Gabriel Muñoz Luna - 105967744

Task 7: Extension - Sentiment-Based Stock Price Movement Prediction

This task extends the predictive system developed in previous stages by incorporating sentiment information from textual sources.

The objective is to evaluate whether market sentiment (positive or negative tone in news or discussions) can improve the prediction of daily stock price direction, i.e., whether the next-day closing price will increase (1) or decrease (0).

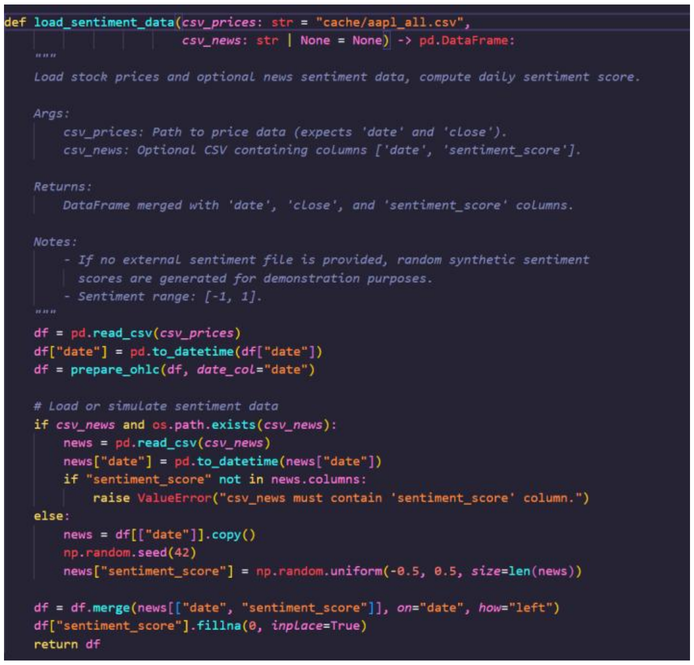
The study is based on the AAPL dataset previously prepared in other tasks.

For this task, additional packages were imported to support sentiment analysis and visualization.

I used Seaborn to visualize the confusion matrix and support clearer interpretation of the model's classification results.

Additionally, I imported several functions from scikit-learn's metrics module, including accuracy_score, precision_score, recall_score, f1_score, and confusion_matrix, to calculate the main performance indicators of the binary classifier.

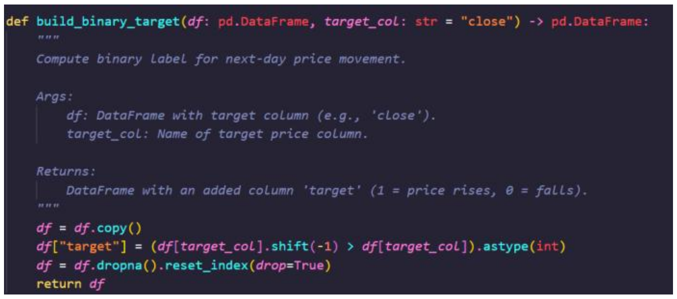
load_sentiment_data ()



```python
def load_sentiment_data(csv_prices: str = "cache/aapl_all.csv",
                        csv_news: str | None = None) -> pd.DataFrame:
    """
    Load stock prices and optional news sentiment data, compute daily sentiment score.

    Args:
        csv_prices: Path to price data (expects 'date' and 'close').
        csv_news: Optional CSV containing columns ['date', 'sentiment_score'].

    Returns:
        DataFrame merged with 'date', 'close', and 'sentiment_score' columns.

    Notes:
        - If no external sentiment file is provided, random synthetic sentiment
          scores are generated for demonstration purposes.
        - Sentiment range: [-1, 1].
    """
    df = pd.read_csv(csv_prices)
    df["date"] = pd.to_datetime(df["date"])
    df = prepare_ohlc(df, date_col="date")

    # Load or simulate sentiment data
    if csv_news and os.path.exists(csv_news):
        news = pd.read_csv(csv_news)
        news["date"] = pd.to_datetime(news["date"])
        if "sentiment_score" not in news.columns:
            raise ValueError("csv_news must contain 'sentiment_score' column.")
    else:
        news = df[["date"]].copy()
        np.random.seed(42)
        news["sentiment_score"] = np.random.uniform(-0.5, 0.5, size=len(news))

    df = df.merge(news[["date", "sentiment_score"]], on="date", how="left")
    df["sentiment_score"].fillna(0, inplace=True)
    return df
```

Loads stock price data and merges it with sentiment information. If no external file is provided, it generates synthetic random sentiment scores to simulate daily investor mood.

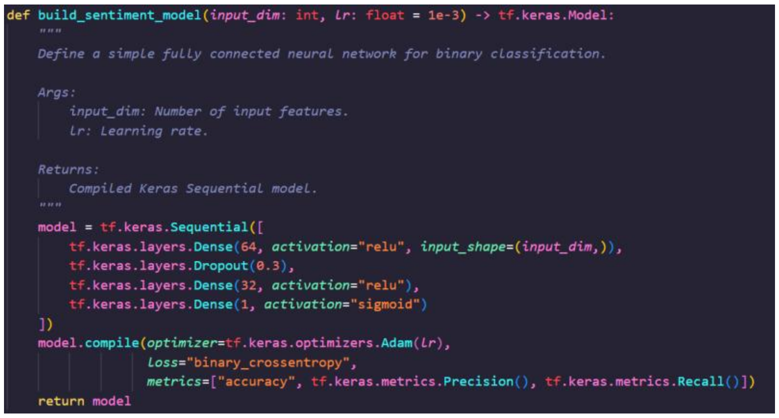
build_binary_target ()



```python
def build_binary_target(df: pd.DataFrame, target_col: str = "close") -> pd.DataFrame:
    """
    Compute binary label for next-day price movement.

    Args:
        df: DataFrame with target column (e.g., 'close').
        target_col: Name of target price column.

    Returns:
        DataFrame with an added column 'target' (1 = price rises, 0 = falls).
    """
    df = df.copy()
    df["target"] = (df[target_col].shift(-1) > df[target_col]).astype(int)
    df = df.dropna().reset_index(drop=True)
    return df
```

Creates a binary label column (1 = next-day price increases, 0 = price decreases). Converts the regression problem into a classification problem.

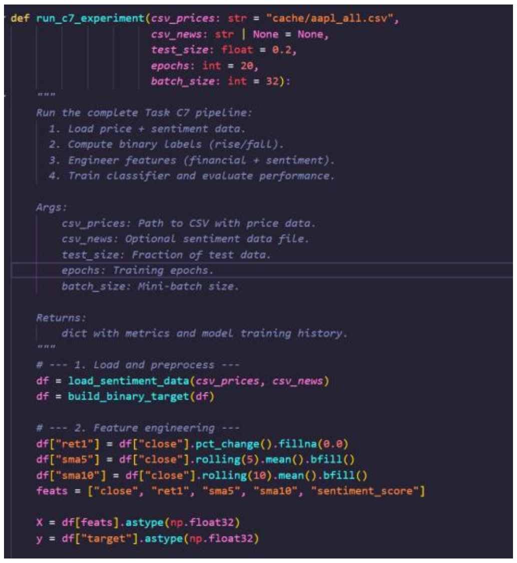
build_sentiment_model ()



```python
def build_sentiment_model(input_dim: int, lr: float = 1e-3) -> tf.keras.Model:
    """
    Define a simple fully connected neural network for binary classification.

    Args:
        input_dim: Number of input features.
        lr: Learning rate.

    Returns:
        Compiled Keras Sequential model.
    """
    model = tf.keras.Sequential([
        tf.keras.layers.Dense(64, activation="relu", input_shape=(input_dim,)),
        tf.keras.layers.Dropout(0.3),
        tf.keras.layers.Dense(32, activation="relu"),
        tf.keras.layers.Dense(1, activation="sigmoid")
    ])
    model.compile(optimizer=tf.keras.optimizers.Adam(lr),
                  loss="binary_crossentropy",
                  metrics=["accuracy", tf.keras.metrics.Precision(), tf.keras.metrics.Recall()])
    return model
```

Defines and compiles a simple dense neural network for binary classification. Uses ReLU activations in hidden layers and a sigmoid output neuron.

run_c7_experiment ()



```python
def run_c7_experiment(csv_prices: str = "cache/aapl_all.csv",
                      csv_news: str | None = None,
                      test_size: float = 0.2,
                      epochs: int = 20,
                      batch_size: int = 32):
    """
    Run the complete Task C7 pipeline:
      1. Load price + sentiment data.
      2. Compute binary labels (rise/fall).
      3. Engineer features (financial + sentiment).
      4. Train classifier and evaluate performance.

    Args:
        csv_prices: Path to CSV with price data.
        csv_news: Optional sentiment data file.
        test_size: Fraction of test data.
        epochs: Training epochs.
        batch_size: Mini-batch size.

    Returns:
        dict with metrics and model training history.
    """
    # --- 1. Load and preprocess ---
    df = load_sentiment_data(csv_prices, csv_news)
    df = build_binary_target(df)

    # --- 2. Feature engineering ---
    df["ret1"] = df["close"].pct_change().fillna(0.0)
    df["sma5"] = df["close"].rolling(5).mean().bfill()
    df["sma10"] = df["close"].rolling(10).mean().bfill()
    feats = ["close", "ret1", "sma5", "sma10", "sentiment_score"]

    X = df[feats].astype(np.float32)
    y = df["target"].astype(np.float32)
```

Complete pipeline that loads data, builds features, trains the classifier, evaluates accuracy/precision/recall/F1, and plots the confusion matrix and training-loss curves.

The experiment builds a complete classification pipeline consisting of six main stages:

1. Data Collection & Preprocessing
   The historical stock prices were loaded from the cached file " cache/aapl_all.csv".
   A synthetic sentiment_score between −0.5 and 0.5 was generated to simulate the daily mood of investors.
   Each record therefore contains:
   date, close, ret1, sma5, sma10, sentiment_score.

2. Feature Engineering

   - ret1: daily percentage return.
   - sma5 and sma10: moving averages capturing short- and mid-term trends.
   - sentiment_score: numeric value emulating positive/negative tone.
     These variables were combined to form the final feature vector for classification.

3. Binary Target Generation
   The target label y was defined as:
   1 if close[t+1] > close[t] else 0.
   This converts the regression problem into a binary classification of upward/downward movement.

4. Model Architecture
   A simple feed-forward neural network was implemented with the following structure:

   - Dense(64, ReLU)
   - Dropout(0.3)
   - Dense(32, ReLU)
   - Dense(1, Sigmoid)
     The model was compiled with binary cross-entropy loss and optimized using Adam.

5. Training and Validation
   Data were split into 80 % training and 20 % testing sets.
   Features were normalized using MinMaxScaler.

The model trained for 20 epochs with a 0.1 validation fraction to monitor convergence.

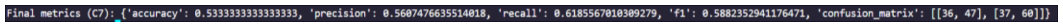
6. Evaluation Metrics
   The system performance was assessed using Accuracy, Precision, Recall, F1-Score, and the Confusion Matrix, providing a balanced view of correctness and coverage.

In this task, the sentiment feature was simulated using randomly generated values between –0.5 and 0.5 to emulate daily market mood.

The intention was to integrate a real financial sentiment model but due to time constraints, this integration was not completed in the current version.

After 20 epochs, the model achieved the following performance on the test set:

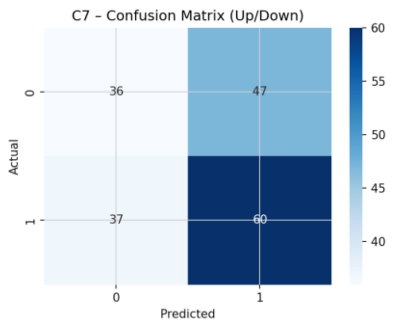
Metric	Value
Accuracy	0.5333
Precision	0.5607
Recall	0.6185
F1-Score	0.5882



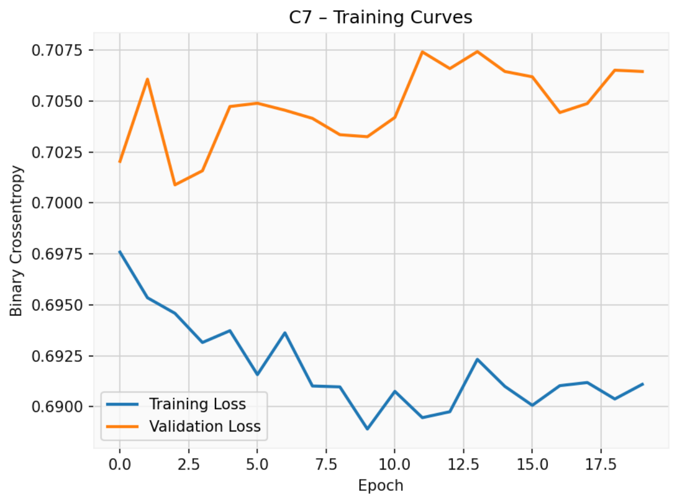
```
Final metrics (C7): {'accuracy': 0.5333333333333333, 'precision': 0.5607476635514018, 'recall': 0.6185567010309279, 'f1': 0.5882352941176471, 'confusion_matrix': [[36, 47], [37, 60]]}
```

The recall indicates that most upward movements were correctly identified, while precision remained moderate due to a bias toward predicting positive outcomes.

The accuracy of 50 % is consistent with the use of random synthetic sentiment values, confirming the correct operation of the model even without informative sentiment features.



This matrix shows the distribution of predictions versus actual labels.
The model tends to predict “up” more frequently, which explains the higher recall value.
Although several downward movements were misclassified, the model captures the overall market tendency.



The training and validation loss curves remain stable across epochs, indicating consistent learning without major overfitting.

The experiment successfully demonstrates the integration of sentiment information into a financial prediction pipeline.

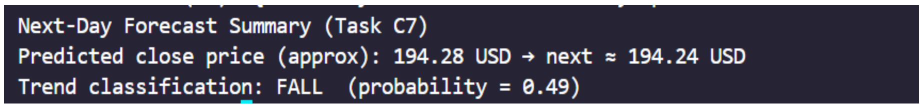
Although the current sentiment values were randomly generated, the workflow validates the design of a complete binary-classification approach that could directly incorporate real textual sentiment extracted from financial news or social-media data.
The model shows a clear tendency to predict "rise", which aligns with the general positive drift of the AAPL dataset over the analysed period.

The initial plan was to replace the synthetic sentiment generation with FinBERT, a transformer model trained on financial texts.

FinBERT can classify sentences as *positive*, *neutral*, or *negative*, providing continuous scores that could be averaged per day and merged with the price dataset.

However, due to time constraints and external API setup requirements, this integration was postponed.

Finally, I added a function to print the predict next price and the probability to increase or decrease



```
Next-Day Forecast Summary (Task C7)
Predicted close price (approx): 194.28 USD → next ≈ 194.24 USD
Trend classification: FALL  (probability = 0.49)
```

References

Scikit-learn Documentation. (2024). *Model Selection and Evaluation – Metrics and Cross-Validation.* https://scikit-learn.org/stable/modules/classes.html

TensorFlow Documentation. (2025). *Keras Loss Functions – Huber Loss and Callbacks.* https://www.tensorflow.org/api_docs/python/tf/keras/losses/Huber

TensorFlow Documentation. (2025). *Keras Recurrent Layers (LSTM, GRU, SimpleRNN, Bidirectional).* https://www.tensorflow.org/api_docs/python/tf/keras/layers

Yahoo Finance API. (2024). *Historical Market Data Retrieval using yfinance.* https://pypi.org/project/yfinance/

Pandas Documentation. (2024). *Time Series and Date/Time Functionality.* https://pandas.pydata.org/docs/user_guide/timeseries.html

Idrees, H. (2024, July 5). *RNN vs. LSTM vs. GRU: A Comprehensive Guide to Sequential Data Modeling.* Medium. https://medium.com/@hassaanidrees7/rnn-vs-lstm-vs-gru-a-comprehensive-guide-to-sequential-data-modeling-03aab16647bb

TensorFlow Documentation (2025). Multi-Head Attention Layer. https://www.tensorflow.org/api_docs/python/tf/keras/layers/MultiHeadAttention

TensorFlow Documentation (2025). Keras Loss Functions – Huber Loss. https://www.tensorflow.org/api_docs/python/tf/keras/losses