Walk-Forward Validation Runner and Analysis

This notebook executes a full Walk-Forward Validation pipeline and analyzes the aggregated out-of-sample results. This provides a more robust estimate of the strategy's performance across different market regimes, serving as the ultimate test of its viability.

In [1]:
```
import sys
import os

# Get the absolute path of the project's root directory
# os.getcwd() gets the current folder ('/notebooks')
# os.path.join(..., '..') goes one level up to the project root
project_root = os.path.abspath(os.path.join(os.getcwd(), '..'))

# Add the project root to the Python path
if project_root not in sys.path:
    sys.path.append(project_root)

# --- Imports and Setup ---
import yaml
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.metrics import classification_report

# Import custom project modules
from src.training_pipeline import TrainingPipeline
from src.backtester import VectorizedBacktester
from src.utils import plot_confusion_matrix, plot_roc_curve

# --- Load Configuration ---
# Construct the absolute path to the config file using project_root
config_path = os.path.join(project_root, 'configs', 'config.yaml')
print(f"Loading configuration from: {config_path}")
with open(config_path, 'r') as file:
    config = yaml.safe_load(file)
print("Configuration loaded successfully.")
```

```
Loading configuration from: C:\Projetos_Python\gld_lstm_strategy\configs\config.yaml
Configuration loaded successfully.
```

In [2]:
```
# Create an instance of the training pipeline, passing the project_root
pipeline = TrainingPipeline(config=config, project_root=project_root)

# Run the entire walk-forward validation process.
# WARNING: This will take a significant amount of time to run,
# as it trains and evaluates the model 5 times.
wf_results = pipeline.run_walk_forward(n_splits=5)

print("\n\n✅ --- Walk-Forward Validation Finished! --- ✅")
print("Aggregated results are now available in the 'wf_results' variable.")
```

```
Random seeds set to 2025 for reproducibility.


===== Starting WALK-FORWARD VALIDATION for model_type='lstm' =====
===== Step 1: Loading and Preparing Full Dataset =====
--- Loading Main Asset Data ---
Loading GLD data from local cache: C:\Projetos_Python\gld_lstm_strategy\data\gld_data.csv

--- Loading Macroeconomic Data ---
Loading DX-Y.NYB data from local cache: C:\Projetos_Python\gld_lstm_strategy\data\dx-y.nyb_data.csv
Loading ^TNX data from local cache: C:\Projetos_Python\gld_lstm_strategy\data\^tnx_data.csv
Loading ^VIX data from local cache: C:\Projetos_Python\gld_lstm_strategy\data\^vix_data.csv
Loading CL=F data from local cache: C:\Projetos_Python\gld_lstm_strategy\data\cl=f_data.csv
Loading SI=F data from local cache: C:\Projetos_Python\gld_lstm_strategy\data\si=f_data.csv
Loading TIP data from local cache: C:\Projetos_Python\gld_lstm_strategy\data\tip_data.csv
Loading HG=F data from local cache: C:\Projetos_Python\gld_lstm_strategy\data\hg=f_data.csv

===== Starting Feature Engineering Pipeline =====
Step 1: Creating custom OHLCV features...
Step 2: Applying 'All' technical indicator strategy from pandas_ta...
```

```
130it [00:16,  8.01it/s]
```

```
 -> Dropped 2 redundant TA columns.
Step 3: Creating custom interaction and ratio features...
Step 4: Creating lagged and momentum features...
Step 5: Merging macroeconomic features...
 -> Macro features merged and forward-filled.
Step 6: Defining target variable...

Pipeline complete. Dropped 77 rows with NaN values.
Final dataset shape: (2438, 247)
===============================================


==================== FOLD 1/5 ====================
Training on 408 samples, testing on 406 samples.
--- Running Feature Selection for this fold ---
Selected 21 features for fold 1.
--- Building and Training LSTM Model for this fold ---
7/7 ━━━━━━━━━━━━━━━━━━━━ 2s 159ms/step

==================== FOLD 2/5 ====================
Training on 814 samples, testing on 406 samples.
--- Running Feature Selection for this fold ---
Selected 34 features for fold 2.
--- Building and Training LSTM Model for this fold ---
7/7 ━━━━━━━━━━━━━━━━━━━━ 7s 1s/step

==================== FOLD 3/5 ====================
Training on 1220 samples, testing on 406 samples.
--- Running Feature Selection for this fold ---
Selected 33 features for fold 3.
--- Building and Training LSTM Model for this fold ---
WARNING:tensorflow:5 out of the last 15 calls to <function TensorFlowTrainer.make_predict_function.<locals>.one_step_on_data_di
stributed at 0x0000023E1D767D00> triggered tf.function retracing. Tracing is expensive and the excessive number of tracings cou
ld be due to (1) creating @tf.function repeatedly in a loop, (2) passing tensors with different shapes, (3) passing Python obje
cts instead of tensors. For (1), please define your @tf.function outside of the loop. For (2), @tf.function has reduce_retracin
g=True option that can avoid unnecessary retracing. For (3), please refer to https://www.tensorflow.org/guide/function#controll
ing_retracing and https://www.tensorflow.org/api_docs/python/tf/function for  more details.
7/7 ━━━━━━━━━━━━━━━━━━━━ 5s 423ms/step

==================== FOLD 4/5 ====================
Training on 1626 samples, testing on 406 samples.
```

```
--- Running Feature Selection for this fold ---
Selected 36 features for fold 4.
--- Building and Training LSTM Model for this fold ---
WARNING:tensorflow:5 out of the last 15 calls to <function TensorFlowTrainer.make_predict_function.<locals>.one_step_on_data_distributed at 0x0000023E27A56680> triggered tf.function retracing. Tracing is expensive and the excessive number of tracings could be due to (1) creating @tf.function repeatedly in a loop, (2) passing tensors with different shapes, (3) passing Python objects instead of tensors. For (1), please define your @tf.function outside of the loop. For (2), @tf.function has reduce_retracing=True option that can avoid unnecessary retracing. For (3), please refer to https://www.tensorflow.org/guide/function#controlling_retracing and https://www.tensorflow.org/api_docs/python/tf/function for  more details.
7/7 ━━━━━━━━━━━━━━━━━━━━ 5s 437ms/step

==================== FOLD 5/5 ====================
Training on 2032 samples, testing on 406 samples.
--- Running Feature Selection for this fold ---
Selected 35 features for fold 5.
--- Building and Training LSTM Model for this fold ---
7/7 ━━━━━━━━━━━━━━━━━━━━ 2s 176ms/step

==================== Walk-Forward Validation Finished ====================


✅ --- Walk-Forward Validation Finished! --- ✅
Aggregated results are now available in the 'wf_results' variable.
```

1. Aggregated Statistical Performance Analysis

Here we analyze the combined performance across all out-of-sample folds. This gives us a single, robust view of the model's predictive power over time.

In [3]:
```
print("Analyzing aggregated out-of-sample results...")

# Extract the combined results from all folds
true_labels = wf_results['true_labels']
pred_probas = wf_results['pred_probas']
binary_preds = (pred_probas > 0.5).astype(int)

# --- Display Overall Metrics ---
print("\n--- Overall Classification Report (All Out-of-Sample Folds) ---")
print(classification_report(true_labels, binary_preds))
```

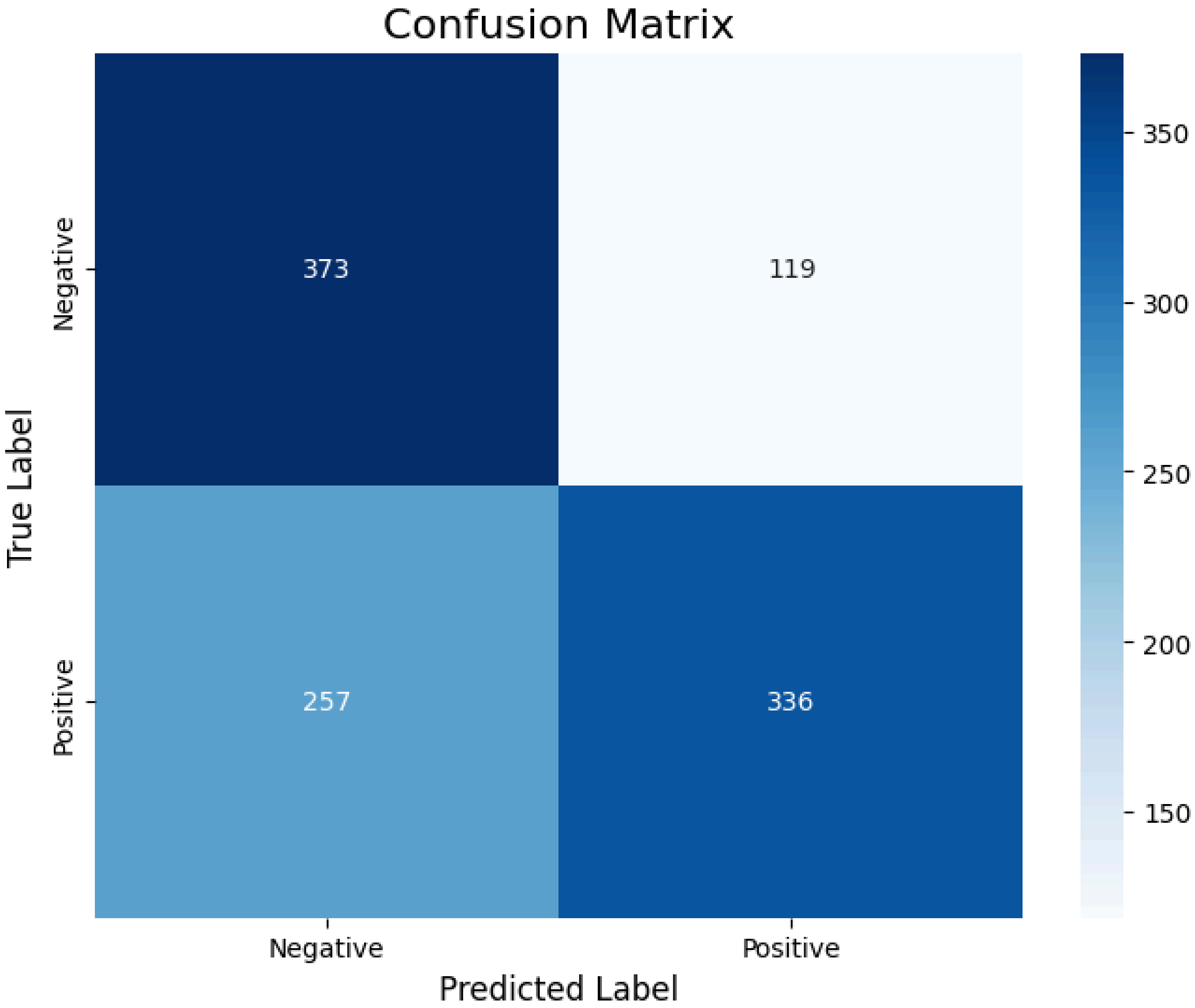
```python
# --- Plot Overall Evaluation Graphs ---
plot_confusion_matrix(true_labels, binary_preds)
plot_roc_curve(true_labels, pred_probas, "Walk-Forward Aggregated")
```

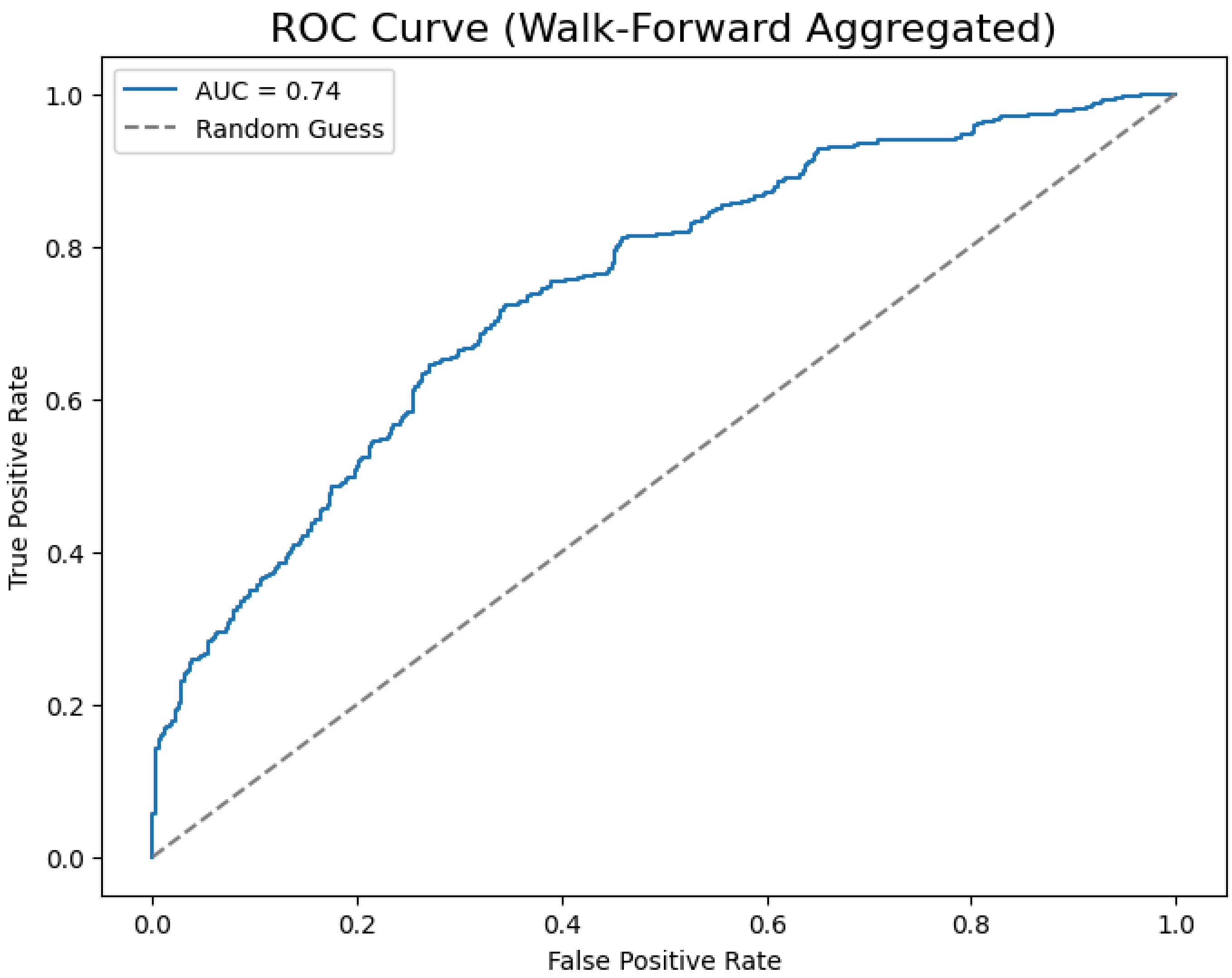
```
Analyzing aggregated out-of-sample results...

--- Overall Classification Report (All Out-of-Sample Folds) ---
              precision    recall  f1-score   support

           0       0.59      0.76      0.66       492
           1       0.74      0.57      0.64       593

    accuracy                           0.65      1085
   macro avg       0.67      0.66      0.65      1085
weighted avg       0.67      0.65      0.65      1085
```


Confusion Matrix
True Label
Negative
Positive
373
119
257
336
Negative
Positive
Predicted Label
350
300
250
200
150



2. Aggregated Backtest Performance

This is the final and most important test. We combine the out-of-sample signals from all folds into a single continuous timeline and run one final backtest. This simulates how the strategy would have performed in a real-world scenario where the model is periodically retrained.

In [4]:
```
# Extract the combined backtest dataframe from the results
backtest_df = wf_results['backtest_df']

print(f"Aggregated backtest will run on {len(backtest_df)} signals.")
print(f"Covering the period from {backtest_df.index.min().date()} to {backtest_df.index.max().date()}.")

# Instantiate and run the backtester on the combined out-of-sample signals
backtester = VectorizedBacktester(
    price_data=backtest_df,
    signals=backtest_df['signal'],
    config=config
)

# Run the backtest with exit signals enabled
portfolio = backtester.run(commission=0.001, slippage=0.001)
```

```
Aggregated backtest will run on 1085 signals.
Covering the period from 2017-09-06 to 2024-12-30.

===== Starting Vectorized Backtest with vectorbt =====

--- Backtest Performance Stats ---
Start                          2017-09-06 00:00:00
End                            2024-12-30 00:00:00
Period                          1085 days 00:00:00
Start Value                                  100.0
End Value                               298.878906
Total Return [%]                        198.878906
Benchmark Return [%]                     89.756336
Max Gross Exposure [%]                       100.0
Total Fees Paid                            11.7292
Max Drawdown [%]                          5.756067
Max Drawdown Duration             37 days 00:00:00
Total Trades                                    30
Total Closed Trades                             30
Total Open Trades                                0
Open Trade PnL                                 0.0
Win Rate [%]                             83.333333
Best Trade [%]                           20.154251
Worst Trade [%]                          -0.988026
Avg Winning Trade [%]                     4.686832
Avg Losing Trade [%]                     -0.565897
Avg Winning Trade Duration        17 days 08:38:24
Avg Losing Trade Duration          4 days 00:00:00
Profit Factor                            34.295711
Expectancy                                6.629297
Sharpe Ratio                               3.10111
Calmar Ratio                              7.736256
Omega Ratio                               2.065786
Sortino Ratio                             6.812991
dtype: object

--- Plotting Equity Curve and Drawdowns ---
```

```
===== Backtest Finished =====
```