AI Stocks Project - Model Technical Documentation

Project Overview

The AI Stocks project is a comprehensive stock prediction system that combines multiple machine learning models to provide buy/sell recommendations for AI-related stocks. The system integrates price data, news sentiment analysis, and fundamental metrics to generate actionable trading insights. The project implements five distinct models: a baseline moving average model, two deep learning stock prediction models (MLP and Transformer), and two sentiment analysis models (LSTM and BERT).

Model 1: Baseline Moving Average Model

Input

- **Stock Price Series**: Historical daily OHLCV (Open, High, Low, Close, Volume) data
- **As-of Date**: Reference date for prediction
- **Minimum Data Requirement**: At least 30 days of historical price data

Output

- **Direction Signal**: "up", "down", or "flat" trend classification
- **Confidence Score**: Float value between 0-1 indicating prediction confidence
- **Buy/Sell Recommendations**:
- `should_buy`: Boolean flag (True if trend is "up")
- `should_sell`: Boolean flag (True if trend is "down")
- **Price Targets**:
- `suggested_buy_price`: 2% below latest close price
- `suggested_sell_price`: 5% above latest close price

Example Input

Stock Information:

- **Ticker**: AAPL (Apple Inc.)
- **As-of Date**: 2025-11-19
- **Data Period**: 365 days of historical OHLCV data

Price Metrics:

- Latest close price: **$267.44**
- 10-day moving average: **$268.50**
- 30-day moving average: **$265.20**

Example Output

Prediction Results:

- **Direction**: `"up"` (■■■■)
- **Confidence Score**: `0.75` (75% confidence)

Trading Recommendations:

- **should_buy**: `True` ✓
- **should_sell**: `False` ✗

Price Targets:

- **Suggested Buy Price**: `$262.09` (98% of latest close)
- **Suggested Sell Price**: `$280.81` (105% of latest close)

Technical Flow and Complete End-to-End Flow

1. **Data Fetching**: Retrieve historical price data from Yahoo Finance via `yfinance` library
2. **Caching**: Store price data in JSON cache files (`cache/prices_*.json`) to minimize API calls
3. **Data Conversion**: Convert cached StockPriceSeries to pandas DataFrame with datetime index
4. **Moving Average Calculation**:
    - Compute 10-day short-term moving average (MA_short)
    - Compute 30-day long-term moving average (MA_long)
5. **Trend Detection**:
    - Calculate difference: MA_short - MA_long
    - Normalize by current close price to get relative strength
    - Classify trend:
    - "up" if relative difference > 2%
    - "down" if relative difference < -2%
    - "flat" otherwise
6. **Confidence Calculation**: Map relative strength between moving averages to confidence score (0.3-0.9 range)
7. **Price Recommendation**: Generate suggested buy/sell prices based on trend direction and simple percentage rules
8. **Output Generation**: Return PredictionOutput with direction, confidence, and price targets

Step-by-Step Example: Price Data to Direction Prediction

Input: 365 days AAPL price history, as-of: 2025-11-19

Step 1: Data Conversion

- Convert StockPriceSeries → DataFrame
- Extract close prices: [227.25, ..., 267.44] (365 values)

Step 2: Moving Averages

- 10-day MA: avg([268.50, ..., 267.44]) = **268.50**
- 30-day MA: avg([265.20, ..., 267.44]) = **265.20**

Step 3: Trend Detection

- Difference: 268.50 - 265.20 = **3.30**
- Relative strength: 3.30 / 267.44 = **1.23%**
- Classification: 1.23% > 0% → Direction: **"up"**

Step 4: Confidence Scoring

- Map 1.23% to confidence: **0.75**

Step 5: Price Targets

- Latest close: **$267.44**
- Buy price: 267.44 × 0.98 = **$262.09**
- Sell price: 267.44 × 1.05 = **$280.81**

Output: Direction="up", Confidence=0.75, should_buy=True, Buy=$262.09, Sell=$280.81

Model 2: MLP Stock Prediction Model

Input

- **Stock Price Series**: Historical daily OHLCV data
- **Sentiment Score**: Optional float in [-1, 1] range from news sentiment analysis
- **Fundamental Metrics**: Dictionary containing P/E ratio and P/S ratio
- **Model Configuration**:
- Input dimension: 8 features
- Hidden dimension: 64 (default)
- Number of layers: 2 (default)

Output

- **Direction Classification**: One of three classes:
- Class 0: "down" (■■)
- Class 1: "flat" (■■/■■■)
- Class 2: "up" (■■)
- **Class Probabilities**: Softmax probabilities for each class
- **Confidence**: Maximum probability value
- **Trading Recommendations**:
- `should_buy`: True for "up" predictions
- `should_sell`: True for "down" predictions
- **Price Targets**: Dynamic based on predicted direction and last close price

Example Input

8-Dimensional Feature Vector:

Index	Feature	Value	Description
0	Last Close Price	`267.44`	Most recent closing price
1	10-day MA	`268.50`	Short-term moving average
2	30-day MA	`265.20`	Long-term moving average
3	10-day Std	`2.15`	Short-term volatility
4	30-day Std	`5.80`	Long-term volatility
5	Sentiment Score	`0.35`	News sentiment (positive)
6	P/E Ratio	`28.5`	Price-to-earnings ratio
7	P/S Ratio	`7.2`	Price-to-sales ratio

Feature Vector: `[267.44, 268.50, 265.20, 2.15, 5.80, 0.35, 28.5, 7.2]`

Example Output

Model Prediction:

- **Predicted Class**: `0` (down / ■■)
- **Confidence**: `0.96` (96% confidence)

Class Probabilities Distribution:

- **Down** (Class 0): `96.0%` ■■■■■■■■■■■■■■■■■■■■■■
- **Flat** (Class 1): `0.0%`
- **Up** (Class 2): `4.0%` ■

Trading Recommendations:

- **should_buy**: `False` ✗
- **should_sell**: `True` ✓

Price Targets:

- **Suggested Buy Price**: `$254.07` (95% of $267.44)
- **Suggested Sell Price**: `$262.09` (98% of $267.44)

Model Performance (Test Set):

- **Accuracy**: `40.93%`
- **Precision (Down)**: `0.41`
- **Recall (Down)**: `0.96`
- **F1-Score (Down)**: `0.58`

Technical Flow and Complete End-to-End Flow

1. **Price Data Collection**: Fetch historical OHLCV data from Yahoo Finance (up to 365 days)
2. **Sentiment Score Retrieval**:
    - Fetch news headlines from NewsAPI for the stock
    - Process headlines through LSTM/BERT sentiment models
    - Aggregate sentiment scores to single value in [-1, 1] range
3. **Fundamental Data Fetching**: Extract P/E and P/S ratios from Yahoo Finance financial data
4. **Feature Engineering**:
    - Extract last close price
    - Calculate 10-day and 30-day moving averages
    - Compute 10-day and 30-day standard deviations
    - Incorporate sentiment score (default 0.0 if missing)
    - Add P/E and P/S ratios (default 0.0 if missing)
    - Result: 8-dimensional feature vector
5. **Model Loading**: Load trained MLP model weights from `saved_models/stock_mlp.pth`
6. **Model Architecture**:
    - Input layer: 8 features
    - Hidden layers: Multiple fully connected layers with ReLU activation
    - Output layer: 3-class logits
7. **Model Inference**: Pass feature vector through MLP network to get class logits
8. **Forward Pass**:
    - Pass feature vector through MLP network
    - Apply softmax to logits to get class probabilities
    - Select class with highest probability
9. **Post-Processing**:
    - Map predicted class to direction string (down/flat/up)
    - Generate buy/sell flags based on direction
    - Calculate price targets:
    - "up": buy at 98% of close, sell at 105% of close
    - "down": buy at 95% of close, sell at 98% of close
    - "flat": buy at 99% of close, sell at 101% of close
10. **Output**: Return PredictionOutput with predictions and recommendations

Step-by-Step Example: Feature Vector to Probabilities

Input: Feature vector [267.44, 268.50, 265.20, 2.15, 5.80, 0.35, 28.5, 7.2]
• Features: close, MA10, MA30, std10, std30, sentiment, P/E, P/S
• Shape: (1, 8)

Step 1: Feature Engineering

• Extract 8 features:
• close=267.44, MA10=268.50, MA30=265.20
• std10=2.15, std30=5.80
• sentiment=0.35, P/E=28.5, P/S=7.2

Step 2: MLP Forward Pass

• Layer 1: x[8] × W1[8×64] + b1 → h1[64], ReLU → [0.45, 0.0, 0.78, ...]
• Layer 2: h1[64] × W2[64×64] + b2 → h2[64], ReLU → [0.0, 0.56, 0.34, ...]
• Output: h2[64] × W3[64×3] + b3 → logits = **[2.5, -1.2, -0.8]**

Step 3: Softmax Calculation

• exp([2.5, -1.2, -0.8]) = [12.18, 0.30, 0.45]
• Sum = 12.93
• Probabilities: [12.18/12.93, 0.30/12.93, 0.45/12.93] = **[0.941, 0.023, 0.035]**

Step 4: Post-Processing

• Predicted class: **0 (Down)**, Confidence=**0.941**
• Recommendations: should_buy=False, should_sell=True
• Prices: Buy=$254.07 (95%), Sell=$262.09 (98%)

Output: Class=0, Probs=[0.941, 0.023, 0.035], Buy=$254.07, Sell=$262.09

Model 3: Transformer Stock Prediction Model

Input

- **Stock Price Series**: Historical daily OHLCV data (sequence)
- **Sentiment Score**: Optional float in [-1, 1] range
- **Fundamental Metrics**: Dictionary with P/E and P/S ratios
- **Sequence Length**: Maximum 128 days (default)
- **Model Configuration**:
- d_model: 32 (default)
- Number of attention heads: 4 (default)
- Number of encoder layers: 2 (default)
- Feedforward dimension: 64 (default)

Output

- **Direction Classification**: Three-class prediction (down/flat/up)
- **Class Probabilities**: Softmax distribution over classes
- **Confidence**: Maximum probability
- **Trading Recommendations**: Buy/sell flags and price targets (same format as MLP)

Example Input

Sequence Data Structure:

- **Sequence Length**: 128 days
- **Features per Day**: 8 dimensions
- **Total Shape**: `(1, 128, 8)`

Sample Sequence Features:

Day	Open	High	Low	Close	Volume	Sentiment	P/E	P/S
1	$225.96	$229.12	$225.64	$227.25	36,211,800	0.35	28.5	7.2
2	$227.03	$228.89	$224.87	$227.97	35,169,600	0.35	28.5	7.2
...	...	...	...	...	...	...	...	...
128	$269.92	$270.70	$265.32	$267.44	43,692,217	0.35	28.5	7.2

Model Configuration:

- **d_model**: 64 (embedding dimension)
- **nhead**: 8 (attention heads)
- **num_layers**: 3 (encoder layers)

Example Output

Model Prediction:

- **Predicted Class**: 2 (up / ■■)
- **Confidence**: `0.90` (90% confidence)

Class Probabilities Distribution:

- **Down** (Class 0): `10.0%` ■■
- **Flat** (Class 1): `0.0%`
- **Up** (Class 2): `90.0%` ■■■■■■■■■■■■■■■■■■■■

Trading Recommendations:

- **should_buy**: `True` ✓
- **should_sell**: `False` ✗

Price Targets:

- **Suggested Buy Price**: `$262.09` (98% of $267.44)
- **Suggested Sell Price**: `$280.81` (105% of $267.44)

Model Performance (Test Set):

- **Accuracy**: `50.00%`
- **Precision (Up)**: `0.49`
- **Recall (Up)**: `0.90`
- **F1-Score (Up)**: `0.64`

Technical Flow and Complete End-to-End Flow

1. **Price Sequence Collection**: Fetch historical daily OHLCV data (up to 128 days)
2. **Sentiment and Fundamental Data Integration**:
    - Retrieve sentiment score from sentiment analysis models
    - Fetch P/E and P/S ratios from fundamental data
3. **Sequence Feature Construction**:
    - For each day in price history, create 8-dimensional feature vector
    - Features: [open, high, low, close, volume, sentiment, P/E, P/S]
    - Truncate or pad sequence to max_len (128 days)
    - Result: (seq_len, 8) feature matrix
4. **Model Loading**: Load trained Transformer model from `saved_models/stock_transformer.pth`
5. **Input Projection**: Project 8 features to d_model dimensions (default: 32)
6. **Positional Encoding**: Add sinusoidal positional encodings to capture temporal order
    - Uses sin/cos functions with different frequencies
7. **Transformer Encoder Processing**:
    - Multi-head self-attention mechanism captures dependencies across time steps
    - Feedforward networks with residual connections
    - Layer normalization for stability
    - Multiple encoder layers stack to learn hierarchical patterns
8. **Sequence Aggregation**: Extract last time step representation as sequence summary
    - This summarizes the entire sequence context (similar to CLS token)
9. **Classification**: Pass through classification head to get 3-class logits
    - Layer normalization
    - Linear projection to 3 classes
    - Softmax for probability distribution
10. **Post-Processing**: Apply softmax, select class, generate trading recommendations
    - Map predicted class to direction (down/flat/up)
    - Generate buy/sell flags and price targets based on predicted direction
11. **Output**: Return PredictionOutput with direction prediction and price targets

Step-by-Step Example: Sequence Features to Probabilities

Input: 128 days × 8 features (OHLCV + sentiment + P/E + P/S)
• Shape: (1, 128, 8)
• Day 1: [225.96, 229.12, 225.64, 227.25, 36211800, 0.35, 28.5, 7.2]
• Day 128: [269.92, 270.70, 265.32, 267.44, 43692217, 0.35, 28.5, 7.2]

Step 1: Input Projection
• Project 8 → 32 dimensions: (1, 128, 8) × W[8×32] → **(1, 128, 32)**

Step 2: Positional Encoding
• Add sinusoidal PE: PE(i, d) = sin/cos(i / 10000^(2d/32))
• Output: **(1, 128, 32)** with temporal order

Step 3: Transformer Encoder (3 layers, 8 heads)
• Each layer: Multi-head attention (8 heads, 4 dims each) + FFN(32→128→32) + Residual + LayerNorm
• Attention: Each of 128 time steps attends to all 128 steps
• Output: **(1, 128, 32)**

Step 4: Sequence Aggregation
• Extract last time step: (1, 128, 32)[:, -1, :] → **(1, 32)**
• Summarizes entire sequence via attention

Step 5: Classification
• LayerNorm + Linear: (1, 32) × W[32×3] + b → logits = **[-1.2, -3.5, 2.1]**

Step 6: Softmax Calculation
• exp([-1.2, -3.5, 2.1]) = [0.301, 0.030, 8.166]
• Sum = 8.497
• Probabilities: [0.301/8.497, 0.030/8.497, 8.166/8.497] = **[0.035, 0.004, 0.961]**

Step 7: Post-Processing
• Predicted class: **2 (Up)**, Confidence=**0.961**
• Recommendations: should_buy=True, should_sell=False
• Prices: Buy=$262.09 (98%), Sell=$280.81 (105%)

Output: Class=2, Probs=[0.035, 0.004, 0.961], Buy=$262.09, Sell=$280.81

Model 4: LSTM Sentiment Analysis Model

Input

- **Text Sequences**: Financial news headlines or text snippets
- **Vocabulary**: Pre-built vocabulary mapping words to indices
- **Sequence Length**: Variable (padded/truncated to fixed max length)
- **Model Configuration**:
- Embedding dimension: 100 (default)
- Hidden dimension: 128 (default)
- Number of layers: 2 (default)
- Dropout: 0.5 (default)
- Number of classes: 5 (Very Negative, Negative, Neutral, Positive, Very Positive)

Output

- **Sentiment Class**: One of five classes (0-4)
- **Class Probabilities**: Softmax probabilities for each sentiment class
- **Aggregate Sentiment Score**: Converted to [-1, 1] range for integration with stock models

Example Input

Text Input:

- **Original Text**: "Apple reports strong quarterly earnings, beats expectations"

Preprocessing Details:

- **Tokenized Sequence**: [apple, reports, strong, quarterly, earnings, beats, expectations]
- **Sequence Length**: 7 tokens (padded to max_length: 128)
- **Vocabulary Size**: 5,000 words
- **Embedding Dimension**: 100-dimensional vectors

Model Configuration:

- **Embedding Dim**: 100
- **Hidden Dim**: 128
- **Layers**: 2
- **Dropout**: 0.5
- **Output Classes**: 5 (sentiment classes)

Example Output

Sentiment Prediction:

- **Predicted Class**: 3 (Positive / ■■)
- **Confidence**: 0.70 (70% probability for Neutral, but class 3 selected)

Class Probabilities Distribution:

- **Very Negative** (Class 0): 5.0% ■
- **Negative** (Class 1): 10.0% ■■
- **Neutral** (Class 2): 70.0% ■■■■■■■■■■■■■■■■■■■■■■
- **Positive** (Class 3): 12.0% ■■■
- **Very Positive** (Class 4): 3.0% ■

Aggregate Sentiment Score:

- **Score**: 0.35 (mapped to [-1, 1] range)
- **Interpretation**: Slightly positive sentiment

Model Performance (Test Set):

- **Accuracy**: 44.74%
- **Macro F1-Score**: 0.1236
- **Note**: Model shows bias toward Neutral class

Technical Flow and Complete End-to-End Flow

1. **News Headline Collection**:
   - Fetch news headlines from NewsAPI for stock ticker
   - Cache headlines in JSON files (`cache/news_*.json`)
2. **Text Preprocessing**:
   - Clean text (remove HTML tags, URLs, normalize whitespace)
   - Convert to lowercase for consistent tokenization
   - Tokenize input text into words
   - Map words to vocabulary indices
   - Pad/truncate sequences to fixed length
3. **Vocabulary Building**:
   - Build vocabulary from training corpus (Financial PhraseBank + news headlines)
   - Map words to indices, handle unknown words with UNK token
4. **Model Training** (one-time process):
   - Train LSTM on Financial PhraseBank dataset with VADER labels
   - Use early stopping based on validation loss
   - Save trained model to `saved_models/sentiment_lstm.pth`
5. **Embedding Layer**:
   - Convert word indices to dense vectors
   - Supports random initialization or pre-trained embeddings
   - Embedding dimension: 100
6. **Model Inference**:
   - Load trained LSTM model
   - Pass tokenized sequence through embedding layer
7. **LSTM/GRU Processing**:
   - Process sequence through bidirectional or unidirectional LSTM/GRU
   - Multiple layers with dropout for regularization
   - Hidden state dimension: 128
8. **Sequence Representation**:
   - Extract final hidden state from last time step
   - This captures the overall sentiment of the sequence
9. **Classification**:
   - Apply dropout for regularization
   - Linear projection to 5 classes
   - Softmax for probability distribution
10. **Class Prediction**: Apply softmax to get class probabilities, select class with highest probability
11. **Score Conversion**: Map 5-class prediction to continuous [-1, 1] sentiment score
    - Used as input feature for stock prediction models
12. **Integration**: Aggregate scores from multiple headlines and feed to stock prediction models

Step-by-Step Example: Sentence to Probabilities

Input: "Apple reports strong quarterly earnings, beats expectations"

Step 1: Text Preprocessing

- Tokenize: ["apple", "reports", "strong", "quarterly", "earnings", "beats", "expectations"]
- Vocabulary mapping: [42, 156, 89, 234, 567, 123, 445]
- Padding to 128: [42, 156, 89, 234, 567, 123, 445, 0, ..., 0]

Step 2: Embedding Layer

- Word indices → 100-dim vectors: (1, 128) → **(1, 128, 100)**
- Example: "apple"[42] → [0.12, -0.45, 0.78, ...]

Step 3: LSTM Processing (2 layers, 128 hidden)

- Layer 1: (1, 128, 100) → (1, 128, 128)
- Layer 2: (1, 128, 128) → (1, 128, 128)
- Extract last step: (1, 128, 128)[:, -1, :] → h_final **(1, 128)**

Step 4: Classification Head

- Dropout(0.5) + Linear: h_final[128] × W[128×5] → logits = **[-2.3, -1.1, 0.8, 1.5, 0.2]**

Step 5: Softmax Calculation

- exp([-2.3, -1.1, 0.8, 1.5, 0.2]) = [0.10, 0.33, 2.23, 4.48, 1.22]
- Sum = 8.36
- Probabilities: [0.10/8.36, 0.33/8.36, 2.23/8.36, 4.48/8.36, 1.22/8.36] = **[0.012, 0.039, 0.267, 0.536, 0.146]**

Step 6: Score Conversion

- Predicted class: **3 (Positive)**, Confidence=**0.536**
- Map to continuous score: **0.5** (in [-1, 1] range)

Output: Class=3, Probs=[0.012, 0.039, 0.267, 0.536, 0.146], Score=0.5

Model 5: BERT Sentiment Analysis Model

Input

- **Text Sequences**: Financial news headlines or text snippets
- **Maximum Length**: 128 tokens (default)
- **Model Base**: Pre-trained BERT model (bert-base-uncased or financial BERT variants)
- **Model Configuration**:
- Number of classes: 5
- Dropout: 0.1 (default)
- Fine-tuning approach: Full model fine-tuning

Output

- **Sentiment Class**: One of five classes (0-4)
- **Class Probabilities**: Softmax probabilities
- **Aggregate Sentiment Score**: Converted to [-1, 1] range

Example Input

Text Input:

- **Original Text**: "NVIDIA announces breakthrough AI chip technology, stock surges"

BERT Tokenization:

- **Tokenized Sequence**: [CLS] nvidia announces breakthrough ai chip technology stock surges [SEP]
- **Input IDs**: [101, 12345, 6789, 2345, 3456, 4567, 5678, 6789, 7890, 8901, 102]
- **Attention Mask**: [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, ...] (padded to 128)
- **Sequence Length**: 11 tokens (padded to max_length: 128)

Model Configuration:

- **Base Model**: bert-base-uncased (or financial BERT variant)
- **Max Length**: 128 tokens
- **Output Classes**: 5 (sentiment classes)
- **Dropout**: 0.1
- **Fine-tuning**: Full model fine-tuning

Example Output

Sentiment Prediction:

- **Predicted Class**: 4 (Very Positive / ■■■■)
- **Confidence**: 0.48 (48% probability)

Class Probabilities Distribution:

- **Very Negative** (Class 0): 2.0%
- **Negative** (Class 1): 5.0% ■
- **Neutral** (Class 2): 15.0% ■■■
- **Positive** (Class 3): 30.0% ■■■■■■
- **Very Positive** (Class 4): 48.0% ■■■■■■■■■■

Aggregate Sentiment Score:

- **Score**: 0.82 (mapped to [-1, 1] range)
- **Interpretation**: Strong positive sentiment

Model Performance Metrics:

Metric	Value
Test Accuracy	50.0%
Validation Accuracy	68.4%
Macro Precision	0.313
Macro Recall	0.310
Macro F1-Score	0.290

Per-Class Performance:

Class	Precision	Recall	F1-Score	Support
Very Negative (0)	0.00	0.00	0.00	3
Negative (1)	0.31	0.50	0.38	8
Neutral (2)	0.59	0.76	0.67	17
Positive (3)	0.67	0.29	0.40	7
Very Positive (4)	0.00	0.00	0.00	3

Training Configuration:

- **Epochs**: 5 (with early stopping, patience=2)
- **Batch Size**: 8
- **Learning Rate**: 2e-5

Technical Flow and Complete End-to-End Flow

1. **News Headline Collection**:
   - Fetch news headlines from NewsAPI for stock ticker
   - Cache headlines in JSON files for reuse
2. **BERT Tokenization**:
   - Use BERT tokenizer to convert text to subword tokens
   - Add special tokens: [CLS] at start, [SEP] for separation
   - Truncate/pad to max_length (128 tokens)
   - Create attention mask for padding tokens
3. **Model Fine-tuning** (one-time process):
   - Load pre-trained BERT model (bert-base-uncased)
   - Fine-tune on Financial PhraseBank dataset using Hugging Face Trainer
   - Training: 5 epochs, batch_size=8, learning_rate=2e-5
   - Early stopping with patience=2 epochs
   - Use Hugging Face Trainer API with Adam optimizer
   - Mixed precision training (FP16) if GPU available
   - Evaluation after each epoch
   - Save fine-tuned model to `saved_models/bert_sentiment/final_model/`
4. **BERT Encoder**:
   - Input embeddings: token embeddings + positional embeddings + segment embeddings
   - Multi-layer Transformer encoder with self-attention
   - Pre-trained weights capture rich linguistic patterns
5. **Model Inference**:
   - Load fine-tuned BERT model and tokenizer
   - Pass tokenized input through BERT encoder
6. **Classification Head**:
   - Extract [CLS] token representation (sequence summary)
   - Linear projection to 5 classes
   - Softmax for probability distribution
7. **Class Prediction**: Apply softmax to get probabilities, select predicted class
8. **Score Conversion**: Map 5-class prediction to continuous [-1, 1] sentiment score
   - Integrated into stock prediction pipeline
9. **Integration**: Aggregate scores from multiple headlines and integrate with stock prediction models

Step-by-Step Example: Sentence to Probabilities

Input: "NVIDIA announces breakthrough AI chip technology, stock surges"

Step 1: BERT Tokenization

- WordPiece tokenization: "breakthrough" → ["break", "##through"]
- Add special tokens: ["[CLS]", "nvidia", "announces", "break", "##through", "ai", "chip", "technology", "stock", "surges", "[SEP]"]
- Token IDs: [101, 12345, 6789, 2345, 3456, 4567, 5678, 6789, 7890, 8901, 102]
- Attention mask: [1, 1, ..., 1, 0, ..., 0] (padded to 128)

Step 2: Input Embeddings

- Token + Position + Segment embeddings: **(1, 11, 768)**
- Combined: E = Token_Embed + Pos_Embed + Seg_Embed

Step 3: BERT Encoder (12 layers)

- Each layer: Multi-head attention (12 heads) + FFN(768→3072→768) + Residual + LayerNorm
- Process through 12 layers: (1, 11, 768) → (1, 11, 768)
- Extract [CLS] token: (1, 11, 768)[:, 0, :] → **(1, 768)**

Step 4: Classification Head

- Dropout(0.1) + Linear: [CLS][768] × W[768×5] + b → logits = **[-3.2, -0.8, 0.5, 2.1, 3.5]**

Step 5: Softmax Calculation

- exp([-3.2, -0.8, 0.5, 2.1, 3.5]) = [0.041, 0.449, 1.649, 8.166, 33.115]
- Sum = 43.420
- Probabilities: [0.041/43.420, 0.449/43.420, 1.649/43.420, 8.166/43.420, 33.115/43.420] = **[0.0009, 0.0103, 0.0380, 0.1881, 0.7627]**

Step 6: Score Conversion

- Predicted class: **4 (Very Positive)**, Confidence=**0.7627**
- Map to continuous score: **1.0** (in [-1, 1] range)

Output: Class=4, Probs=[0.0009, 0.0103, 0.0380, 0.1881, 0.7627], Score=1.0

Model Integration Pipeline

Data Flow

1. **Data Collection**: Fetch price history, news headlines, and fundamental data
2. **Sentiment Analysis**: Process news headlines through LSTM/BERT models
3. **Feature Extraction**: Combine price, sentiment, and fundamental features
4. **Stock Prediction**: Run through Baseline, MLP, or Transformer models
5. **Scenario Generation**: Optional Monte Carlo simulation for risk assessment
6. **Output Aggregation**: Combine predictions from multiple models for final recommendation

Performance Summary

- **MLP Model**: 40.93% accuracy, struggles with class imbalance
- **Transformer Model**: 50.00% accuracy, better sequence modeling
- **LSTM Sentiment**: 44.74% accuracy, small dataset limitation
- **BERT Sentiment**: 50.0% test accuracy, 68.4% validation accuracy, superior to LSTM
- **Baseline Model**: Simple but interpretable moving average strategy

---

Complete Sentiment Analysis Pipeline

The sentiment analysis system provides a complete pipeline from data collection to model inference, enabling integration with stock prediction models.

1. Data Collection

- **News Headlines**: Collected from NewsAPI for each stock ticker
- **Storage**: Headlines cached in JSON files (`cache/news_*.json`)
- **Format**: Each cache file contains ticker and list of headline dictionaries with title, description, and metadata

2. Label Generation

Labels are generated from two sources:

Source 1: Financial PhraseBank Dataset

- Pre-labeled financial sentences with manual annotations
- Contains examples like "Stock price crashed 50% following fraud allegations" (Label 0: Very Negative)
- Stored in `data/financial_phrasebank.csv`
- Provides high-quality ground truth labels for training

Source 2: VADER-Labeled News Headlines

- News headlines collected from NewsAPI are automatically labeled using VADER sentiment analyzer
- VADER produces compound scores in range [-1, 1]
- **Label Mapping** (VADER compound score → 5 classes):
- `compound < -0.6` → Label 0 (Very Negative)
- `-0.6 ≤ compound < -0.2` → Label 1 (Negative)
- `-0.2 ≤ compound ≤ 0.2` → Label 2 (Neutral)
- `0.2 < compound ≤ 0.6` → Label 3 (Positive)
- `compound > 0.6` → Label 4 (Very Positive)

3. Dataset Preparation

- **Combination**: Financial PhraseBank and labeled news headlines are combined
- **Splitting**: Dataset split into train/validation/test sets (70%/15%/15%)
- **Stratification**: Splits maintain class distribution for balanced evaluation
- **Caching**: Prepared datasets cached in `data/sentiment_splits_cache.json` for reproducibility

4. Model Training

LSTM Model Training:

- Build vocabulary from training texts
- Initialize embeddings (random or pre-trained GloVe)
- Train bidirectional LSTM with 2 layers, 128 hidden units
- Use early stopping based on validation loss
- Save best model checkpoint

BERT Model Training:

- Load pre-trained BERT model (bert-base-uncased)
- Fine-tune on financial sentiment dataset using Hugging Face Trainer
- Training configuration: 5 epochs, batch_size=8, learning_rate=2e-5
- Early stopping with patience=2 epochs
- Save fine-tuned model to `saved_models/bert_sentiment/final_model/`

5. Inference Pipeline

1. **Text Input**: Receive news headline or financial text
2. **Preprocessing**: Clean text (remove HTML, URLs, normalize whitespace)
3. **Tokenization**:
   - LSTM: Word-level tokenization with vocabulary mapping
   - BERT: Subword tokenization with special tokens ([CLS], [SEP])

4. **Model Inference**: Pass tokenized sequence through trained model
5. **Class Prediction**: Extract predicted class (0-4) from model output
6. **Score Conversion**: Map 5-class prediction to continuous [-1, 1] sentiment score
    - Class 0 (Very Negative) → -1.0
    - Class 1 (Negative) → -0.5
    - Class 2 (Neutral) → 0.0
    - Class 3 (Positive) → 0.5
    - Class 4 (Very Positive) → 1.0

6. Integration with Stock Prediction

- **Aggregation**: Multiple headlines processed and scores averaged
- **Feature Integration**: Aggregate sentiment score added as feature to stock prediction models
- **Real-time Updates**: New headlines fetched and analyzed when stock analysis is requested
- **Fallback**: If deep learning models unavailable, VADER used directly for sentiment scoring

---

Conclusion

The AI Stocks project demonstrates a comprehensive approach to stock prediction by integrating multiple machine learning paradigms: traditional technical analysis (baseline), deep learning for tabular data (MLP), sequence modeling (Transformer), and natural language processing (LSTM/BERT). Each model contributes unique insights, and their combination provides a robust framework for financial decision-making. The system successfully integrates heterogeneous data sources (price, sentiment, fundamentals) and demonstrates the practical application of modern deep learning techniques to financial markets.