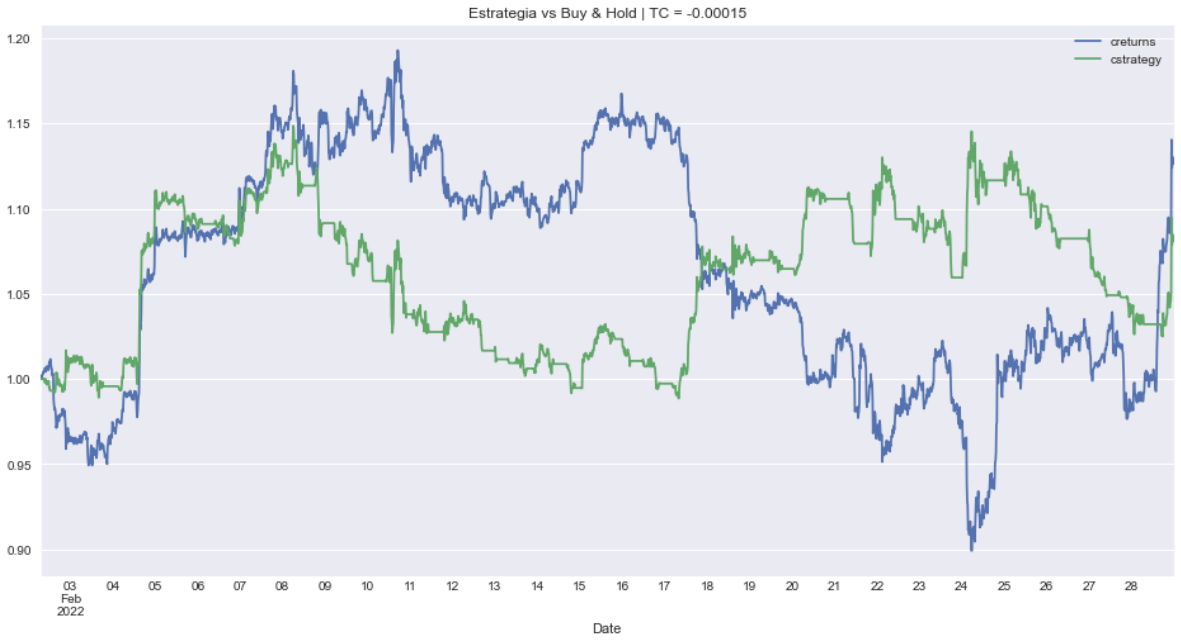
```
tester = Tester(filepath, tc)
```

```
tester.test_strategy(inputs = (20, 2.5, 2.5), start = '2022-02', end = '2022-02')
tester.plot_results()
```

```
================================================================================================================
SIMPLE PRICE & VOLUME STRATEGY | Inputs = (20, 2.5, 2.5)
----------------------------------------------------------------------------------------------------------------
PERFORMANCE MEASURES:


Multiple (Strategy):         1.082182
Multiple (Buy-and-Hold):     1.127555
-------------------------------------
Out-/Underperformance:       -0.045373


CAGR:                        2.032867
Annualized Mean:             1.120523
Annualized Std:              0.577622
Sharpe Ratio:                3.519373
================================================================================================================
```



4.2.3. Optimització de l'estratègia basada en triple SMA

La primera prova de backtesting ens mostra que l'estratègia podria arribar a ser bona, però encara no ens dona el rendiment desitjat. Procedim a l'optimització dels inputs per mirar de millorar el seu rendiment. Per aquesta tasca definim rangs de valors per cada input i els passem com a paràmetres per la funció **optimize_strategy:**

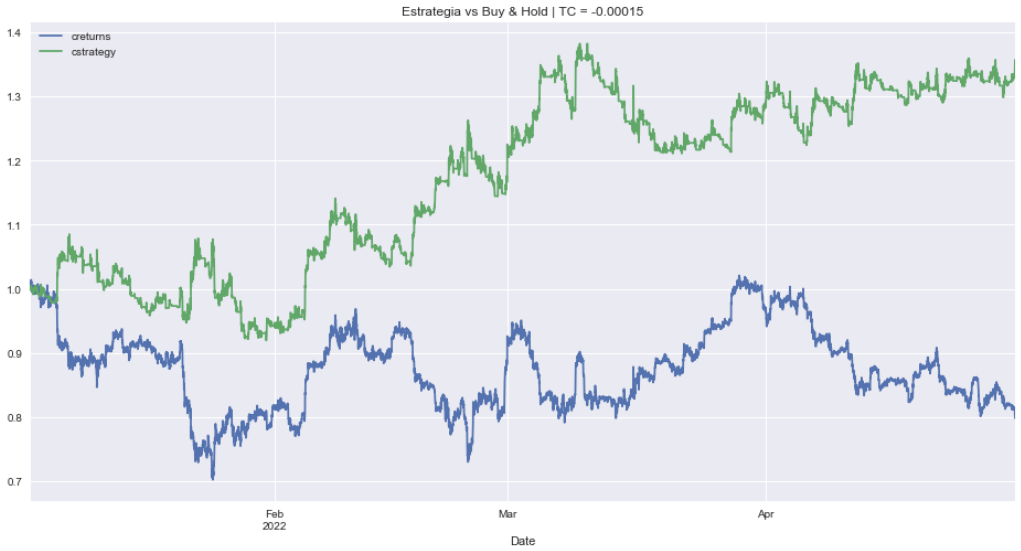
```
s_range = (5, 120, 5)
mul_m = (1.5, 5, 0.5)
mul_l = (1.5, 5, 0.5)
```

```
tester.optimize_strategy(ranges = (s_range, mul_m, mul_l), start = '2022-01', end = '2022-04')
tester.plot_results()
```

```
SMA_S: 20.0 | SMA_M: 60 | SMA_L: 150 | Multiple: 1.3571
=============================================================================================================
SIMPLE PRICE & VOLUME STRATEGY | Inputs = (20.0, 3.0, 2.5)
-------------------------------------------------------------------------------------------------------------
PERFORMANCE MEASURES:


Multiple (Strategy):         1.357061
Multiple (Buy-and-Hold):     0.798437
--------------------------------------
Out-/Underperformance:       0.558624


CAGR:                        1.573002
Annualized Mean:             0.949495
Annualized Std:              0.508034
Sharpe Ratio:                3.096253
=============================================================================================================
```



Com podem observar l'estratègia millora molt amb els paràmetres d'entrada optimitzats per un conjunt de dades específics. Mostrant un múltiple d'inversió de 1.35 i un Sharpe de 3.09. Tot i així els paràmetres d'entrada ara estan optimitzats per un conjunt de dades concret, el que vol dir que hem fet que l'estratègia s'adapti perfectament als canvis de preu donats en aquest període de temps. Ara cal que testejar com funcionaria l'estratègia optimitzada per un conjunt de dades diferent.

4.2.4. Rendiment mensual mitjà de l'estratègia basada en triple SMA

En aquest punt només ens queda comprovar si l'estratègia optimitzada funciona correctament per un conjunt de dades diferent de l'utilitzat per l'optimització: Fem proves mensuals per veure com es comporta en l'escenari plantejat:

```
meses = ['2022-05', '2022-06', '2022-07', '2022-08', '2022-09', '2022-10']
strategy_multiples = []
outperfs = []

for mes in meses:
    tester.test_strategy(inputs = (20, 4, 2.5), start=mes, end=mes)
    strategy_multiples.append(tester.strategy_multiple)
    outperfs.append(tester.outperf)

print(100 * "=")
print("Average investment multiple: {}".format(pd.DataFrame(strategy_multiples).mean()))
print("Average outperformance versus (Buy & Hold): {}".format(pd.DataFrame(outperfs).mean()))
print(100 * "=")
```

```
====================================================================================================
Average investment multiple: 0   1.0146
dtype: float64
Average outperformance versus (Buy & Hold): 0   0.0921
dtype: float64
====================================================================================================
```

Les proves realitzades mostren que l'estratègia podria generar un 1.5% aproximat de beneficis mensuals, un 18% anual. En termes econòmics, planteja un rendiment superior al que ofereix un fons d'inversió borsari que sol oferir un 8-12% anual.

4.3. Estratègia basada en RSI i Bandes de Bollinger

Aquesta estratègia combina dos indicadors tècnics per identificar les oportunitats de compra o venda.

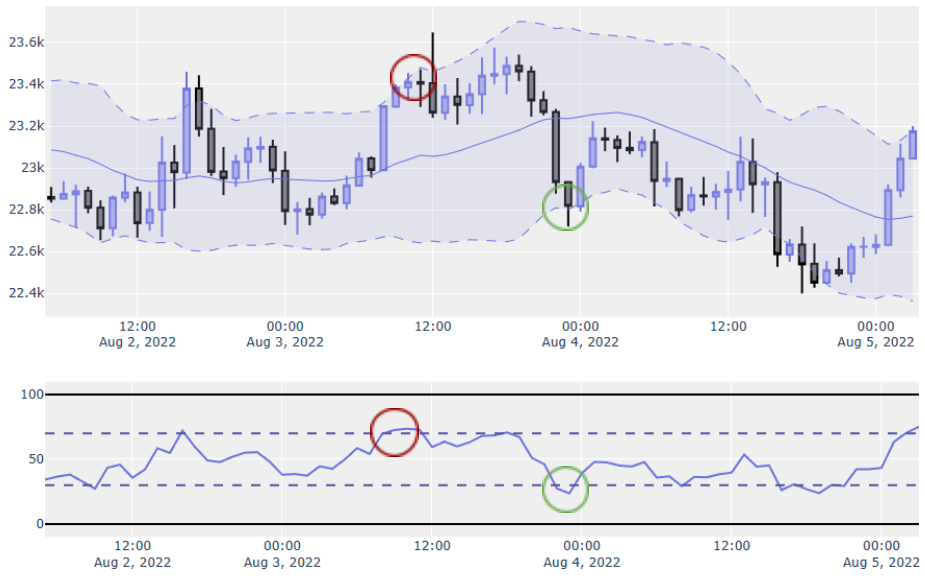
- **RSI**: és un indicador que mesura el moment dels moviments del mercat posant el preu actual de l'actiu amb relació als preus que ha tingut en un període de temps determinat. Si el preu és molt alt respecte als darrers preus diem que està sobrecomprat. Si, per contra, el preu és baix comparat amb els darrers preus de la sèrie, diem que està sobrevenut.
- **Bandes de Bollinger**: un indicador que mostra la volatilitat de l'actiu. Està compost de tres bandes: una SMA que representa el preu mitjà, i dues bandes per sobre i per sota de la SMA que es distancien d'aquesta en proporció a la volatilitat present. Quan les bandes inferior i superior s'allunyen de la SMA vol dir que la volatilitat està creixent. Si, per contra, les bandes s'apropen a la SMA vol dir que la volatilitat està decreixent.

L'estratègia consisteix a combinar RSI i Bollinger Bands per determinar quan és el millor moment per a comprar o vendre un actiu. Per exemple, si el RSI indica que un actiu està sobrecomprat i les Bandes de Bollinger mostren que el preu està per sobre de la banda superior, és possible que sigui un bon moment per vendre. D'altra banda, si el RSI indica que un actiu està sobrevenut i les Bandes de Bollinger mostren que el preu està per sota de la banda inferior, és possible que sigui un bon moment per comprar.

4.3.1. Càlcul i exploració visual dels indicadors RSI i Bandes de Bollinger

```
rsi_length = 8
bb_length = 20
```

```
data['RSI'] = ta.RSI(np.array(data['Close']), rsi_length)
data['BB_UPPER'], data['BB_MID'], data['BB_LOWER'] = ta.BBANDS(np.array(data['Close']), timeperiod = bb_length)
```



En el gràfic podem observar com els indicadors ens detecten oportunitats de compra (en verd) quan el RSI (segon gràfic) està per sota del llindar inferior i el preu està per sota de la banda de Bollinger inferior (primer gràfic). També oportunitats de venda (en vermell) quan el RSI està per sobre del llindar superior i el preu està per sobre de la banda de Bollinger superior.

4.3.2. Backtesting de l'estratègia basada en RSI i Bandes de Bollinger

Per realitzar les proves de backtesting cal adaptar la classe "tester" per l'estratègia actual. Mostrem a continuació la funció **prepare_data** que conté les principals modificacions de la classe, el càlcul dels indicadors i la generació del senyal de compravenda:

```
def prepare_data(self, start, end, inputs):
    ''' Prepara les dades pel testing.
    '''

    data = self.data.loc[start:end, ["Open", "Close", "Low", "High", 'returns']].copy()
    self.tp_year = (data.Close.count() / ((data.index[-1] - data.index[0]).days / 365.25))
    self.inputs = inputs

    ######################### Strategy-Specific ############################

    rsi_length = inputs[0]
    bb_length = inputs[1]
    ema_length = inputs[2]
    #lma = inputs[3]

    data['RSI'] = ta.RSI(np.array(data['Close']), rsi_length)
    data['BB_UPPER'], data['BB_MID'], data['BB_LOWER'] = ta.BBANDS(np.array(data['Close']), timeperiod = bb_length)
    data['EMA'] = ta.DEMA(np.array(data['Close']), ema_length)

    out_over_sold = (data['RSI'].shift() <= 30)
    out_over_bought = (data['RSI'].shift() >= 70)

    below_lower_bb = data['Close'] < data['BB_LOWER']
    over_upper_bb = data['Close'] > data['BB_UPPER']

    upper_trend = data['Close'] > data['EMA']
    lower_trend = data['Close'] < data['EMA']

    cond_long = out_over_sold & below_lower_bb #& upper_trend
    cond_short = out_over_bought & over_upper_bb #& upper_trend

    data['position'] = 0
    data.loc[cond_long, 'position'] = 1
    data.loc[cond_short, 'position'] = -1

    # Com nomes hem marcat les entrades hem de extendre les senyals
    for i in range(ema_length):
        from_long = (data['position'].shift(1) == 1) # Venim d'una entrada a llarg
        from_short = (data['position'].shift(1) == -1) # Venim d'una entrada a curt
        data.loc[from_long & (data['RSI'] < 70), 'position'] = 1
        data.loc[from_short & (data['RSI'] > 30),  'position'] = -1

    ###########################################################################

    data.dropna(inplace = True)

    self.results = data
```

Després de realitzar les adaptacions a la classe procedim a l'execució de la funció **test_strategy** passant com a paràmetres els inputs de l'estratègia. Després utilitzem la funció **plot_results** per tenir un gràfic dels retorns de l'estratègia. Observem a continuació un primer output que ens informa del rendiment que ha oferit l'estratègia així com una exploració gràfica dels guanys i les pèrdues que produeix amb els inputs donats i pel període de temps seleccionat:

```
filepath = "../csv/BTCUSDT_15m.csv"
tc = -0.00015 # Simulació dels costos del Bid-Ask spread
```

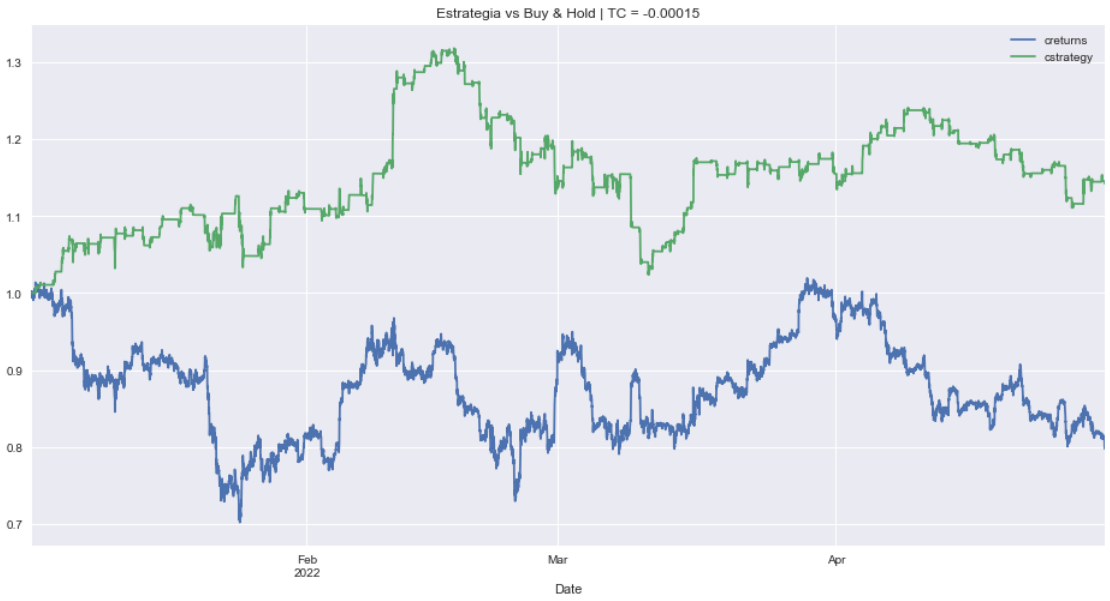
```
tester = Tester(filepath, tc)
```

```
tester.test_strategy(inputs = (12, 25, 10), start = '2022-01', end = '2022-04')
tester.plot_results()
```

```
=========================================================================================================
SIMPLE PRICE & VOLUME STRATEGY | Inputs = (12, 25, 10)
---------------------------------------------------------------------------------------------------------
PERFORMANCE MEASURES:


Multiple (Strategy):         1.1424
Multiple (Buy-and-Hold):     0.7974
--------------------------------------
Out-/Underperformance:       0.345


CAGR:                        0.5049
Annualized Mean:             0.4096
Annualized Std:              0.3498
Sharpe Ratio:                1.4435
=========================================================================================================
```



4.3.3. Optimització de l'estratègia basada en RSI i Bandes de Bollinger

La primera prova de backtesting ens mostra que l'estratègia podria arribar a ser bona, però encara no ens dona el rendiment desitjat. Procedim a l'optimització dels inputs per mirar de millorar el seu rendiment. Per aquesta tasca definim rangs de valors per cada input i els passem com a paràmetres per la funció **optimize_strategy:**

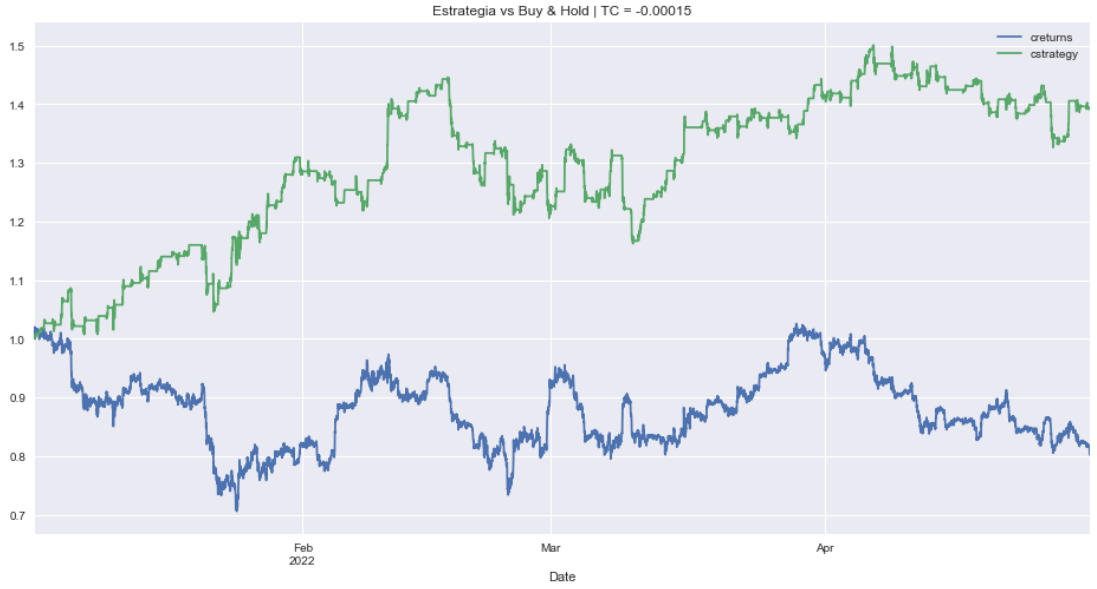
```
RSI_range = (5, 50, 5)
BB_range = (10, 110, 10)
EMA_range = (5, 110, 10)
```

```
tester.optimize_strategy(ranges=(RSI_range, BB_range, EMA_range), start='2022-01', end='2022-04')
tester.plot_results()
```

```
RSI: 10 | BB: 60 | EMA: 15 | Multiple: 1.3909
=========================================================================================================
SIMPLE PRICE & VOLUME STRATEGY | Inputs = (10, 60, 15)
---------------------------------------------------------------------------------------------------------
PERFORMANCE MEASURES:


Multiple (Strategy):         1.3909
Multiple (Buy-and-Hold):     0.8023
---------------------------------------
Out-/Underperformance:       0.5886


CAGR:                        1.753
Annualized Mean:             1.018
Annualized Std:              0.415
Sharpe Ratio:                4.2244
=========================================================================================================
```



Com podem observar l'estratègia millora molt amb els paràmetres d'entrada optimitzats per un conjunt de dades específics. Mostrant un múltiple d'inversió de 1.39 i un Sharpe de 4.22. Tot i això, els paràmetres d'entrada ara estan optimitzats per un conjunt de dades concret, el que vol dir que hem fet que l'estratègia s'adapti perfectament als canvis de preu donats en aquest període de temps. Ara cal que testejar com funcionaria l'estratègia optimitzada per un conjunt de dades diferent.

4.3.4. Rendiment mensual mitjà de l'estratègia basada en RSI i Bandes de Bollinger

En aquest punt només ens queda comprovar si l'estratègia optimitzada funciona correctament per un conjunt de dades diferent de l'utilitzat per l'optimització: Fem proves mensuals per veure com es comporta en l'escenari plantejat:

```
strategy_multiples = []
outperfs = []

for mes in meses:
    tester.test_strategy(inputs = (5, 65, 15), start=mes, end=mes)
    strategy_multiples.append(tester.strategy_multiple)
    outperfs.append(tester.outperf)

print(100 * "=")
print("Average investment multiple: {}".format(pd.DataFrame(strategy_multiples).mean()))
print("Average outperformance versus (Buy & Hold): {}".format(pd.DataFrame(outperfs).mean()))
print(100 * "=")
```

```
====================================================================================================
Average investment multiple: 0   1.0792
dtype: float64
Average outperformance versus (Buy & Hold): 0   0.1602
dtype: float64
====================================================================================================
```

Les proves realitzades mostren que l'estratègia podria generar un 8% aproximat de beneficis mensuals, un 96% anual. En termes econòmics, planteja un rendiment molt superior al que ofereix un fons d'inversió borsari que sol oferir un 8-12% anual.

4.4. Estratègia basada en MACD i DEMA

Aquesta estratègia està basada en l'indicador MACD o Moving Average Convergence Divergence i l'indicador DEMA o Double Exponential Moving Average, amb els que es tracta de detectar les oportunitats de compra o venda.

- El MACD és un indicador tècnic que mesura la relació entre dues SMA (una curta i una llarga) per detectar canvis en les tendències del mercat. Aquest indicador també inclou una línia de senyal que és una mitjana mòbil de la diferència entre les dues mitjanes mòbils. Quan el MACD creua per sobre de la seva línia de senyal, es considera un senyal de compra, mentre que, quan el MACD creua per sota de la seva línia de senyal, es considera un senyal de venda. A més, quan el MACD i la seva línia de senyal es troben per sobre de zero, és considerat que hi ha una tendència alcista, mentre que quan es troben per sota de zero, es considera una tendència baixista.
- La DEMA és un indicador tècnic similar a la SMA, però en lloc de donar el mateix pes a cada preu de l'històric, li dona un pes més gran als preus més recents. Això significa que l'DEMA és més sensible als canvis recents en el preu i, per tant, pot ser més útil per detectar canvis en la tendència termini. Per aquesta estratègia la DEMA ens permetrà operar a favor de la tendència més general del preu, entrant en compres només si la tendència és alcista i en vendes en només si la tendència és baixista.

4.4.1. Càlcul i exploració visual dels indicadors MACD i DEMA

```
data['SMA12'] = data['Close'].rolling(12).mean()
data['SMA26'] = data['Close'].rolling(26).mean()
data['MACD'] = data['SMA12'] - data['SMA26']
data['SIGNAL'] = data['MACD'].rolling(9).mean()
data['HIST'] = data['MACD']-data['SIGNAL']
data['MA200'] = ta.MA(np.array(data['Close']), timeperiod = 200)
```

Observem el segon gràfic que representa els tres components del MACD: la diferència de mitjanes (blau clar), la línia de senyal (negre), i l'histograma de barres, la diferència dels dos primers en forma d'oscil·lador. En aquest cas (cercle vermell) podem veure com el MACD creua per sota de la línia de senyal indicant-nos una possible entrada per una posició a curt o venda.

Al primer gràfic de veles podem veure la DEMA que ens marca la tendència general del preu. En aquest gràfic el preu passa per sota de la DEMA mostrant-nos l'inici d'una tendència baixista. Prenem això com la confirmació de l'entrada que senyalitzava el MACD.

4.4.2. Backtesting de l'estratègia basada en MACD i DEMA

Per realitzar les proves de backtesting cal adaptar la classe "tester" per l'estratègia actual. Mostrem a continuació la funció **prepare_data** que conté les principals modificacions de la classe, el càlcul dels indicadors i la generació del senyal de compravenda:

```
def prepare_data(self, start, end, inputs):
    ''' Prepara les dades pel testing.
    '''

    data = self.data.loc[start:end, ["Open", "Close", "Low", "High", 'returns']].copy()
    self.tp_year = (data.Close.count() / ((data.index[-1] - data.index[0]).days / 365.25))
    self.inputs = inputs

    ########################## Strategy-Specific ############################

    sma_length = inputs[0]
    lma_mul_length = inputs[1]
    signal_length = inputs[2]
    trend_length = inputs[3]

    data['SMA'] = ta.EMA(data['Close'], sma_length)
    data['LMA'] = ta.EMA(data['Close'], sma_length*lma_mul_length)
    data['MACD'] = data['SMA'] - data['LMA']
    data['SIGNAL'] = data['MACD'].rolling(int(signal_length)).mean()
    data['TREND'] = data['Close'].rolling(int(trend_length)).mean()

    # find cross overs
    crossover = (data['MACD'].shift(1) <= data['SIGNAL'].shift(1)) & (data['MACD'] > data['SIGNAL'])
    # find cross unders
    crossunder = (data['MACD'].shift(1) >= data['SIGNAL'].shift(1)) & (data['MACD'] < data['SIGNAL'])

    # locate macd sign
    under_zero = data['MACD'] < 0
    over_zero = data['MACD'] > 0

    # detect positions reversals
    upper_trend = data['Close'] > data['TREND']
    cond_long = crossover & under_zero & upper_trend
    cond_short = crossunder & over_zero & upper_trend



    data['position'] = 0
    data.loc[cond_long, 'position'] = 1
    data.loc[cond_short, 'position'] = -1

    # replicate position until next reversal sign
    for i in range(100):
        from_long = (data['position'].shift(1) == 1) # Venim d'una entrada a llarg
        from_short = (data['position'].shift(1) == -1) # Venim d'una entrada a curt
        data.loc[from_long & (data['MACD'] > data['SIGNAL']), 'position'] = 1
        data.loc[from_short & (data['MACD'] < data['SIGNAL']), 'position'] = -1

    ###########################################################################

    data.dropna(inplace = True)

    self.results = data
```

Després de realitzar les adaptacions a la classe procedim a l'execució de la funció **test_strategy** passant com a paràmetres els inputs de l'estratègia. Després utilitzem la funció **plot_results** per tenir un gràfic dels retorns de l'estratègia. Observem a continuació un primer output que ens informa del rendiment que ha oferit l'estratègia així com una exploració gràfica dels guanys i les pèrdues que produeix amb els inputs donats i pel període de temps seleccionat:

```
filepath = "../csv/BTCUSDT_30m.csv"
tc = -0.00015
```

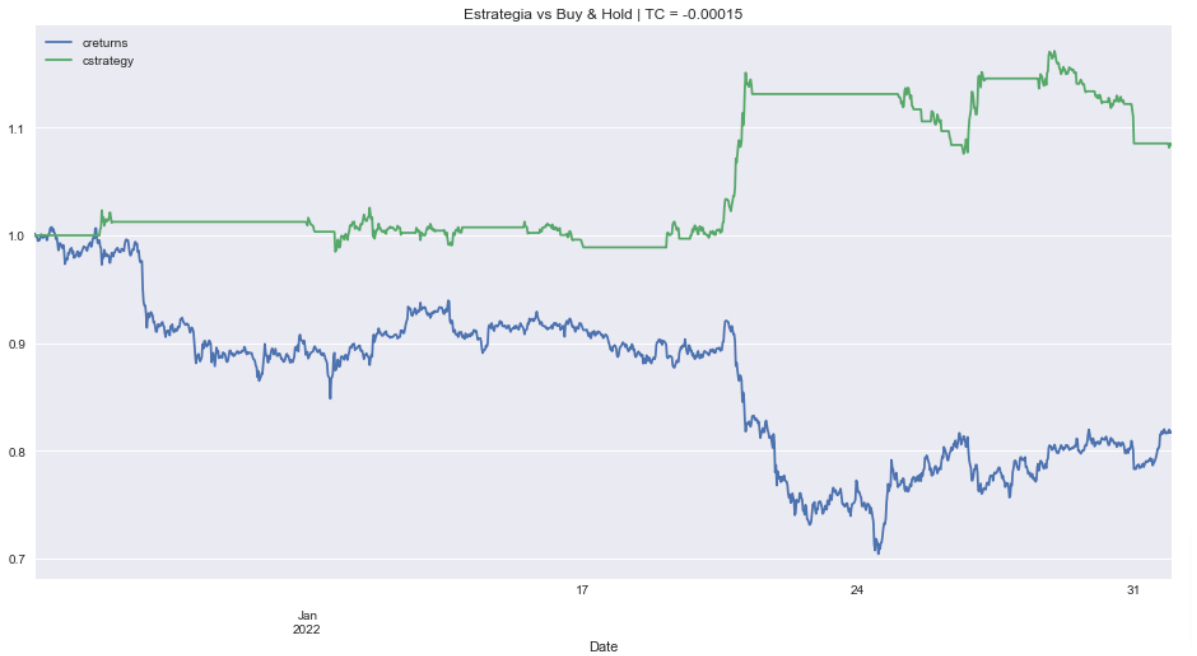
```
tester = Tester(filepath, tc)
```

```
tester.test_strategy(inputs = (12, 2, 10, 100), start = "2022-01", end = "2022-01")
tester.plot_results()
```

```
================================================================================================================
SIMPLE PRICE & VOLUME STRATEGY | Inputs = (12, 2, 10, 100)
----------------------------------------------------------------------------------------------------------------
PERFORMANCE MEASURES:


Multiple (Strategy):         1.0836
Multiple (Buy-and-Hold):     0.8178
--------------------------------------
Out-/Underperformance:       0.2658


CAGR:                        1.8499
Annualized Mean:             1.0479
Annualized Std:              0.3691
Sharpe Ratio:                5.0116
================================================================================================================
```



4.4.3. Optimització de l'estratègia basada MACD i DEMA

La primera prova de backtesting ens mostra que l'estratègia podria arribar a ser bona, però encara no ens dona el rendiment desitjat. Procedim a l'optimització dels inputs per mirar de millorar el seu rendiment. Per aquesta tasca definim rangs de valors per cada input i els passem com a paràmetres per la funció **optimize_strategy:**

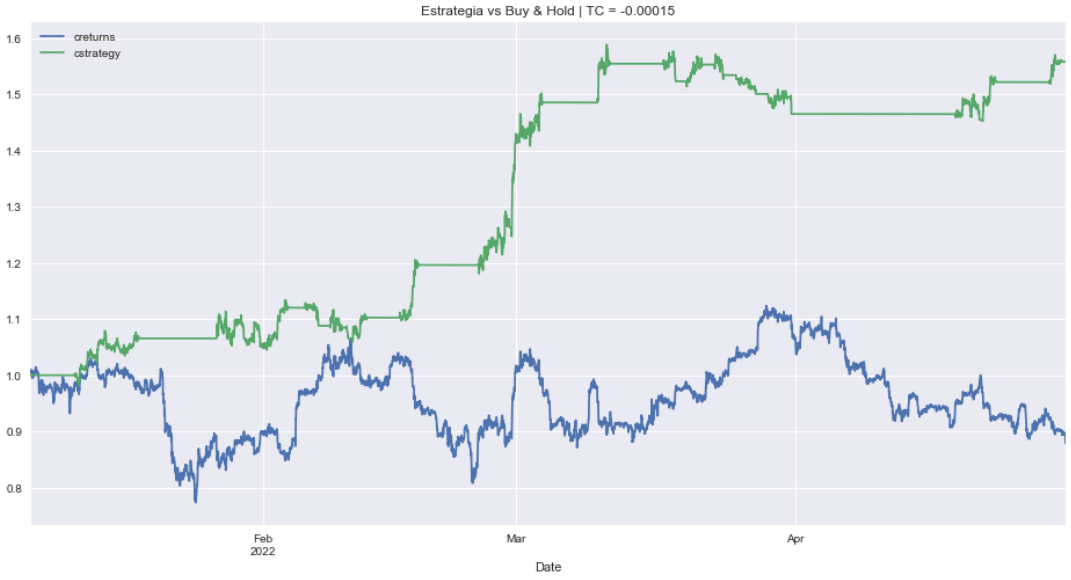
```
s_range = (5, 75, 5)
mul_l_range = (1.5, 4, 0.5)
signal_range = (5, 75, 5)
trend_range = (50, 300, 50)
```

```
tester.optimize_strategy(ranges = (s_range, mul_l_range, signal_range, trend_range), start='2022-01', end='2022-04')
tester.plot_results()
```

```
SMA: 30.0 | LMA: 1.5 | SIGNAL: 70.0 | TREND: 250.0 | Multiple: 1.5574
=========================================================================================================
SIMPLE PRICE & VOLUME STRATEGY | Inputs = (30.0, 1.5, 70.0, 250.0)
---------------------------------------------------------------------------------------------------------
PERFORMANCE MEASURES:


Multiple (Strategy):         1.5574
Multiple (Buy-and-Hold):     0.8788
-------------------------------------
Out-/Underperformance:       0.6786


CAGR:                        3.1347
Annualized Mean:             1.4215
Annualized Std:              0.3459
Sharpe Ratio:                9.0615
=========================================================================================================
```



Com podem observar l'estratègia millora molt amb els paràmetres d'entrada optimitzats per un conjunt de dades específics. Mostrant un múltiple d'inversió de 1.55 i un Sharpe de 9.06. Tot i això, els paràmetres d'entrada ara estan optimitzats per un conjunt de dades concret, el que vol dir que hem fet que l'estratègia s'adapti perfectament als canvis de preu donats en aquest període de temps. Ara cal que testejar com funcionaria l'estratègia optimitzada per un conjunt de dades diferent.

4.4.4. Rendiment mensual mitjà de l'estratègia basada en MACD I DEMA

En aquest punt només ens queda comprovar si l'estratègia optimitzada funciona correctament per un conjunt de dades diferent de l'utilitzat per l'optimització: Fem proves mensuals per veure com es comporta en l'escenari plantejat:

```
strategy_multiples = []
outperfs = []

for mes in meses:
    tester.test_strategy(inputs = (10, 1.5, 15, 250), start=mes, end=mes)
    strategy_multiples.append(tester.strategy_multiple)
    outperfs.append(tester.outperf)

print(100 * "=")
print("Average investment multiple: {}".format(pd.DataFrame(strategy_multiples).mean()))
print("Average outperformance versus (Buy & Hold): {}".format(pd.DataFrame(outperfs).mean()))
print(100 * "=")
```

```
====================================================================================================
Average investment multiple: 0   1.0870
dtype: float64
Average outperformance versus (Buy & Hold): 0   0.1651
dtype: float64
====================================================================================================
```

Les proves realitzades mostren que l'estratègia produiria un 9% aproximat de pèrdues mensuals, un 108% anual. En termes econòmics, planteja un rendiment molt superior al que ofereix un fons d'inversió borsari que sol oferir un 8-12% anual. Una estratègia que sens dubte podem considerar per la implementació final dels robots.

4.5. Estratègia basada en SQUEEZM i ADX

Per aquesta estratègia utilitzarem un indicador més complex anomenat Squeez Momentum o SM. Es calcula a partir de dos indicadors: els Keltner Channels o KC i les Bandes de Bollinger o BB.

Tant els Keltner Channels com les Bandes de Bollinger disposen una línia que és una mitjana mòbil i mesuren la variabilitat del preu dibuixant dues línies més, una superior i una inferior. La principal diferència entre les Bollinger Bands i les Keltner Channels és la manera en què es calculen les bandes superiors i inferiors. Les Bollinger Bands utilitzen desviacions estàndard per calcular les bandes, mentre que les Keltner Channels utilitzen ATR o Average True Range. I l'Average True Range és la mitjana dels rangs màxims i mínims. Això fa que les Bollinger Bands siguin més sensibles a les fluctuacions dels preus i les Keltner Channels siguin més sensibles a la volatilitat dels preus.

Per entendre aquest indicador cal saber que el mercat passa per períodes en els quals es mou en alguna direcció, i moments en els quals el preu es consolida mostrant una volatilitat més baixa. L'Squeeze Momentum tracta de detectar els moments en els quals el mercat passa d'una consolidació a una tendència. Això passa quan les Bollinger Bands s'obren i surten per fora dels Keltner Channels, mostren el possible inici d'un moviment alcista o baixista, depenent si el preu va per sobre o per sota respectivament de la mitjana mòbil.

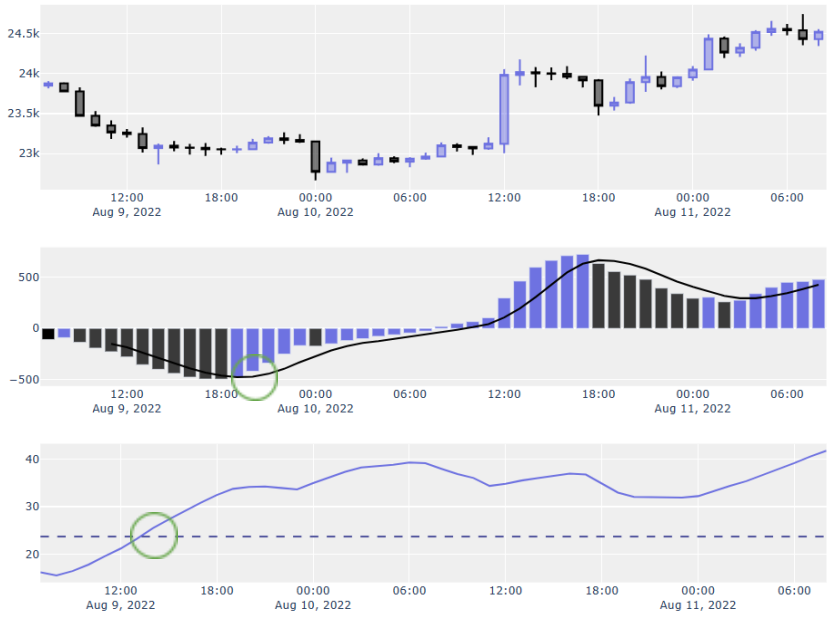
Addicionalment, per confirmar la consolidació dels moviments i detectar bons moments de compra i venda utilitzarem l'indicador ADX o Average Directional Index, un càlcul estadístic del preu que varia entre 0 i 100, on valors alts indiquen una tendència forta i valors baixos indiquen una tendència feble o absència de tendència.

4.5.1. Càlcul i exploració visual dels indicadors SQUEEZM i ADX

```
kc_length = 15
adx_length = 15
adx_thresh = 50

# Calcular els Keltner Channels
m_avg = data['Close'].rolling(kc_length).mean()
atr = ta.TRANGE(data["High"], data["Low"], data["Close"]).rolling(kc_length).mean()
data['ADX'] = ta.ADX(data["High"], data["Low"], data["Close"], adx_length)
data['KC_UPPER'] = m_avg + atr * 1.5
data['KC_LOWER'] = m_avg - atr * 1.5

# Moment
highest = data['High'].rolling(kc_length).max()
lowest = data['Low'].rolling(kc_length).min()
m1 = (highest + lowest) / 2
data['MOM'] = (data['Close'] - (m1 + m_avg)/2)
fit_y = np.array(range(0, kc_length))
data['MOM'] = data['MOM'].rolling(kc_length).apply(
    lambda x : np.polyfit(fit_y, x, 1)[0] * (kc_length-1) + np.polyfit(fit_y, x, 1)[1], raw=True)
```



En aquest cas només ens interessa la part de l'indicador que constitueix el que anomenem un oscil·lador (segon gràfic), una estadística que oscil·la al voltant de zero. L'oscil·lador del SQUEEZM pren valors positius quan un moviment alcista del mercat agafa força, i valors negatius quan un moviment baixista agafa força. Hem calculat una mitjana mòbil simple SMA de l'oscil·lador (línia negra) per detectar millor el començament dels moviments alcistes quan l'oscil·lador mostra barres blaves i estan per sobre de la SMA, marcant l'entrada a una possible compra.

Si mirem el tercer gràfic, que correspon a l'ADX, veurem que l'hem complementat amb una línia que fa de llindar marcant-nos quan els moviments es consoliden. L'ADX ha d'estar per sobre del llindar per considerar que el moviment està consolidat i és segur entrar en una operació. En aquest cas podem veure com l'ADX ens confirma l'entrada del SQUEEZM que ens faria guanyar diners.

4.5.2. Backtesting de l'estratègia basada en SQUEEZM i ADX

Per realitzar les proves de backtesting cal adaptar la classe "tester" per l'estratègia actual. Mostrem a continuació la funció **prepare_data** que conté les principals modificacions de la classe, el càlcul dels indicadors i la generació del senyal de compravenda:

```
def prepare_data(self, start, end, inputs):
    ''' Prepara les dades pel testing.
    '''

    data = self.data.loc[start:end, ["Open", "Close", "Low", "High", 'returns']].copy()
    self.tp_year = (data.Close.count() / ((data.index[-1] - data.index[0]).days / 365.25))
    self.inputs = inputs

    ########################## Strategy-Specific ############################

    kc_length = inputs[0]
    adx_length = inputs[1]
    adx_thresh = inputs[2]

    # Keltner Channel
    m_avg = ta.EMA(data['Close'], kc_length)
    atr = ta.TRANGE(data["High"], data["Low"], data["Close"]).rolling(kc_length).mean()
    data['ADX'] = ta.ADX(data["High"], data["Low"], data["Close"], adx_length)
    data['KC_UPPER'] = m_avg + atr * 1.5
    data['KC_LOWER'] = m_avg - atr * 1.5

    # Moment
    highest = data['High'].rolling(kc_length).max()
    lowest = data['Low'].rolling(kc_length).min()
    m1 = (highest + lowest) / 2
    data['MOM'] = (data['Close'] - (m1 + m_avg)/2)
    data['MOM'] = data['MOM'].rolling(kc_length).mean()

    bull_mom = data['MOM'] > data['MOM'].rolling(5).mean()
    bear_mom = data['MOM'] < data['MOM'].rolling(5).mean()
    adx_over = data['ADX'] > np.percentile(np.array(data['ADX'].dropna()), adx_thresh)

    cond_short = bear_mom & adx_over
    cond_long = bull_mom & adx_over

    data['position'] = 0
    data.loc[cond_long, 'position'] = 1
    data.loc[cond_short, 'position'] = -1

    for i in range(100):
        from_long = (data['position'].shift(1) > 0)
        from_short = (data['position'].shift(1) < 0)
        data.loc[from_long & (data['MOM'] > data['MOM'].rolling(5).mean()), 'position'] = 1
        data.loc[from_short & (data['MOM'] < data['MOM'].rolling(5).mean()), 'position'] = -1

    ###########################################################################

    data.dropna(inplace = True)

    self.results = data
```

Després de realitzar les adaptacions a la classe procedim a l'execució de la funció **test_strategy** passant com a paràmetres els inputs de l'estratègia. Després utilitzem la funció **plot_results** per tenir un gràfic dels retorns de l'estratègia. Observem a continuació un primer output que ens informa del rendiment que ha oferit l'estratègia així com una exploració gràfica dels guanys i les pèrdues que produeix amb els inputs donats i pel període de temps seleccionat:

```
filepath = "../csv/BTCUSDT_15m.csv"
tc = -0.00015
```

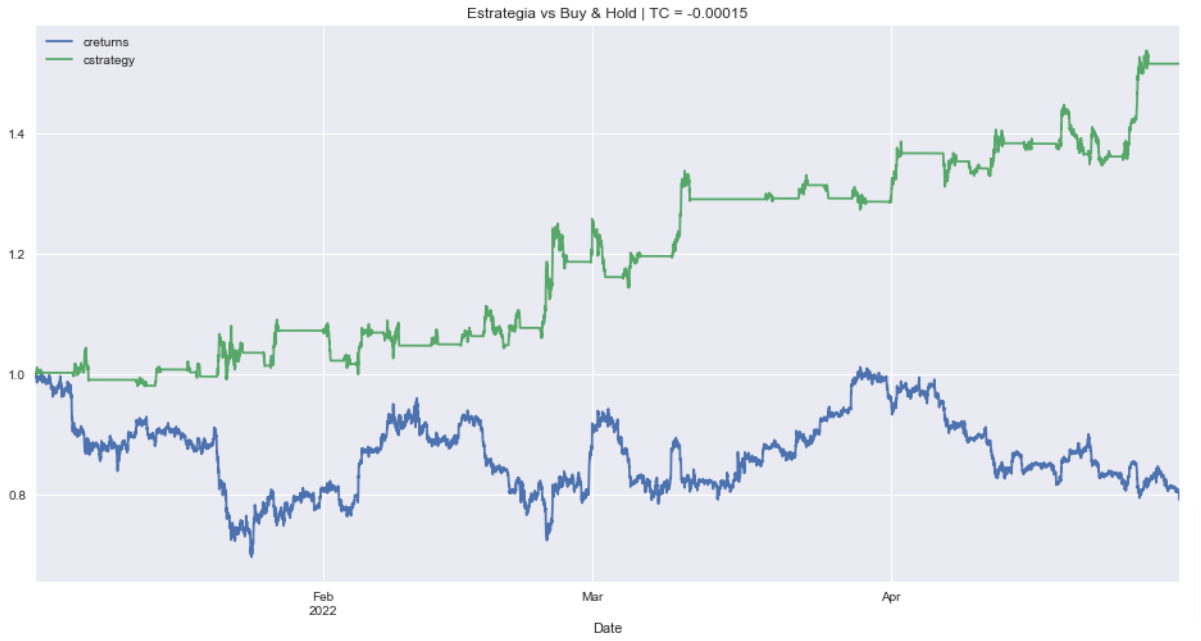
```
tester = Tester(filepath, tc)
```

```
tester.test_strategy(inputs = (50, 40, 70), start="2022-01", end="2022-04")
tester.plot_results()
```

```
================================================================================================================
SIMPLE PRICE & VOLUME STRATEGY | Inputs = (50, 40, 70)
----------------------------------------------------------------------------------------------------------------
PERFORMANCE MEASURES:


Multiple (Strategy):         1.5163
Multiple (Buy-and-Hold):     0.7915
--------------------------------------
Out-/Underperformance:       0.7248


CAGR:                        2.6274
Annualized Mean:             1.2887
Annualized Std:              0.3993
Sharpe Ratio:                6.5806
================================================================================================================
```



4.5.3. Optimització de l'estratègia basada SQUEEZM i ADX

La primera prova de backtesting ens mostra que l'estratègia podria arribar a ser bona, però encara no ens dona el rendiment desitjat. Procedim a l'optimització dels inputs per mirar de millorar el seu rendiment. Per aquesta tasca definim rangs de valors per cada input i els passem com a paràmetres per la funció **optimize_strategy:**

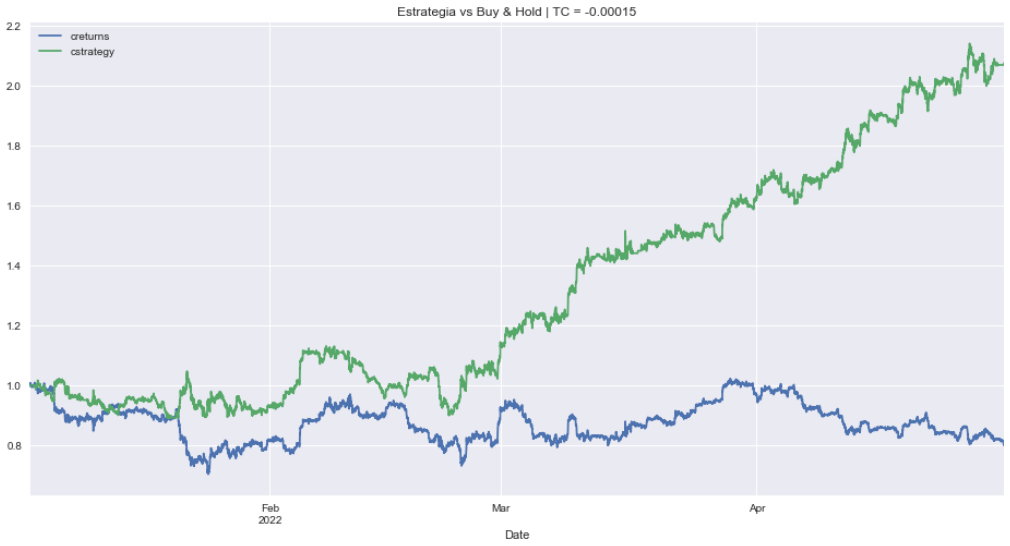
```
KC_range = (90, 151, 10)
ADX_range = (10, 101, 10)
THRESH_range = (10, 100, 10)
```

```
tester.optimize_strategy(ranges = (KC_range, ADX_range, THRESH_range), start='2022-01', end='2022-04', metric = 'Multiple')
tester.plot_results()
```

```
KC: 90 | ADX: 20 | THRESH: 20 | Multiple: 2.0746
================================================================================================================
SIMPLE PRICE & VOLUME STRATEGY | Inputs = (90, 20, 20)
----------------------------------------------------------------------------------------------------------------
PERFORMANCE MEASURES:


Multiple (Strategy):         2.0746
Multiple (Buy-and-Hold):     0.8001
-------------------------------------
Out-/Underperformance:       1.2745


CAGR:                        8.5728
Annualized Mean:             2.2753
Annualized Std:              0.6177
Sharpe Ratio:                13.8784
================================================================================================================
```



Com podem observar l'estratègia millora molt amb els paràmetres d'entrada optimitzats per un conjunt de dades específics. Mostrant un múltiple d'inversió de 2.07 i un Sharpe de 13.87. Tot i així els paràmetres d'entrada ara estan optimitzats per un conjunt de dades concret, el que vol dir que hem fet que l'estratègia s'adapti perfectament als canvis de preu donats en aquest període de temps. Ara cal que testejar com funcionaria l'estratègia optimitzada per un conjunt de dades diferent.

4.5.4. Rendiment mensual mitjà de l'estratègia basada en SQUEEZM i ADX

En aquest punt només ens queda comprovar si l'estratègia optimitzada funciona correctament per un conjunt de dades diferent de l'utilitzat per l'optimització: Fem proves mensuals per veure com es comporta en l'escenari plantejat:

```
strategy_multiples = []
outperfs = []

for mes in meses:
    tester.test_strategy(inputs = (90, 20, 20), start=mes, end=mes)
    strategy_multiples.append(tester.strategy_multiple)
    outperfs.append(tester.outperf)
    tester.plot_results()

print(100 * "=")
print("Average investment multiple: {}".format(pd.DataFrame(strategy_multiples).mean()))
print("Average outperformance versus (Buy & Hold): {}".format(pd.DataFrame(outperfs).mean()))
print(100 * "=")
```

```
====================================================================================================
Average investment multiple: 0   1.1288
dtype: float64
Average outperformance versus (Buy & Hold): 0   0.1930
dtype: float64
====================================================================================================
```

Les proves realitzades mostren que l'estratègia podria generar un 12% aproximat de beneficis mensuals, un 144% anual. En termes econòmics, planteja un rendiment molt superior al que ofereix un fons d'inversió borsari que sol oferir un 8-12% anual. Una estratègia que sens dubte podem considerar per la implementació final dels robots.

5. Desenvolupament dels robots

Després de les proves de backtesting realitzades, considerem que l'enfocament basat en indicadors és l'únic que ens pot servir com a solució del problema. Escollirem aquest sistema com a eina que faran servir els nostres robots finals. Això vol dir que pel desenvolupament dels robots haurem de traslladar l'aplicació de les estratègies basades en indicadors a un entorn real.

La complexitat de la implementació plantejada recau sobre la connexió i la interacció amb el mercat. És a dir, hem de veure com donar als robots la capacitat de connectar-se i interactuar amb la blockchain. Per acomplir aquesta tasca utilitzarem la Binance API.

Binance és una plataforma on es poden comprar i vendre criptomonedes. La seva API permet accedir a les dades de la plataforma i efectuar operacions. Així els desenvolupadors poden crear aplicacions, que es connectin a la plataforma.

Cal esmentar que els robots connectaran amb la testnet de Binance. Una testnet és una xarxa de prova que s'utilitza per testejar aplicacions sense utilitzar diners reals. Aquesta xarxa que farem servir és una còpia de la xarxa principal de la blockchain de Bitcoin, però utilitza monedes simulades en lloc de monedes reals.

Respecte a la generació dels senyals de compravenda, és una tasca que faran els robots de forma contínua de la mateixa forma que en les proves de backtesting. Per tant ens portarem part del codi de la classe "tester" per la creació dels robots. A continuació explicarem la implementació final, la classe "trader".

5.1. Implementació de la classe “trader”

Aquesta classe serà la base per la creació de cada robot. Un cop creada, només caldrà canviar algunes línies de codi per adaptar els robots a cada estratègia Ara explicarem un per un cada mètode de la classe “trader”:

- **__init__**: és el constructor de la classe que s'executa quan es crea una nova instància de la classe. Serveix per establir els valors inicials dels atributs de classe. Podem observar atributs genèrics que tindran tots els robots i específics de l’estratègia que farà servir.

```
def __init__(self, symbol, bar_length, inputs, units, position = 0):

    self.symbol = symbol
    self.bar_length = bar_length
    self.available_intervals = ["1m", "3m", "5m", "15m", "30m", "1h", "2h",
                                "4h", "6h", "8h", "12h", "1d", "3d", "1w", "1M"]
    self.units = units
    self.position = position
    self.trades = 0
    self.trade_values = []
    self.cum_profits = 0

    #***************** Atributs especifics de la estrategia *****************
    self.kc_length = inputs[0]
    self.adx_length = inputs[1]
    self.thresh = inputs[2]
    #************************************************************************
```

- **start_trading**: s'utilitza per iniciar el trading utilitzant una estratègia específica. En la funció, es crea un thread que permet que el robot comenci a rebre cada pocs segons un missatge/actualització del preu de la criptomoneda. A més, serveix per cridar la funció que demana a l'API les dades històriques del preu

```
def start_trading(self, historical_days):

    self.twm = ThreadedWebsocketManager()
    self.twm.start()

    if self.bar_length in self.available_intervals:
        self.get_most_recent(symbol = self.symbol, interval = self.bar_length,
                             days = historical_days)
        self.twm.start_kline_socket(callback = self.stream_candles,
                                    symbol = self.symbol, interval = self.bar_length)
```

- **get_most_recent**: té la responsabilitat de demanar a l'API les dades històriques OCHL perquè el robot comenci a calcular el senyal de compravenda. El paràmetre d'entrada *days* índica el nombre de dies de dades històriques que demanem a l'API. Cal tenir en compte quantes dades són necessàries per al càlcul dels indicadors de cada estratègia i demanar suficients dades.

```
def get_most_recent(self, symbol, interval, days):

    now = datetime.utcnow()
    past = str(now - timedelta(days = days))

    bars = client.get_historical_klines(symbol = symbol, interval = interval,
                                        start_str = past, end_str = None, limit = 1000)
    df = pd.DataFrame(bars)
    df["Date"] = pd.to_datetime(df.iloc[:,0], unit = "ms")
    df.columns = ["Open Time", "Open", "High", "Low", "Close", "Volume",
                  "Clos Time", "Quote Asset Volume", "Number of Trades",
                  "Taker Buy Base Asset Volume", "Taker Buy Quote Asset Volume", "Ignore", "Date"]
    df = df[["Date", "Open", "High", "Low", "Close", "Volume"]].copy()
    df.set_index("Date", inplace = True)
    for column in df.columns:
        df[column] = pd.to_numeric(df[column], errors = "coerce")
    df["Complete"] = [True for row in range(len(df)-1)] + [False]

    self.data = df
```

- **stream_candles**: aquesta funció es crida recurrentment cada cop que el robot rep una actualització de dades OCHL. La seva responsabilitat és tractar la nova informació i detectar si s'ha tancat la darrera espelma, és a dir, l'última vela de la sèrie ja disposa de preu de clausura. En cas afirmatiu crida a les funcions responsables d'executar l'estratègia.

```
def stream_candles(self, msg):

    # Extracció dels items del missatge
    event_time = pd.to_datetime(msg["E"], unit = "ms")
    start_time = pd.to_datetime(msg["k"]["t"], unit = "ms")
    first   = float(msg["k"]["o"])
    high    = float(msg["k"]["h"])
    low     = float(msg["k"]["l"])
    close   = float(msg["k"]["c"])
    volume  = float(msg["k"]["v"])
    complete=       msg["k"]["x"]

    # Un print com a feedback visual de que el robot esta actiu
    print(".", end = "", flush = True)

    # Afegim / Actualitzem última temporalitat / espelma
    self.data.loc[start_time] = [first, high, low, close, volume, complete]

    # Si hem tancat temporalitat / espelma apliquem estrategia
    if complete == True:
        self.define_strategy()
        self.execute_trades()
```

- **define_strategy**: aquesta funció realitza els càlculs dels indicadors de l'estratègia cada cop que es rep una nova espelma tancada. Després genera el senyal de trading segons el que diuen els indicadors sobre la situació del mercat.

```
def define_strategy(self):

    data = self.data.copy()

    #******************** Definició de la estratègia ***********************
    # Inputs
    kc_length = self.kc_length
    adx_length = self.adx_length
    adx_thresh = self.thresh


    # True Range
    data['tr0'] = abs(data["High"] - data["Close"])
    data['tr1'] = abs(data["High"] - data["Close"].shift())
    data['tr2'] = abs(data["Close"] - data["Close"].shift())
    data['tr'] = data[['tr0', 'tr1', 'tr2']].max(axis=1)

    # Keltner Channel
    range_ma = data['tr'].rolling(kc_length).mean()
    m_avg = data['Close'].rolling(kc_length).mean()
    data['KC_UPPER'] = m_avg + range_ma * 1.5
    data['KC_LOWER'] = m_avg - range_ma * 1.5

    # Moment
    highest = data['High'].rolling(kc_length).max()
    lowest = data['Close'].rolling(kc_length).min()
    m1 = (highest + lowest) / 2
    data['MOM'] = (data['Close'] - (m1 + m_avg)/2)
    fit_y = np.array(range(0, kc_length))
    data['MOM'] = data['MOM'].rolling(kc_length).apply(
        lambda x : np.polyfit(fit_y, x, 1)[0] * (kc_length-1) + np.polyfit(fit_y, x, 1)[1], raw=True)

    data['ADX'] = ta.ADX(data["High"], data["Close"], data["Close"], adx_length)

    bull_mom = data['MOM'] > data['MOM'].rolling(5).mean()
    bear_mom = data['MOM'] < data['MOM'].rolling(5).mean()
    adx_over = data['ADX'] > np.percentile(np.array(data['ADX'].dropna()), adx_thresh)

    cond_short = bear_mom & adx_over
    cond_long = bull_mom & adx_over

    data['position'] = 0
    data.loc[cond_long, 'position'] = 1
    data.loc[cond_short, 'position'] = -1
    #***********************************************************************

    self.prepared_data = data.copy()
```

- **execute_trades**: aquest mètode es crida cada cop que el robot actualitza el senyal de compravenda i s'encarrega d'executar les accions necessàries. Desenvolupa tota una lògica condicional on es considera tant la posició actual adoptada pel robot com la darrera posició que indica el senyal actualitzat. Després executa les ordres que calgui al mercat. Un aspecte que es pot ressaltar és que les ordres realitzades estan adaptades perquè ens portin a un escenari molt similar al plantejat en el backtesting, on la inversió inicial era 1$ i reinvertíem de forma contínua els beneficis. En aquest cas la inversió inicial ve definida per l'atribut de classe "units" i observem que sempre afegim els beneficis acumulats "cum_profits" per realitzar la reinversió dels beneficis igual que al backtesting. Això ho hem fet per permetre la comparació directa entre els rendiments financers del robot i els del backtesting.

```
def execute_trades(self):

    if self.prepared_data["position"].iloc[-1] == 1: # Si la posicio es a llarg -> anem a llarg
        if self.position == 0:
            order = client.create_order(
                symbol = self.symbol, side = "BUY", type = "MARKET", quoteOrderQty = self.units + self.cum_profits)
            self.report_trade(order, "GOING LONG")
        elif self.position == -1:
            order = client.create_order(
                symbol = self.symbol, side = "BUY", type = "MARKET", quoteOrderQty = self.units + self.cum_profits)
            self.report_trade(order, "GOING NEUTRAL")
            time.sleep(0.1)
            order = client.create_order(
                symbol = self.symbol, side = "BUY", type = "MARKET", quoteOrderQty = self.units + self.cum_profits)
            self.report_trade(order, "GOING LONG")
        self.position = 1
    elif self.prepared_data["position"].iloc[-1] == 0: # Si la nova posicio es neutral -> anem a neutral
        if self.position == 1:
            order = client.create_order(
                symbol = self.symbol, side = "SELL", type = "MARKET", quoteOrderQty = self.units + self.cum_profits)
            self.report_trade(order, "GOING NEUTRAL")
        elif self.position == -1:
            order = client.create_order(
                symbol = self.symbol, side = "BUY", type = "MARKET", quoteOrderQty = self.units + self.cum_profits)
            self.report_trade(order, "GOING NEUTRAL")
        self.position = 0
    if self.prepared_data["position"].iloc[-1] == -1: # Si la posicio es a curt -> anem a curt
        if self.position == 0:
            order = client.create_order(
                symbol = self.symbol, side = "SELL", type = "MARKET", quoteOrderQty = self.units + self.cum_profits)
            self.report_trade(order, "GOING SHORT")
        elif self.position == 1:
            order = client.create_order(
                symbol = self.symbol, side = "SELL", type = "MARKET", quoteOrderQty = self.units + self.cum_profits)
            self.report_trade(order, "GOING NEUTRAL")
            time.sleep(0.1)
            order = client.create_order(
                symbol = self.symbol, side = "SELL", type = "MARKET", quoteOrderQty = self.units + self.cum_profits)
            self.report_trade(order, "GOING SHORT")
        self.position = -1
```

- **report_trade**: és responsable de calcular els guanys que el robot obté en les seves operacions en el mercat i informar sobre aquestes operacions. Un detall important és que aquesta funció només realitza càlculs de guanys cada dues operacions. Això es deu al fet que per calcular un guany es necessiten dos valors: el preu en el moment d'entrar en una operació i el preu en el moment de sortir d'aquesta operació.

```
def report_trade(self, order, going):

    # extract data from order object
    side = order["side"]
    time = pd.to_datetime(order["transactTime"], unit = "ms")
    base_units = float(order["executedQty"])
    quote_units = float(order["cummulativeQuoteQty"])
    price = round(quote_units / base_units, 5)

    # Calcula els beneficis del trade
    self.trades += 1
    if side == "BUY":
        self.trade_values.append(-quote_units)
    elif side == "SELL":
        self.trade_values.append(quote_units)

    if self.trades % 2 == 0:
        real_profit = round(np.sum(self.trade_values[-2:]), 3)
        self.cum_profits = round(np.sum(self.trade_values), 3)
    else:
        real_profit = 0
        self.cum_profits = round(np.sum(self.trade_values[:-1]), 3)

    # Mostra el rendiment del trade
    print(2 * "\n" + 100* "-")
    print("{} | {}".format(time, going))
    print("{} | Base_Units = {} | Quote_Units = {} | Price = {} ".format(time, base_units, quote_units, price))
    print("{} | Profit = {} | CumProfits = {} ".format(time, real_profit, self.cum_profits))
    print(100 * "-" + "\n")
```

5.2 Especificació d'una prova en temps real pels robots finals

Un cop implementada la classe "trader" només ha de ser adaptada a cada estratègia que vulguem que executi. De les quatre estratègies amb indicadors hem decidit executar-ne les tres que ens han donat millors resultats en l'escenari de backtesting.

Tenim proves realitzades per saber el rendiment mitjà mensual que ens ofereix cada estratègia amb dades del passat. Volem, per tant, fer una prova amb la qual els robots operin durant un mes i puguem fer una comparació directa de rendiment.

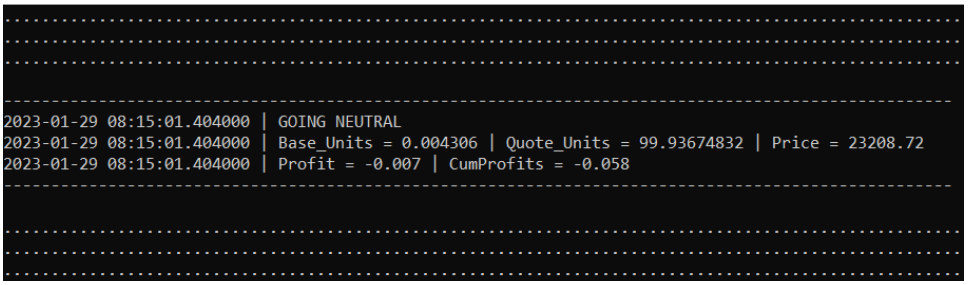
Durant aproximadament un mes des del 26 de desembre del 2022 tres robots han estat funcionant de la sguent forma:

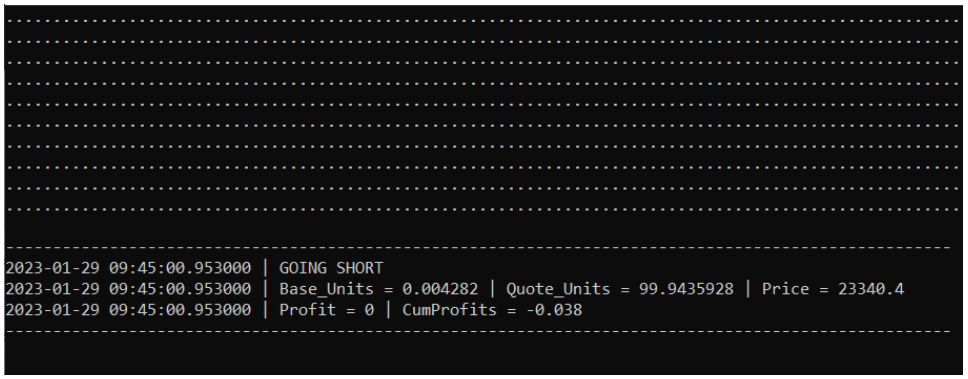
- Un robot executant l'estratègia basada en **MACD i DEMA** amb temporalitats de 15 minuts, és a dir, cada 30 minuts es completa una espelma i actualitza el seu senyal de compravenda.
- Un robot executant l'estratègia basada en els indicadors **RSI i Bandes de Bollinger** amb temporalitats de 15 minuts, és a dir, cada 15 minuts es completa una espelma i actualitza el seu senyal de compravenda.
- Un robot executant l'estratègia basada en els indicadors **SQUEEZM i ADX** amb temporalitats de 15 minuts, és a dir, cada 15 minuts es completa una espelma i actualitza el seu senyal de compravenda.

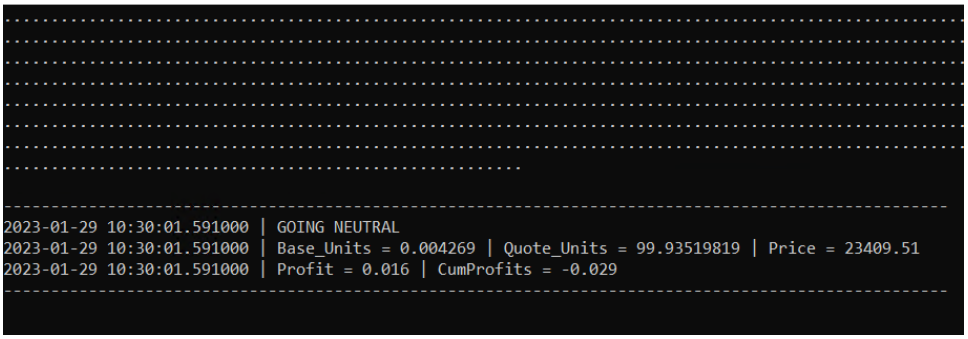
Per poder deixar els robots funcionant sense interrupcions degudes a la connectivitat o altres incidències, hem considerat que la millor idea era executar-los en el cloud. En concret, hem fet servir els serveis de cloud que ofereix Amazon.

Un problema que va sorgir és que la primera instància de cloud creada estava allotjada en North Virginia en els Estats Units, on l'API de Binance no dona servei a causa de restriccions legals. Com a resultat, el deployment dels robots no funcionava. Per tant, vam haver de cercar una regió geogràfica compatible amb l'API de Binance. Això ens demostra la importància d'investigar les restriccions legals i tècniques abans d'implementar un sistema en un entorn de cloud.

Finalment, vam executar els robots en una instància allotjada en Londres. Mostrem a continuació una mostra de l'execució dels robots funcionant al cloud:







Aquests són els darrers informes d'operacions que han donat els robots. Si ens fixem en les Quote_Units els robots reporten pèrdues d'aproximadament 0.6-0.7% en el mes de gener de 2023. Si aquest rendiment es repliqués pels mesos posteriors parlaríem de pèrdues d'un 8-9% anual en el 2023.

De totes maneres, el backtesting mostrava mesos en els quals el rendiment de les estratègies baixava i produïen pèrdues similars. El cas és que també hi havia mesos que el rendiment era molt superior al mitjà. Per tant, a favor del sistema, podem llegir els resultats de la prova final com un mes dolent, que no reflecteix el rendiment mensual mitjà real de les estratègies a llarg termini.

6. Conclusions

En aquest treball, hem abordat el desafiant problema d'automatitzar el trading de criptomonedes. Per fer-ho hem explorat la bondat de diferents sistemes que ens podien servir per prendre les decisions de compravenda adequades.

En primer lloc, hem estudiat la possibilitat de realitzar prediccions del preu amb models de regressió lineal. Hem pogut concloure que aquest mètode no era vàlid pel nostre context. La relació lineal entre els valors passats i futurs que pressuposen aquests models no era present en la variable a predir, tal com reflectia l'estudi d'autocorrelació.

També hem generat un model de classificació esperant que la transformació del problema en una categorització de les mostres ens conduïs a models de IA amb millors resultats. Però aquest sistema també ha sigut descartat com a solució pel problema. Ni la precisió ni el rendiment financer associat al senyal que generava el sistema eren satisfactoris.

En resum, segons el nostre estudi del problema hem conclòs que els sistemes basats en models de IA no són prou eficaços pel trading de criptomonedes. En aquest punt atribuïm la no idoneïtat d'aquest tipus de sistemes a la complexitat del mercat de criptomonedes, on els preus són molt volàtils i sensibles a l'especulació, fet que dificulta el tipus de prediccions que fan els models de IA.

Per altra banda, els indicadors tècnics mostren un rendiment prou satisfactori en l'escenari del backtesting, el que ens porta a concloure que són una eina útil per prendre decisions de compravenda en el mercat de criptomonedes.

De fet, una de les observacions més significatives realitzades en aquest projecte té molt a veure amb l'èxit d'aquest enfocament. És el fet que la mateixa naturalesa especulativa del mercat de criptomonedes premia el bon ús dels indicadors. Al final els indicadors són eines molt populars entre els traders de criptomonedes. Per tant, quan un indicador ens planteja una compra, molts traders faran aquesta mateixa lectura i sovint es començaran a produir compres que fan que el preu es dispari. Teoritzem que aquesta regla de comportament, encara que no és inflable, afavoreix un enfocament basat en indicadors tècnics.

Pel que fa a la prova en temps real, si bé els resultats dels indicadors no són encoratjadors, encara considerem possible que en un entorn real el sistema sigui beneficiós. En primer lloc, perquè els resultats al backtesting avalen un rendiment excel·lent en el passat i, en segon lloc, perquè només hem pogut veure que fan els robots durant el mes de gener del 2023, és a dir, considerem que la mostra del comportament observat és massa curta per ser significativa del veritable rendiment a llarg termini que ofereix el sistema.

7. Bibliografia

[1] Udemy (2021). Cryptocurrency Algorithmic Trading with Python and Binance. Disponible en: https://www.udemy.com/course/cryptocurrency-algorithmic-trading-with-python-and-binance

[2] TradingView (2021). Gráficos de Trading. Disponible en: https://www.tradingview.com/chart/

[3] Binance Academy (2021). Academia Binance. Disponible en: https://www.academy.binance.com/es

[4] Binance API Docs (2021). Documentación de la API de Binance. Disponible en: https://www.binance-docs.github.io/apidocs/spot/en/#spot-account-trade

[5] Technical Analysis Library in Python (2021). Biblioteca de Análisis Técnico en Python. Disponible en: https://www.technical-analysis-library-in-python.readthedocs.io/en/latest/ta.html

[6] LightGBM (2021). Documentació de la llibreria LightGBM. Disponible en: https://www.lightgbm.readthedocs.io/en/v3.3.2/

[7] Universidad de Deusto (2021). Criptomonedas - Algoritmos de trading en Python. Disponible en: https://www.youtube.com/watch?v=MlktVhReO0E&t=2054s&ab_channel=UniversidaddeDeusto%2FDeustukoUnibertsitatea

[8] Ciencia de Datos (2021). Pronóstico de criptomonedas: Bitcoin con aprendizaje automático en Python. Disponible en: https://www.cienciadedatos.net/documentos/py41-forecasting-cryptocurrency-bitcoin-machine-learning-python.html

[9] scikit-learn (2021). Búsqueda aleatoria con RandomizedSearchCV. Disponible en: https://www.scikit-learn.org/stable/modules/generated/sklearn.model_selection.RandomizedSearchCV.html

[10] Alpha Algorithms (2021). Cálculo de MACD en Python. Disponible en: https://www.alpharithms.com/calculate-macd-python-272222/

[11] Alpha Algorithms (2021). Índice de Fuerza Relativa (RSI) en Python. Disponible en: https://www.alpharithms.com/relative-strength-index-rsi-in-python-470209/

[12] Alpha Algorithms (2021). Bandas de Bollinger en Python. Disponible en: https://www.alpharithms.com/bollinger-bands-590615/

[13] TradingLab (2021). Trading con Python. Disponible en: https://www.youtube.com/watch?v=rf_EQvubKlk&list=PLH-wQt3UtmKMkGuIWkrjyLW_PLeE3v7v3&ab_channel=TradingLab

[14] Medium (2021). Implementing the Most Popular Indicator on TradingView Using Python. Disponible en: www.medium.com/geekculture/implementing-the-most-popular-indicator-on-tradingview-using-python-239d579412ab