**1. Project Overview**

* **Project Title:** Forecasting Tool for Swing Trading
* **Project Description:**
  + Project Goals & Objectives
    - The goal of this project is to create a stock forecasting web application specifically designed for short-term predictions. This tool aims to assist swing traders in making informed decisions by providing accurate and timely forecasts of stock movements. The objectives include developing a user-friendly interface, integrating reliable data sources, and employing advanced algorithms to predict stock price trends effectively.
  + Problem Statement
    - The primary problem this project seeks to address is the challenge faced by swing traders in accurately predicting short-term stock price movements. Swing trading requires making quick and informed decisions based on price trends, yet many traders struggle with the lack of reliable tools that offer precise short-term forecasts. Traditional stock analysis tools may not provide the granularity or timeliness needed for effective swing trading. By developing this forecasting tool, we aim to bridge this gap and equip traders with a robust, data-driven application that enhances their trading strategies and decision-making processes.
  + Key Research Questions
    - How can we develop a forecasting tool that provides accurate short-term stock price predictions?
    - What advanced algorithms are most effective for predicting stock price trends in swing trading?
    - How can we ensure the reliability and timeliness of the data sources integrated into the tool?
    - What features are essential for a user-friendly interface tailored to swing traders?
    - How can the forecasting tool enhance the decision-making processes of swing traders?
* **Project Timeline:**
  + 2024.12.04 – 2025.02.04
* **Team Members:**
  + Shane Peterson – Machine Learning Engineer

**2. Data**

* **Data Sources:**
  + Where did the data come from?
    - For updated symbols, we will utilize Finnhub, which provides real-time data and comprehensive financial information. In addition, yfinance will be employed for retrieving daily ticker information, ensuring we have the latest and most accurate market data available.
* **Data Description:**
  + Data dictionary:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Variable Name | Description | Data Type | Units | Range/Allowable Values |
| Date | The date of the trading day | DatetimeIndex | N/A | Excludes Weekends and Holidays |
| Open | The price of the stock at the beginning of the trading day | Float64 | Relevant Currency | Non-negative numerical values |
| High | The highest price reached by the stock during the trading day. | Float64 | Relevant Currency | Non-negative numerical values |
| Low | The lowest price reached by the stock during the trading day. | Float64 | Relevant Currency | Non-negative numerical values |
| Close | The price of the stock at the end of the trading day | Float64 | Relevant Currency | Non-negative numerical values |
| Volume | The number of shares or contracts traded during a trading day | Int64 | Shares (for stocks), Contracts (for options), or other relevant units depending on the asset. | Non-negative numerical values |
| Dividends | The value represents the dividend amount per share. | Float64 | Currency per Share | Non-negative numerical values |
| Stock Splits | Corporate actions where a company increases the number of its outstanding shares by issuing additional shares to existing shareholders | Float64 | Ratio | Non-negative numerical values |
| Capital Gains | Profits earned from the sale of a capital asset | Float64 | Relevant Currency | Non-negative numerical values |

* **Data cleaning and preprocessing steps:**

1. **Data Source**
   1. The data for this project is obtained from reliable financial APIs (Finnhub and yfinance), which typically provide clean and well-formatted data.
   2. Data Cleaning:
      1. Minimal cleaning required: Given the source of the data, minimal data cleaning is necessary.
      2. Data checks: Basic checks for data types, missing values, and outliers were performed.
2. **Feature Engineering**
   1. Percentage Change
      1. Description: This feature represents the percentage change in the closing price from the previous day's close.
      2. Formula: (Close\_Today - Close\_Yesterday) / Close\_Yesterday (Implemented in the code as data.Close.pct\_change() and assigned to the column named 'daily\_returns')
      3. Purpose: Captures the relative price movement and is used to assess volatility and returns.
   2. Bollinger Bands
      1. Description: These bands consist of three lines:
         1. Middle Band: A simple moving average (SMA) of the closing price over 20 days
         2. Upper Band: The middle band plus 2 standard deviations.
         3. Lower Band: The middle band minus 2 standard deviations.
      2. Calculation:
         1. Middle Band: Calculated using ‘ta.volatility.BollingerBands(close=data['Close'], window=20, window\_dev=2)’.
         2. Upper Band: Calculated using ‘indicator\_bb.bollinger\_hband()’.
         3. Lower Band: Calculated using ‘indicator\_bb.bollinger\_lband()’.
      3. Purpose: Visualizes price volatility and potential overbought/oversold conditions.
   3. 50-Day Simple Moving Average
      1. Description: The average of the closing prices over the previous 50 trading days.
      2. Calculation: Implemented in the code as ‘data['Close'].rolling(window=50).mean()’.
      3. Purpose: Acts as a trend indicator, smoothing out short-term fluctuations.
   4. Annualized Volatility:
      1. Description: This feature captures the overall volatility of the closing price over a year.
      2. Calculation: The code calculates the standard deviation of the daily returns and annualizes it by multiplying by the square root of 252 (assuming 252 trading days in a year). Then it assigns a category label ("Low", "Medium-Low", etc.) based on the volatility range.
      3. Purpose: Provides a quick estimate of the stock's overall riskiness.
   5. yhat:
      1. Description: This column represents the forecasted value for the time series at each point in the future.
      2. Creation: Generated by the Prophet model based on the fitted trend, seasonality, and other model parameters.
      3. Purpose: Provides the central prediction of the time series at future time points.
   6. yhat\_lower:
      1. Description: The lower bound of the prediction interval for the forecast.
      2. Creation: Calculated by the Prophet model to quantify the uncertainty associated with the forecast.
      3. Purpose: Defines the lower limit of a range of plausible future values, capturing the inherent uncertainty in the forecasting process.
   7. yhat\_upper:
      1. Description: The upper bound of the prediction interval for the forecast.
      2. Creation: Calculated by the Prophet model to quantify the uncertainty associated with the forecast.
      3. Purpose: Defines the upper limit of a range of plausible future values, capturing the inherent uncertainty in the forecasting process.
   8. Trend:
      1. Description: The estimated trend component of the time series.
      2. Creation: Extracted by the Prophet model to isolate the long-term growth or decline in the time series.
      3. Purpose: Provides insights into the overall direction and magnitude of change in the time series over time.
   9. Seasonal Components (e.g., Yearly, Weekly, Daily)
      1. Description: These columns represent the seasonal patterns identified by the Prophet model.
      2. Creation: Extracted by the Prophet model to capture recurring patterns within the time series at specific time intervals (e.g., annually, weekly, daily).
      3. Purpose: Allows for the identification and quantification of recurring patterns that influence the time series, improving forecast accuracy.
   10. winsorized (output of winsorizer function)
       1. Description: This column represents the closing price after applying a winsorization technique based on volatility.
       2. Creation: Created by capping extreme values in the Closing price column based on volatility. Stocks with higher volatility have thresholds closer together, while stocks with lower volatility have thresholds further apart.
       3. Purpose: Reduces the impact of outliers on the closing price, stabilizing the data for analysis and modeling.

* **Data Exploration:**
  + 1. Why did I chose SMAPE? (symmetrical mean absolute percentage error)
       1. I chose this metric because it's scale-independent, making it easy to compare across stocks with different price ranges. It penalizes over-forecasting and under-forecasting relatively symmetrically, which is often desirable in financial forecasting.
    2. Performance Findings

|  |  |  |  |
| --- | --- | --- | --- |
| volatility | smape\_naive | smape\_standard | smape\_tuned |
| Low | 2.325 | 0.068 | 0.047 |
| Medium-Low | 9.655 | 0.206 | 0.130 |
| Medium | 10.642 | 0.368 | 0.215 |
| Medium-High | 12178.034 | 0.552 | 0.431 |
| High | 114156.350 | 1.320 | 1.064 |

* + - 1. The tuned model holds practical significance across 1) all volatility categories and 2) both the naïve persistence model and standard models. As you might expect, the accuracy of the tuned model has a negative correlation to volatility, meaning it yields the highest accuracy on low volatility tickers and low accuracy with those with high volatility.
    1. Winsorization
       1. A graph of a number of blue bars

          AI-generated content may be incorrect.
       2. After identifying the volatility of a ticker, the program that tunes that model then decides on thresholds for winsorization. Tickers with lower volatility are given tighter thresholds (0.15 & 0.85) while tickers with higher volatility are given looser thresholds (0.05 & 0.95). These thresholds are then applied to the Close price, smoothening outliers outside of the given thresholds. The goal of this strategy is to increase model accuracy without overfitting. Summarized in the bar graph above, we can confirm that tickers with the lowest accuracy have 58% of their data points smoothed while the most volatile have 28% smoothed.
    2. Statistical Significance.
       1. A Wilcoxon signed-rank test was conducted to compare the SMAPE of the standard Prophet model and the custom Prophet model. The test statistic was W = 1119.0, and the p-value was 2.46-e17. At a significance level of α = 0.05, the results indicate a statistically significant difference in SMAPE between the two models, with the custom Prophet model demonstrating superior performance.
       2. The Cliff's Delta (d) for the difference in SMAPE between the standard Prophet model and the custom Prophet model was 0.69. This demonstrates a strong tendency for the custom Prophet model to produce significantly lower SMAPE values, indicating a substantial improvement in prediction accuracy compared to the standard Prophet model.
    3. Summarize findings in context of project.
       1. Summary
          1. The analysis demonstrates that the custom-tuned model significantly outperforms both the naive persistence and standard Prophet models across all volatility categories, as evidenced by consistently lower SMAPE values. This improvement highlights the practical significance of the tuning process, particularly in mitigating the challenges posed by varying volatility levels. As expected, model accuracy exhibits a negative correlation with volatility, with the highest accuracy observed for low volatility tickers. The winsorization strategy, which adjusts thresholds based on ticker volatility, effectively smooths outliers and contributes to the improved performance without overfitting. This is supported by the increasing percentage of winsorized data points in higher volatility categories, indicating the model's adaptability in handling extreme price fluctuations.
       2. Insights
          1. These findings suggest that a dynamic approach to model tuning, incorporating volatility-based adjustments and winsorization, is crucial for enhancing forecasting accuracy in financial markets. The consistent improvement across diverse volatility regimes underscores the robustness of the custom model. However, the substantial SMAPE values for the naive model in high volatility categories point to the inherent difficulty in predicting such volatile tickers, even with advanced models. Future research could explore more sophisticated techniques for capturing and modeling extreme price movements, potentially further improving accuracy in these challenging scenarios. Additionally, investigating the impact of other factors, such as trading volume or market sentiment, could provide further insights into model performance and refine the tuning process.

**3. Methodology**

* **Model Selection:**
  + A Prophet model is a procedure for forecasting time series data based on an additive model where non-linear trends are fit with yearly, weekly, and daily seasonality, plus holiday effects. It's often chosen for its ability to handle data with strong seasonal patterns and its ease of use, especially for business forecasting. Prophet is particularly well-suited for time series with strong seasonality and readily handles missing data, making it a robust choice for forecasting tasks where these characteristics are prominent, unlike some other models that require more pre-processing or struggle with seasonality.
* **Model Training:**
  + To ensure robust model evaluation and mitigate the risk of overfitting, we employed cross-validation during the training process. Specifically, we partitioned the historical data into multiple folds, iteratively training the model on a subset of these folds and validating its performance on the remaining fold. This approach provided a more reliable estimate of the model's generalization ability compared to a single train-validation split, allowing us to optimize model parameters and assess performance across diverse data segments.
  + Hyperparameter tuning techniques (e.g., grid search, random search)
  + Model evaluation metrics (e.g., accuracy, precision, recall, F1-score, AUC)
* **Model Validation:**
  + Cross-validation techniques (e.g., k-fold cross-validation)
  + Model performance on the validation set
* **Model Deployment (if applicable):**
  + Deployment environment (e.g., cloud, on-premise)
  + Deployment process and considerations

**4. Results and Findings**

* **Model Performance:**
  + Present the final model's performance metrics.
  + Compare the performance of different models (if applicable).
* **Insights and Conclusions:**
  + Key findings and insights from the analysis.
  + How do these findings address the original business or research questions?
  + Limitations of the analysis and potential areas for improvement.

**5. Appendix (Optional)**

* **Code:**
  + Include relevant code snippets or links to a repository (e.g., GitHub).
* **Detailed Technical Notes:**
  + More in-depth information on specific technical aspects.
* **References:**
  + List of any external resources used (papers, articles, libraries).