✡ RSI - Indicator:-

Date
Page

$RS = \frac{\text{Avg. Gains}}{\text{Avg. losses}}$

Buy ↑ ↓ sell

calculate → Momentum → Signals.

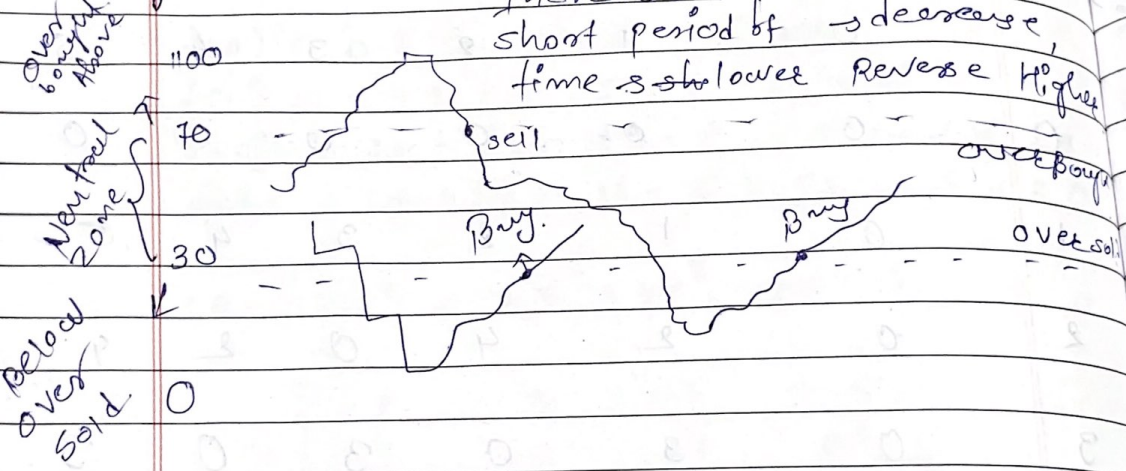
↙ ↘

Generally 14 days Relative strength

↙ over Bought

Increase Rapid short period of time → decrease, slower Reverse Higher

↘ over Sold



↳ Relative strength index:-

↳ Value Range from 0 to 100.

↳ It is Momentum indicator.

Date:- 202

↳ saving

✡ Bullish

stock:-

RSI :-

✡ Bearish

stock:-

RSI

✡ Risk

1) I

2)

3)

Buy ↑ ↓ sell

n → Signals.

ver over

uyht sold.

old ↓

t → decrease,

t Reverse Higher

overbought

overSold

Data :- 2022 to 2025 years.

Swing trading :- short term investing.

* Bullish Divergen :- over sold.

stock :-

RSI :-

Buy.

BO

* Bearish Divergence :- over bought

stock :-

RS I.

sold.

To.

* Risk :- Signal Not accurate

1) Economic News.
2) Earning.
3) Frondarmental Aspects.

1) False Signals :-

↳ Indicated overbought or over sold but the price doesn't move as expected.

2) Ineffectiveness in strong trends.

↳ During strong trends RSI can stay overbought or oversold for a long time without reversing.

3) Lagging Indicator :-

↳ RSI is based on historical Price data, so it may react slowly to sudden change in market conditions.

File

1) Financ
and te

a) Date

2) Rolling

↳ Calcul
level

↳ Simp
20 &

Δ Movi

20 day
50 da

↳ Why
1)
2) I
3) C

over sold but
s expected.

ng trends.

sf Com stay
a long time

Price data,
den change

File 01 :-

1) Financial data Analysis with yfinance and its libraries.

•) Data Download :- yfinance to fetch BTC-USD.

2) Rolling Metrics :-

b Calculates Rolling Support & Resistance level Using a 20-day window.

b Simple Moving Average (SMA) for 20 & 50 days.

A Moving Average (SMA).

20 days → short-term trend. Avg. Price-20 day.
50 days → Medium " " " "

b Why Use 20 days & 50 days :-
i) spot trend direction
2) Identify Entry & Exit points.
3) Confirm price movements.

* Golden Cross :- Buy Signal
Death Cross :- sell Signal.

2) ~~Over~~ observe the Slop :-

Upward slope :- Bullish trend
Down ward slope :- Bearish "

Risk :-

1) Lagging indicator :- Historical price, not React quickly.

2) False Signal :- give Misleading Signals.

3) Market Condition :- Works better in trending Market than Sideway market.

* observation :-

1) Multi index column :-
2) Data Integrity :- (Min, Max, mean).
3) RSI inclusion :-

code explanation

ta - ~~tab~~ library for technical Analysis

comprehensive indicator :-

(SMA), (RSI).

* data = yf.download(ticker, start = "2024-05-01", end = "2024-12-04", interval = "1D").
ticker = stock symbol you want data for
start = start date
end = end date
interval = The frequency of data (daily 1-Day intervals)

* data.reset_index(inplace = True)
↳ index back to a Regular column, usually labeled "Date" After Downloading.

* ↳ inplace = True :-
The function modifies the original Data Frame directly, & Nothing is Returned.

↳ without true :- Returns a Modified df.

20 days data
* data['SMA_20'] = data['close'].rolling(window = 20).mean().
↳ closing price of stock
↳ 20 data points (20 days) or less.

Part of col label. using '_' separator. Not an empty string. Remove

- data.columns = ['_'.join(filter(None, col)).strip() for col in data.columns]

this Remove any leading or trailing spaces.

- mean_squared_error :- Measure the average of the Sequence of errors. It indicates How close the predicting are to the actual values.

- R-Squared (R^2) :- It tells How well the model explains the variance in the target Variable. An R^2 value closer to 1 means better fit.

- Input Feature (X_train) & Relation b/w target Variable (y_train) & linear Regression.

- Linear Regression :- One of the most commonly Used statical methods for predictive modelling.
  1) Simplicity & interpretability.
  2) Quick to train - is inexpensive.
  3) Works well for linear Relationship.
  4) Baseline :- Good starting point
  5) Easy to evaluate :- provides R^2 & mean square Error

wide

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

feature price

- plt.scatter(range(len(y_test)), y_test, label="Actual prices", color="blue", alpha=0.6)

scatter plot

transparency of point.

Range(len(y_test)) creates an array of Value from 0 to the length of the y_test array, Representing each data point on the x-axis

- plt.legend() - is display legend on the plot price(blue) & predicted (Red).

- ffill :- Fillna :- used to fill missing (NaN) Values in the DataFrame.

ffillna :- ffill :- Forward fill :- fill missing value with the last known value, Assume that previous day's closing price is similar to current days.

- bfill :- backward fill :- Any Remaining missing values at the start are filled by the sequence data.

NN → Neural Network.

* Supervised Learning :-
  - trained on labeled data. includes both input feature & corresponing correct output
  - The model learn to map the input to the correct output by identifying patterns in the data

* Tensorflow :- open-source Machine learning framework developed by google.
  - also for Neural Network used
  - tensorflow.keras → high-level API with tensorflow.
  - Sequential :- is class used to create a linear stack of a NN.
    - i) Dense layer :- every layer connect to the previous layer
    - Used in classification & Regression.

* classification :- Supervised learning task where the goal is predict a category or class label.

Ex :- Email spam, Image Recognition, Medical diagnosis.

i) Binary classification :- two classes.
ii) Multi class " :- three or more
iii) Multi label " :- multiple class.
iv) Desicion boundry :- line or surface separate diff class feature

Common Algorithm for classification :-
1) logistic Regression.
2) Desicion tree
3) Random forest
4) Support Vector Machines
5) K-NearestNeighbour
6) Neural Network.

* overfitting :- perform well on training data but poorly on new data
* Underfitting :- Model is simple to capture complexity of data.

(O|I)
output.

Minimize the error by adjusting the weights in n/w.

* LSTM :- Long Short team Memory :-

(RNN) Recurrent Neural Network

↳ for sequential data.
discard info ← New info stored in cell, output delete.
↳ special gates :- forget, input, output.

↳ Regular RNN struggles to Capture long-term dependenci due to Vanishing gradient problem.

* Vanishing gradient problem :- & O/I
Back propagation: calculating the error. (diffⁿ b/w predicted

subtracts Current Value from the previous val → for diffⁿ b/w consecutive data pair in series,
* delta = series. diff() price Change Represent.

→ Index clean, sequence order
* data. reset_Index () :- for past data ← to avoid out-of-bounds indexing.

* for i in range (lookback, len(scaled_data)):
x.

i → Represent the Current Time step.

as Numpy. unscaled closing prices — skip 1 lookback.

* actual_close = data ['close'] . values [lookback:]
→ beginning (index 0)
X_train = x [: train_size]
↳ ends at train_size - 1
Unscaled closing prices corresponding test
Actual_close_test = actual_close [train_size:]

* LSTM :-
64 Neurons (Unit) — layer output the entire seq. (one output per time step) rather than final time step.
model. add (LSTM (64, return_sequence = true,
input_shape = (X_train.shape [1], X train_shape [2])) Number of time steps in each sequen.
Numb. of feature

→ 20% Randomly drop to zero during training.
Dropout (0.2) → overfitting by Reducing.

* model. compile (optimizer = 'adam', loss = 'mean_square_error')
optimizer :- Model Updates its weight during training to Minimize the loss function.
Adam' :- Adaptive Moment Estimation :.
↳ It Adjust the learning Rate for each Weight dynamically, making faster & more efficient for wide Range of problems.

* Gradient descent :- optimization algo Used to train ML by minimizing the loss fun.

→ loss fun :- How far the predicted value from the Actual Rute. Smaill loss, better fit model.

* model.fit :- train your model

* epochs :- one Complete pass through the entire training dataset
ex :- epochs 10 = Model go troy the data 10 time
↳ too Many epochs overfitting.

* Verbose :- Control the Amount of info display during training :-

0 : No output.
1 : display a progress bar for each epoches (default)
2 : " " one line per epoches without Progress bar.

* np.zeros() → create a Matrix filled with zeros

* len(y_pred_scaled) :- len() function in python is Used to determine the number of element in an object, Such as a list, tuple, Num string, dictionary, iterable. Total of elements

* scaler.inverse_transform() :- Take a Scaled dataset & ~~convert~~ converts it back to its original scale.

↳ interpret the Result, we need to invert the transformation and bring the prediction back to their original scale.

* Mean Square Error [MSE] : Actual values & predicted values compute.

* ~~Spare~~ sqrt(mse) :- Root Mean square Error (RMSE)
↳ provides more interpretable Measure of Error compared to MSE.

* R^2 = coefficient of determination
  - How well a model's prediction explain the variability in the target variable.

  $R^2 = 1$ → Model is best. perfectly matches the actual values.
  $R^2 = 0$ → Model is average.
  $R^2 = -1$ or negative → Worst the situation.

* print(f"Test MSE : {mse:.4f}") → formatting string
  formatting string
  date starting from the certain point. ← skip past data used in training model, so dates starting from testing begins.

* plt.plot(data['date'][lookback + train_size:], actual_close_test, label='Actual close', color)

* plt.legend() :- explain which line for actual stock prices & which is for predicted stock prices.

* plt.tight_layout() :- Adjust the layout of the plot so that every thing fits nicely without overlapping.

* pred_inverted = Scaler.inverse_tranform(placeholder) holder
  - inverse the scalling of placeholder Array.
  - placeholder is passed into inverse_tranform so that the predicted "close" value can be converted back to original price Range.

* threshold = 0.02 :- 2%
  - prediction needs to be 2% higher or lower.

* zip() → pairs each element of actual close test price & pred_close.

* MinMaxScaler() → data scalling Technique.
  Range b/w 0 & 1.
  - Used that Makesure that all the feature in dataset are on same scale.

* Dense() :- Add a Dense layer with 1 unit to the model.
  - every Neuron in this layer connect to the previous layer.
  - 1 output Neuron. → Used for Regression.

Random forest = RF

* enumerate() :- like in list or Array, it get both the index & value.
  → telling a position.
* .iloc[] → Used for index or slicing.
* .values → used to extract the data as Numpy.
* gca() → Get Current Axes, where data plot.
* param_grid :- list of possible Values to the hyper parameter. → Random forest model.

1) n_estimators :- Num of trees in RF. More trees lead to better performance but also model slower.

* Rolling Window 20 :- 30 days are better for performance in Months. for Small term investment ; it's ideal.

* classification :- categories outcomes.

* Regression :- Continuous outcome.

* Support :- level where the trends to stop falling and bounce back up.
  ↳ 100 $ fall price drop & stop falling that means 100 $. → support level.

* Resistances :- stop Raising. 100 Highest Range & stop Raising.

* $R^2 = 1 - \frac{SS_{res}}{SS_{tot}}$ → total sum of square → dif^n b/w actual price & the mean of Actual price.

Sum of squared residual (dif^n b/w Actual & predicted prices).

* What is hyper parameter Tuning ?
  ↳ process of selecting the optimal Values for hyperparameters in a ML Model to improve its performance.

* Reinforcement learning :-
  ↳ interacting with environment & Receiving feedback through Rewards or punishment. trial & Error where learn from outcomes from action.

* clustering :- used in Unsupervised learning where the group of a set of objects are data points into cluster (groups) based on their Similarity.

* cluster :- Simply a collection of data points that are Similar to each other.

* k-cluster :- k number of clusters.

* why lstm over LR ?

* Handling Sequential Data :-
1) stock data is Sequential.
2) LSTM is designed for Sequential data. Types of RNN. → specifically design for Sequential data, & learn pattern over time & Capture long-term dependencies.
3) complex & Non-linear Relationship
3) Learning from the past.
4) performance & Accuracy.

* CNN (Convolutional Neural Network)
is type of Artifical Neural Net. Analyzed Visual data, image & Videos. widely used for image classification, object detection & more.

* Convolution layer :- Edges & shape, Pattern identify of Image.

* padding :- Adding Extra layer pixel, or Rows.
Valid " :- No padding. output more smaller than input.
Same " = Same Size I/O.
Down Sampling operation.

* ~~Max~~ pooling :- Reduce the Matrix dimention & Noise Reduce.

* Fully Connected :- Flattening & dense layer

* optimizer :- An algorithm or Method used to Adjust the parameter (weights & bias) of a ML Model during training in order to minimize the loss function.

Math operation help to identify pattern in image pattern texture.

Minimize the loss function.

* Loss function :- In ML & optimizer to Measure how well a model's prediction diff with Actual ~~target~~ target Value.
  - training a model is to Minimize this loss, thereby improving model's Accuracy.
  - For Example :- scorecard → how far your prediction from the Current answer.
  - It gives Number (called loss) that shows how bad your model's prediction Are.

* Types of optimizer :-

1) Gradient Descent - Based optimizer :-
   - It looks at all the Data at once & make a Single update to Improve the Model.
   - Good for small dataset & Bad for huge. (too slow)

2) stochastic Gradient Descent (SGD)
   - Instead of looking at all data, it looks at one piece at a time & update the model after every piece.

3) Mini - Batch Gradient Descent :-
   - It's Mix of the two. It look at small chunks of the data (Mini - batches) & update the Model after each chunk.
   - Balancing speed & Accuracy.

* smart optimizers (Adaptive)
  - these optimizers automatically Adjust how much they change Model's weight based on how thing are going.

• Momentum :-
  - As the ball rolls, it build speed.
  - Even if it hits a small bump (like more or minor object), it keeps moving forward instead of stopping.
  - It Remember the past updates & adds a fraction of that 'Extra push' to Current steps.

After computing the average error across the entire dataset

* RMSprop (Root Mean Square propagation)
- Monitors how much each weight is changing during the training:
  i) if a weight keeps changing alot, it slow down updates for that weight so it doesn't overshoot.
  ii) if another weight isn't changing much, it allows larger update to make progress faster.

b) Learning Smoother & more stable, for data that changes over time.

c) In time-series data tends to change over time. RMSprop's ability to adjust learning Rate dynamically for each weight helps it learn efficiently in Situation where data constantly evolving.

☆ Learning Rate:- Control how big or small the steps are when the Model updates itself to minimize error (Loss).

* Adam (Adaptive moment Estimation)

b) Smart helper that combines the benefit of two techniques (Momentum & RMSprop) to help learn model faster and more effectively.

i) Momentum:- The average of past gradients. (like the direction of the model has been moving in).

ii) Second moment (RMSprop):- the variance or spread of gradients. (How much the gradients are fluctuating.

for i) Adam Remember the direction based on previous gradients & pushes the learning in the same direction.

ii) Adjust the learning Rate for each parameter if one parameter is fluctuating alot, Adam will make smaller adjustment. if Another is stable they will make bigger updates.

Error :- Singel Pre. Val - Act. Val.

Loss :- Average of total Errors.

* learning Rate :- Step Size when you are trying to find Answer. It Control how much the model Adjust its guess during training.

- Control How much the Model "learns" on "Update" After each training step.
- Small Learning Rate takes a time, but More Accurate.
- Large learning Rate faster but may miss the best Solution.

* MSE :- How Far the prediction from the Actual price. it gives you a Single number that tells you how "wrong" your predictions on Average.

* Error :- is diff" blw pre. Val. & Act. Val.
Errors = prediction - Actual

LSTM Basics :-

1) tanh Activation fun :- input Values into a Range blw -1 & 1. for Candidate value & scaling outputs.

2) Sigmoid :- input Values into a 0 & 1. Used for gates :- input, output, forget.

3) Concatenation Combines two Vector by stacking their elements together end to end to form a Single Vector.

4) x → point wise Multiplication :- Forget state Ex :- [1, 0, 1] they forget 0.

+ → Point wise Addition :- Combine info, New info to cell state.

* Using bias to improve flexibility.

* Minimal Tuning :- Model with Minimal tunning Requirements are convenient because they perform Reasonably well with default setting. data is Noisy & change dynamically, so Using default setting saves effort while Archiving good results. Ensure less effort Adjusting settings, more time focus on Result

script :-

1) After build powerful LSTM model - Capable of processing sequential data for stock prices, for further How we can ensure that this model learns efficiently & How it make Smarter with evaluate?

2) for Navigator of the Machine learning we use optimizer. it guide our model to Minimize the error by adjusting the internal's parameter like weights & bias.

* X_train [shape [1]] :- Number of time steps. For Example time-series dataset, Represent days as Minutes.

* X_train [shape [2]] :- The Number of feature at each time steps.

(1000) (80) (5)

input_shape :- (X_train.shape[1], X_train.shape[2]) :- Timesteps, Feature

why it best :-

1) Adam :- Adaptive Learning Rate
- Works well with LSTM.
- Efficient & Adiable combination

2) Mean square Error :-
1) Capture Magnitude of Error.
2) Continuous output :- Numerical, so it Natural choice for Measuring performance.

3) Measure the Average squared diff^n b/w the Actual Value & predicted value.

4) Lower Value Mean prediction is closed to the A.V
5) How much the error or diff^n b/w p.v & A.v