Report on Financial Data Analysis and Trading Strategy

1. Data Collection and Cleaning (5%)

The data is collected from Yahoo finance given the 10 tickers in the provided tickers.csv file. It is directly downloaded within the timeframe given, '2000-01-01' to '2021-11-12.' As the data is downloaded, empty values are replaced with 0 and the data is forward filled.

The preprocess function is designed to read, process, and save stock market data from a CSV file. It performs several key operations to prepare the data for further analysis or modeling. Here's a breakdown of its functionality:

- Reading the CSV File: The function begins by reading data from a CSV file located at the file_path. It loads this data into a Pandas DataFrame, which is a 2D labeled data structure with columns of potentially different types.
- Date Parsing: If enable_date_parsing is set to True, the function parses the column specified by date_column as dates. It uses the format specified by date_format to correctly interpret the date strings. If there are any errors in parsing (like incorrect format), those dates are replaced with NaT (Not a Time), which is Pandas' way of indicating missing or invalid dates.
- Sorting by Date: The DataFrame is then sorted based on the date column. This step is crucial for time series analysis as it ensures that the data is in chronological order.

2. Features Used in the Model (5%)

The preprocess function also creates a new column named 'target' in the DataFrame. This column is a binary indicator where each entry is 1 if the closing price of the next day (closing_price_column.shift(-1)) is greater than the closing price of the current day, and 0 otherwise.

The new_features class performs various technical analysis calculations on a given DataFrame. Here's a detailed explanation of its functionality:

Calculate moving averages: This method calculates moving averages for the closing prices over specified time windows. The window_list parameter is a list of integers representing the number of days for each moving average. For each window in window_list, a new column is created in the DataFrame with the moving average over that window.

Calculate Relative Strength Index (RSI) (calculate_rsi): This method calculates the RSI, a popular momentum indicator. RSI is calculated using the formula based on the average gains and losses over a 14-day window.

Calculate Moving Average Convergence Divergence (MACD) (calculate_macd): This method computes the MACD, another momentum indicator. It involves calculating two exponential moving averages (EMAs) over different spans (short-term and long-term), and then finding their difference.

Process Data (process): Resets the index of the DataFrame and then sets the date column as the new index. Calls the methods to calculate the moving averages, RSI, and MACD. Extracts the calculated indicators (moving averages, RSI, and MACD) along with the adjusted closing prices into separate DataFrames. Concatenates these DataFrames into a single DataFrame (combined_indicators_df), aligning them by the index (date). Drops rows with missing values in key columns to ensure data completeness.

Overall Functionality: The class serves as a tool for adding technical indicators to stock market data, which are often used in trading strategy development and financial analysis. It modularizes the process of calculating common technical indicators, making the code reusable and maintainable.

3. Model Selection and Expectations (10%)
I chose to use Logistic Regression and Random Forests. The choice of these algorithms was guided by their unique capabilities in dealing with classification challenges. My approach, as delineated in the provided code, involved partitioning the data, normalizing feature sets, and fine-tuning the algorithms with specific hyperparameters.

Logistic Regression:

Advantages: Known for its simplicity and efficiency, Logistic Regression excels in binary classification. Its linear nature makes it highly interpretable.

Performance Projections: Expected to deliver robust results with datasets where the predictors and the response have a linear correlation. Its strength lies in elucidating the effect of individual predictors on the response, though its performance might wane with complex, non-linear relationships.

Key Parameters: Regularization strength (C with values [0.1, 1, 10]), penalty norm (['l1', 'l2']), solver algorithm (['liblinear', 'saga']), and iteration limit (max_iter set to 100).

Objective: The focus here is on curtailing overfitting while achieving a balance in model bias and variance, crucial for effective generalization.

Random Forest Classifier:

Advantages: This method stands out for its ability to process complex, non-linear datasets and handle a myriad of features without succumbing to overfitting.

Performance Projections: Anticipated to uncover intricate patterns within the data, Random Forest is poised for high accuracy, leveraging its ensemble approach to reduce error susceptibility.

Key Parameters: Number of trees (n_estimators with values [50, 100, 150]) and tree depth (max_depth with options [3, 5, 7]).

Objective: These parameters are tuned to find an equilibrium between model complexity, accuracy, and overfitting risk.

4. Trading Reports and Analysis (5%)

The trading reports are in the reports folder, there is one report for each model per top ten stocks.

Overall Logistic Regression outperformed Random Forests in terms of trading strategies. Logistic regression was able to get higher sharpe ratios, average returns and max profit.

5. Conclusion and Model Comparison (10%)
the Logistic Regression and Random Forest classifiers. Their performance was intriguingly similar across most metrics. However, the Logistic Regression model edged out slightly in

profitability, demonstrating a more consistent and less volatile performance. Notably, the worst-case scenario under the Random Forest strategy led to a significant 44.6% loss in brokerage value, a downside not observed with Logistic Regression.

Despite some profitable trades, the overall accuracy of these models was moderate and their predictions highly correlated. This poses a challenge for all but the most risk-tolerant investors, as the observed profits were often offset by trading commissions, typically resulting in a marginal decline in account value over the period, barring a few exceptional cases.

6. Additional Considerations (10%)

The current implementation of these models, while having its limitations, shows no signs of overfitting. This is evidenced by the modest predictive accuracy, which did not exceed 60% for any stock in our analysis. The rigorous approach, including exhaustive grid search cross-validation and standard scaling, especially fortifies the Random Forest model against overfitting, a characteristic inherent to its design.