ML Stock Algo High Overview

1. Define the Problem

- **Objective:** Create a model to classify stock actions into "buy," "sell," and "hold" based on momentum patterns from senator trading data. Use probabilistic outputs to make these classifications.
- **Success Metric:** Achieve high classification performance, e.g., accuracy, precision, recall, or F1 score.

2. Gather and Prepare Data

- **Data Collection:**
  - **Senator Trading Data:** Collect data on stocks bought by senators from sources like OpenSecrets.
  - **Stock Price Data:** Obtain historical stock price data (e.g., from Alpha Vantage, Yahoo Finance).
  - **News Data:** Acquire historical news articles related to the stocks from APIs like News API.
- **Data Sources:** Use Python libraries such as requests or BeautifulSoup, Scrappy for web scraping, or APIs for data collection.
- **Data Cleaning:**
  - Handle missing values and inconsistencies in the trading and price data.
  - Preprocess news articles (tokenization, stemming, etc.) using libraries like NLTK or spaCy.
- **Feature Engineering:**
  - Extract features related to trading volume, stock price changes, and senator trading patterns.
  - Use NLP techniques to derive features from news articles (e.g., sentiment scores, named entities).

3. Choose a Model

- **Select Algorithm:**
  - **Classification Algorithms:** Use classifiers such as Logistic Regression, Random Forest, Gradient Boosting, or XGBoost to predict the probability of "buy," "sell," or "hold."
  - **Neural Networks:** For more complex patterns, consider using models like Multi-Layer Perceptrons (MLP) or Recurrent Neural Networks (RNN) with libraries like TensorFlow or PyTorch.
- **Model Complexity:** Start with simpler models and increase complexity as needed to improve performance.

4. Split the Data

- **Training Set:** Use historical data up to a certain date to train the model.
- **Validation Set:** Use a portion of the data to tune hyperparameters and validate the model.
- **Test Set:** Evaluate the model's performance on a separate, unseen set of data.

5. Train the Model

- **Fit the Model:** Train your chosen classification model on the training data.

- **Hyperparameter Tuning:** Optimize the model's hyperparameters using techniques such as grid search or random search.

6. Evaluate the Model

- **Performance Metrics:** Use classification metrics such as accuracy, precision, recall, F1 score, and ROC AUC.
- **Confusion Matrix:** Analyze the confusion matrix to understand misclassifications.

7. Integrate NLP/NLU

- **NLP Processing:**
    - Use libraries like NLTK, spaCy, or Transformers to process news data.
    - Extract features like sentiment scores, named entities, or topic modeling from news articles.
- **NLU Integration:**
    - Correlate extracted features with stock price movements to improve classification.

8. Deploy the Model

- **Integration:**
    - Develop a Python application or API to integrate the model with real-time or batch data sources.
    - Implement data pipelines for updating stock and news data.
- **Scalability:** Ensure the system can handle large datasets and real-time data if required.

9. Maintain and Update

- **Retraining:** Periodically retrain the model with new data to maintain accuracy.
- **Model Monitoring:** Continuously monitor performance and update features or algorithms as needed.