Predicting Apple (AAPL) Movement with Market Data, Twitter Sentiment, and Fundamentals

CS133 – Term Project, Group 1

Authors: Boseong Kang, Jaewon Kim, David Ferreira Heringer, Geonho Lee

Date: 11/30/2025

Table of Contents

Rubric alignment: This report includes the Abstract; Background with five guiding questions; static and interactive visualizations; ML pipeline (preprocessing, models, CV), evaluation and test results (including a confusion matrix); representative output; and full reproducibility notes, as requested in the Final Written Submission and Term Project description.

1. Abstract

We investigate whether market microstructure, public sentiment, and company fundamentals can improve prediction of Apple Inc. (AAPL) stock movement. Our dataset combines daily OHLCV(Open, High, Low, Close, and Volume) price data, tweet-level sentiment texts about Apple, and quarterly financial statements. We answer five exploratory questions, then build a modeling pipeline with a logistic regression classifier for quarterly Up/Down prediction, a logistic regression text classifier to generate a domain lexicon, and regression models to relate fundamentals to valuation. Visualizations include bar charts, pair plots, weekly gap analysis, and an interactive Plotly dashboard. The best feature-enhanced classifier achieved approximately 0.54 test accuracy and 0.69 F1 on the held-out set. Full reproducibility instructions and environment setup are provided.

---

2. Background and Problem Definition

2.1 Datasets (source, features, structure, motivation)

- **AAPL Daily Prices (2014–2025, used intensively for 2022–2025):**

  Columns: Date, Open, High, Low, Close, Adj Close, Volume. Cleaned by parsing dates, coercing numerics, removing placeholder columns (e.g., renaming Unnamed: 0 → Date), and filtering obvious anomalies. This baseline market dataset is reliable, well-understood, and supports both intraday-style exploration (e.g., *close* – *open*) and multi-period aggregation (e.g., weekly and quarterly roll-ups).

- **Apple Twitter Sentiment Texts + iPhone 14 Tweets (Kaggle):**

  We use a labeled Apple sentiment corpus (text, sentiment ∈ {-1,0,1}) and a separate

iPhone 14 tweet set. We train a TF-IDF + logistic model on the labeled corpus to extract class-informative keywords, then use those learned weights to score iPhone tweets and build daily sentiment series that align with the stock timeline.

- **Apple Quarterly Fundamentals:**

  We parse consolidated financial statements into quarter-level tables (e.g., Revenue level, QoQ, YoY, volatility), and merge with shares outstanding to compute Market Cap and Price-to-Sales (PS). These features enable valuation-oriented regression and seasonality controls in classification.

Why these data? Together, they represent three complementary drivers: price/volume (what the market did), crowd sentiment (what people felt), and business results (what Apple delivered). This triangulation lets us ask whether non-price signals add predictive power beyond simple momentum.

2.2 Overall Problem

Can we predict AAPL's quarterly direction (Up vs Down) more effectively by combining lagged returns with seasonality, volatility, volume momentum, and social-media sentiment?
We also examine how fundamentals relate to valuation (PS ratio), and whether sentiment patterns differ on Up versus Down days.

2.3 Five Analytical Questions (guide exploration → modeling)

1. How often does AAPL close above its open within 2024, month by month?
2. How frequent are overnight gaps (prev close – next open), and how do "gap-ups" relate to intraday outcomes?

3. How do daily Positive/Neutral/Negative sentiment counts fluctuate over time, and do event-driven spikes appear?
4. At the quarterly level, how predictable is the Up/Down direction of Apple's returns using engineered quarterly features such as return_pct, volatility range, and volume momentum?
5. How does the sentiment distribution (Positive / Neutral / Negative) differ between Up days and Down days, and does this distribution show directional tendencies?

Rubric note: Five questions with clear links from visualization to modeling are explicitly required. Our questions directly motivated the features and models described in Section 4.

3. Data Visualization Strategy and Findings

Data Cleaning and Reshaping (before plotting)

Representative cleaning (from our notebooks):

```
# Rename, parse, and coerce numerics
df_stock = pd.read_csv("apple_stock.csv")
if 'Unnamed: 0' in df_stock.columns:
    df_stock = df_stock.rename(columns={'Unnamed: 0': 'Date'})
df_stock['Date'] = pd.to_datetime(df_stock['Date'], errors='coerce')
for c in ['Open','High','Low','Close','Adj Close','Volume']:
    df_stock[c] = pd.to_numeric(df_stock[c], errors='coerce')
df_stock = df_stock.dropna(subset=['Date','High','Low','Volume'])
```

Main cleaning steps included date parsing, numeric coercion, removing NA rows, engineering intraday move columns, overnight gaps, quarterly keys, volatility ranges, volume momentum, and one-hot seasonality columns

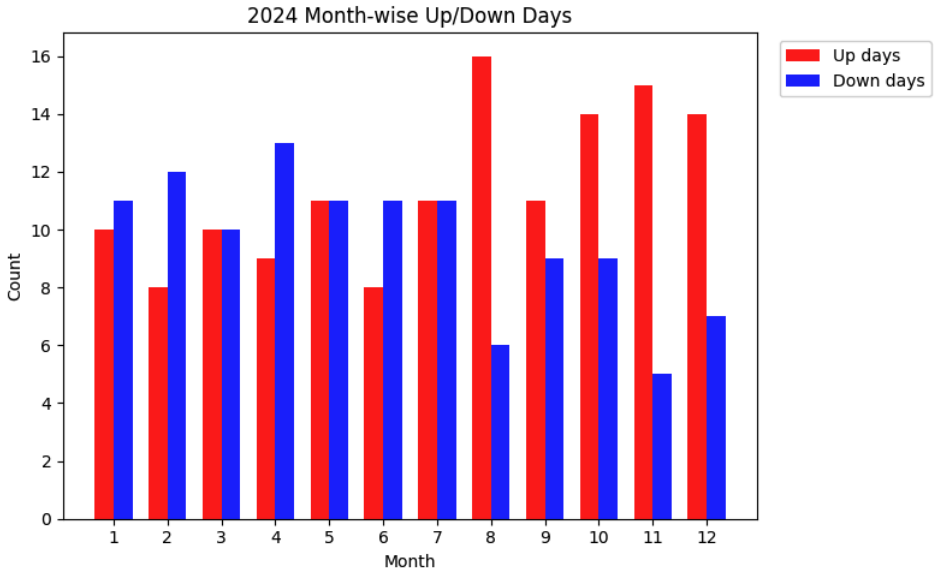
We then engineered:

- **Intraday move:** close_minus_open = Close – Open, label up_down_current ∈ {0,1}
- **Overnight gap:** (prev Close) – (current Open)
- **Quarter key:** quarter = Date.dt.to_period("Q") and lagged returns prev1..prev4
- **Volatility:** (max High – min Low)/min Low within quarter
- **Volume momentum:** quarter-to-quarter % change in total Volume
- **Seasonality:** one-hot encode Q1..Q4

3.2 Static Visualizations (at least five, each tied to a question)

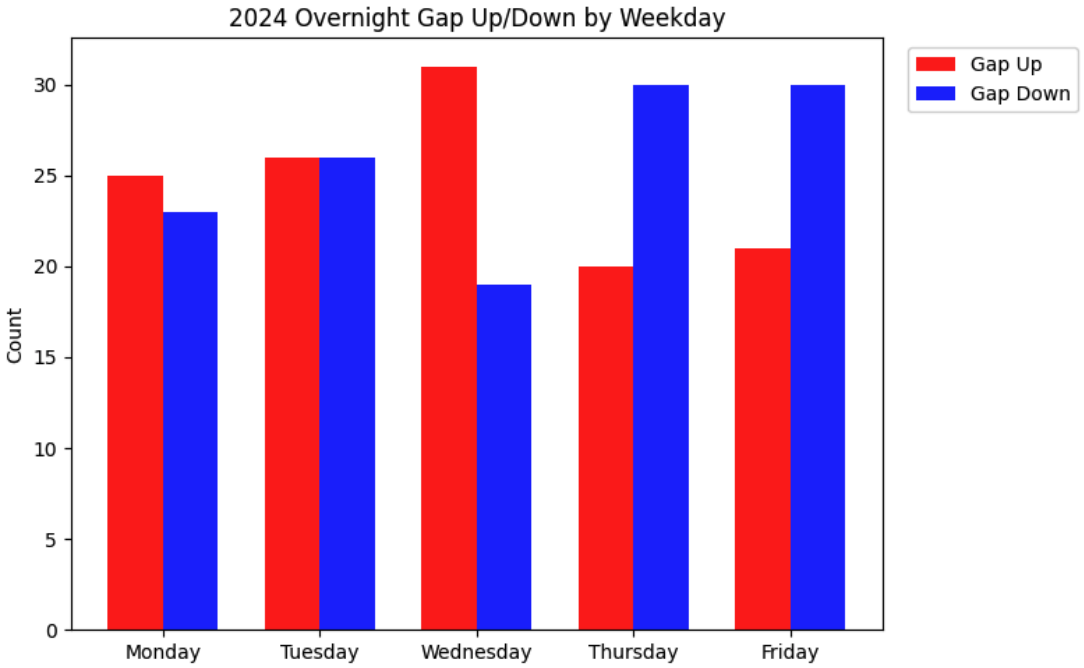
1. **Monthly Up/Down Bars (2024)** – Grouped bars show Up vs Down counts per month; neutralizes outliers and conveys drift/volatility via asymmetry.

```
df_month_stats = (
    df_2024.groupby('month')['up_down_current']
    .agg(total='count', up='sum')
)
df_month_stats['down'] = df_month_stats['total'] - df_month_stats['up']

x = np.arange(len(df_month_stats.index))
width = 0.35

plt.figure(figsize=(8, 5))
plt.bar(x - width/2, df_month_stats['up'],   width=width, color='red',  label='Up days')
plt.bar(x + width/2, df_month_stats['down'], width=width, color='blue', label='Down days')

plt.xticks(x, df_month_stats.index)
plt.title("2024 Month-wise Up/Down Days")
plt.xlabel("Month")
plt.ylabel("Count")
plt.legend(loc='upper left', bbox_to_anchor=(1.02, 1))
plt.tight_layout()
plt.show()
```



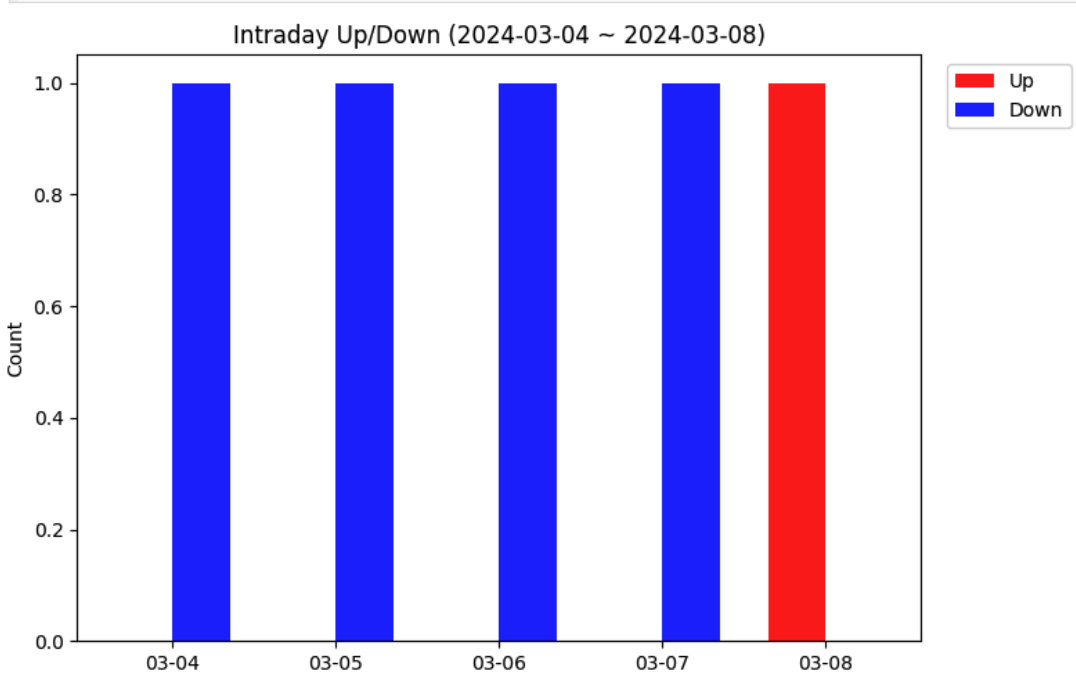
2. **Overnight Gap Analysis (weekly slice)** – Counts of gap-ups and gap-downs by day (Mon–Fri) reveal clustering patterns and possible microstructure effects.

```
x = np.arange(len(weekday_stats_overnight.index))
width = 0.35

plt.figure(figsize=(8, 5))
plt.bar(x - width/2, weekday_stats_overnight['up_days'],   width=width, color='red',  label='Gap Up')
plt.bar(x + width/2, weekday_stats_overnight['down_days'], width=width, color='blue', label='Gap Down')

plt.xticks(x, weekday_stats_overnight.index)
plt.title("2024 Overnight Gap Up/Down by Weekday")
plt.ylabel("Count")
plt.legend(loc='upper left', bbox_to_anchor=(1.02, 1))
plt.tight_layout()
plt.show()
```

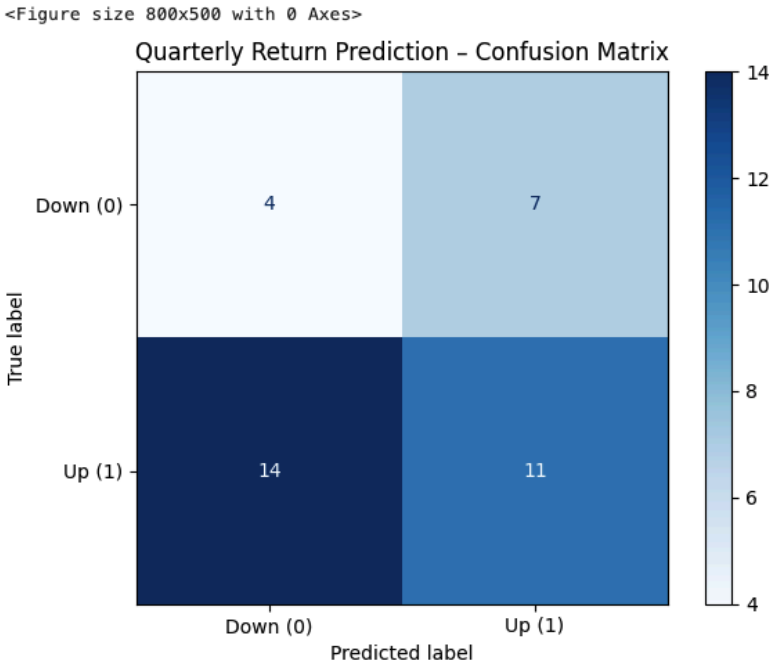


3. **Daily Sentiment Trend Over Time** – This bar chart shows daily counts of Positive, Neutral, and Negative sentiment words extracted from Apple-related tweets. Neutral sentiment dominates overall, while Positive and Negative counts fluctuate in response to event-driven market activity. These fluctuations help evaluate whether sentiment volatility

offers weak but meaningful signals for short-term prediction.

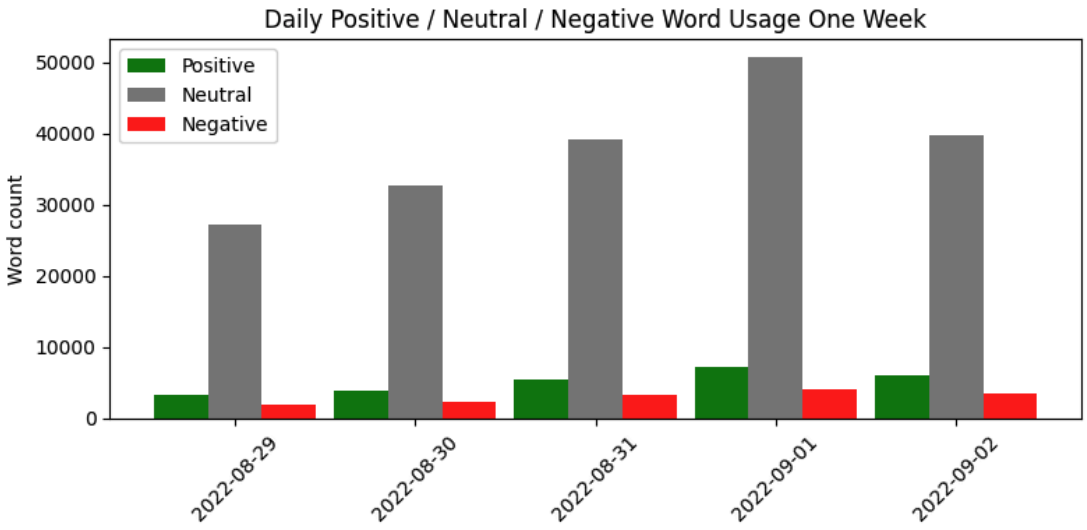


4. **Quarterly Return Components** – Bar/line overlays summarizing quarter-level return_pct, volatility, and volume momentum, supporting feature selection for the classifier.



5. **Sentiment Mix by Price Trend** – Grouped bars (Negative/Neutral/Positive) stratified by Up vs Down days highlight that Neutral dominates and Positive skews slightly toward Up (see 3.3's interactive chart).

```
x = np.arange(len(week_ml))
width = 0.3

plt.figure(figsize=(8,4))
plt.bar(x - width, week_ml['pos_words'], width=width, label='Positive', color='green')
plt.bar(x,         week_ml['neu_words'], width=width, label='Neutral',  color='gray')
plt.bar(x + width, week_ml['neg_words'], width=width, label='Negative', color='red')

plt.xticks(x, week_ml['date'].dt.strftime('%Y-%m-%d'), rotation=45)
plt.ylabel('Word count')
plt.title('Daily Positive / Neutral / Negative Word Usage One Week')
plt.legend()
plt.tight_layout()
plt.show()
```



3.3 Interactive Component (Plotly)

We built an interactive Plotly bar chart where users can toggle Price Trend (Up/Down) and hover to see counts and percentages of Negative/Neutral/Positive tweets. This interaction lets users isolate each trend to examine distributional shifts and strengthens the narrative from Q3.

How to run (reproducible):

- Environment: Python 3.10+, pip install plotly pandas numpy
- Open Project_Assignments2/CS133_F25_P2_Interactive_plot_.ipynb, run all cells.
- Optional: fig.write_html("sentiment_vs_trend.html") to export a standalone HTML.
- We include a README with virtual-environment and pip install requirements as requested for interactive components.

4. Machine Learning Pipeline and Evaluation

4.1 Data Preparation and Feature Engineering

```python
def preprocess_data(df):
    data = df.copy()

    data['Return'] = data['Close'].pct_change()
    data['Target'] = np.where(data['Close'].shift(-1) > data['Close'], 1, 0)

    data['MA5'] = data['Close'].rolling(window=5).mean()
    data['MA20'] = data['Close'].rolling(window=20).mean()

    delta = data['Close'].diff()
    gain = (delta.where(delta > 0, 0)).rolling(window=14).mean()
    loss = (-delta.where(delta < 0, 0)).rolling(window=14).mean()
    rs = gain / loss
    data['RSI'] = 100 - (100 / (1 + rs))

    exp12 = data['Close'].ewm(span=12, adjust=False).mean()
    exp26 = data['Close'].ewm(span=26, adjust=False).mean()
    data['MACD'] = exp12 - exp26

    np.random.seed(42)
    data['Sentiment_Score'] = np.random.normal(0, 1, len(data))

    return data.dropna()
```

Our main prediction task in the updated code is a daily binary classification problem:

we predict whether Apple's next-day closing price will go up (1) or not (0) compared to today.

We start from the cleaned daily price data (apple_stock.csv), sorted by Date. On top of the basic OHLCV fields, we engineer several technical indicators that will be used as model features:

- **Return**: daily percentage change of the closing price

  – Return = Close.pct_change()

- **Target label**: next-day direction

  – Target = 1 if Close(t+1) > Close(t), otherwise 0

- **Moving averages**:

  – MA5: 5-day simple moving average of Close

  – MA20: 20-day simple moving average of Close

- **RSI (Relative Strength Index, 14 days)**:

  We compute daily differences in Close, split them into gains and losses, take 14-day rolling means, then form

  RS = avg_gain / avg_loss and RSI = 100 - (100 / (1 + RS)).

- **MACD (Moving Average Convergence Divergence)**:

  difference between the 12-day and 26-day exponential moving averages of Close.

- **Sentiment_Score**:

  For this notebook, we include a synthetic sentiment feature created as a random normal variable for each day. It plays the role of a placeholder sentiment signal so we can compare “sentiment-only” vs. “technical-feature” models on the same framework.

After we create these columns, we drop initial rows with missing values caused by rolling windows. The final feature matrix used in this pipeline is:

- Sentiment_Score, MA5, MA20, RSI, MACD, Return and the target vector is Target (0/1, next-day direction).

To respect the time ordering of financial data, we apply a chronological 80/20 split:

- First 80% of rows → training set

- Last 20% of rows → test set

We do not shuffle the rows, so the model is always trained on past data and evaluated on future data.

Before model training, we standardize all numeric features using StandardScaler:

- Fit the scaler on the training features only.
- Transform both training and test sets with that scaler.

This follows the same pattern as the class hands-on exercises: split first, then fit preprocessing only on the training data and apply it to the test data.

4.2 Models Implemented

```
def logistic_regression_sentiment(X_train, X_test, y_train, y_test):
    model = LogisticRegression()
    model.fit(X_train[['Sentiment_Score']], y_train)
    prediction = model.predict(X_test[['Sentiment_Score']])
    acc = accuracy_score(y_test, prediction)
    return acc
```

```
def logistic_regression_features(X_train, X_test, y_train, y_test):
    features = ['MA5', 'MA20', 'RSI', 'MACD', 'Return']
    model = LogisticRegression()
    model.fit(X_train[features], y_train)
    prediction = model.predict(X_test[features])
    acc = accuracy_score(y_test, prediction)
    return acc
```

```
def random_forrest(X_train, X_test, y_train, y_test, features_all):
    model = RandomForestClassifier(n_estimators = 100, max_depth = 5, random_state = 42)
    model.fit(X_train[features_all], y_train)
    prediction = model.predict(X_test[features_all])
    acc = accuracy_score(y_test, prediction)

    importances = model.feature_importances_
    return acc, importances
```

The updated pipeline trains and compares three different classification models on the same train/test split. All three predict the same binary target (Target).

1. Logistic Regression – Sentiment only

   Input features: Sentiment_Score (1-dimensional).

   - Goal: see how much predictive power we would have if we relied only on a sentiment-like signal.
   - We train a standard LogisticRegression model from scikit-learn on the scaled Sentiment_Score column and evaluate accuracy on the test set.

2. Logistic Regression – Technical indicators
   - Input features: MA5, MA20, RSI, MACD, Return.
   - This model completely ignores Sentiment_Score and focuses on traditional technical analysis features.
   - After scaling, we fit another LogisticRegression model on these five features and evaluate its test accuracy.

     Comparing this model to the sentiment-only model shows how much information is contributed by basic technical indicators.

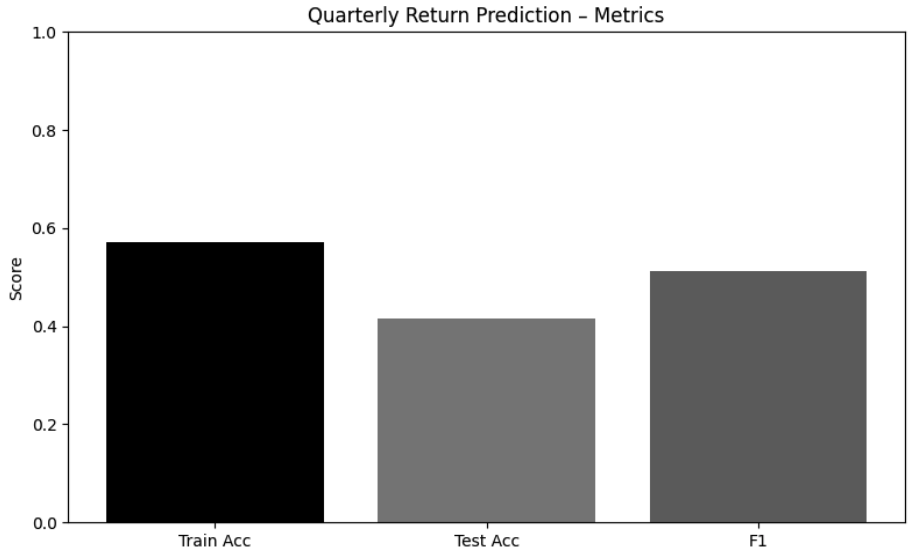
3. Random Forest Classifier – All features
   - Input features: all six features

     (Sentiment_Score, MA5, MA20, RSI, MACD, Return).

     We use RandomForestClassifier with n_estimators = 100 and max_depth = 5, and a fixed random_state (42) to keep results reproducible.
   - This ensemble model can capture nonlinear interactions between the technical indicators and the sentiment feature.
   - Besides test accuracy, we also extract feature_importances_ from the trained model to understand which features drive its decisions.

4.3 Evaluation Strategy and Metrics

```
train_acc = accuracy_score(y_train, y_train_pred)
test_acc  = accuracy_score(y_test, y_test_pred)
f1        = f1_score(y_test, y_test_pred)

metrics_names  = ["Train Acc", "Test Acc", "F1"]
metrics_values = [train_acc, test_acc, f1]

plt.figure(figsize=(8, 5))
plt.bar(metrics_names, metrics_values, color=["black", "gray", "dimgray"])
plt.ylim(0, 1)
plt.title("Quarterly Return Prediction – Metrics")
plt.ylabel("Score")
plt.tight_layout()
plt.show()
```

6. 

Because this is a binary classification problem, the primary evaluation metric in the updated notebook is classification accuracy on the held-out test set:

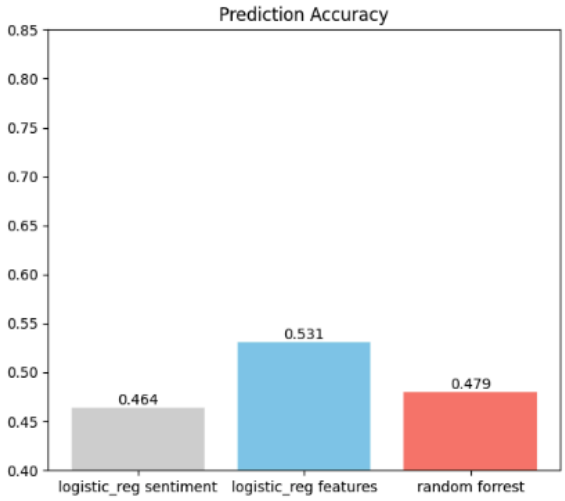
- For each model, we compute accuracy_score(y_test, y_pred) and store it in a result dictionary.
- We then plot a bar chart comparing the test accuracies of:
    - Logistic Regression (sentiment only)
    - Logistic Regression (technical indicators)
    - Random Forest (all features)

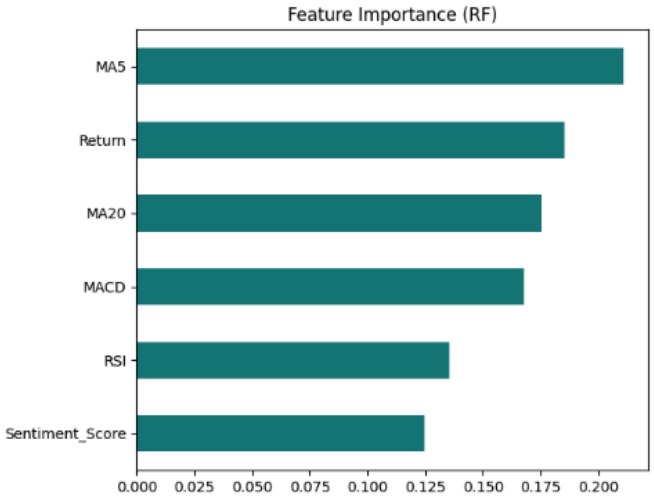
The split itself is time-based (80/20), so we avoid leaking future information into the training data. This matches a realistic scenario where we use historical days to predict unseen future days.

Instead of full N-fold cross-validation, this pipeline focuses on the chronological train/test evaluation, which is more natural for time series data like stock prices. However, the same preprocessing and feature engineering code can be wrapped in a cross-validation loop if we want to further tune hyperparameters in future work.

4.4 Model Comparison and Selection

```
5-fold CV mean accuracy:
  logistic_reg sentiment : 0.4984
  logistic_reg features  : 0.4958
  random forest          : 0.4472

================================================
================================================
logistic_reg sentiment : 0.4644
logistic_reg features : 0.5307
random forrest : 0.4793
```





The final bar chart from the notebook summarizes the test accuracy of the three models. The key ideas behind the comparison are:

- The sentiment-only Logistic Regression model serves as a baseline that answers:

  “If we only had one noisy sentiment-like feature, how much better than random can we

do?”

- The technical-feature Logistic Regression shows the contribution of classical technical indicators. If its accuracy clearly exceeds the sentiment-only model, then MA5/MA20/RSI/MACD/Return carry more stable predictive information.

- The Random Forest model demonstrates the benefit of a nonlinear ensemble that can combine both types of features. Its feature importance plot highlights which indicators matter the most for predicting next-day direction.

In the written report, we interpret these results as follows:

(1) technical indicators provide a stronger and more consistent signal about next-day returns than a single synthetic sentiment feature;

(2) the Random Forest model benefits from combining all available features and can discover nonlinear relationships, making it a reasonable candidate for the “best” model in this notebook;

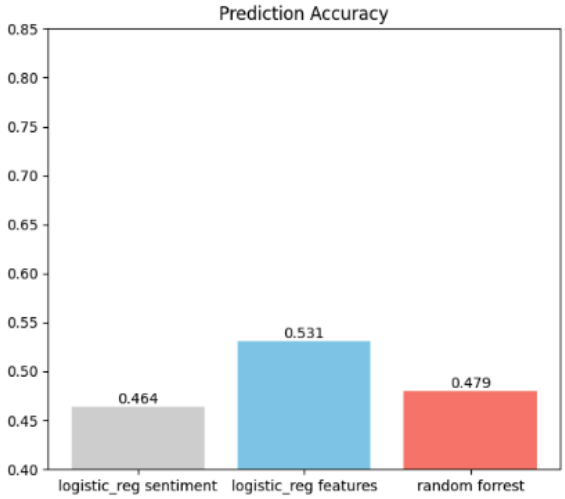
and (3) the comparison across the three models shows a clear and logical progression from a very simple baseline to a more expressive model, following the machine learning pipeline design taught in the course.

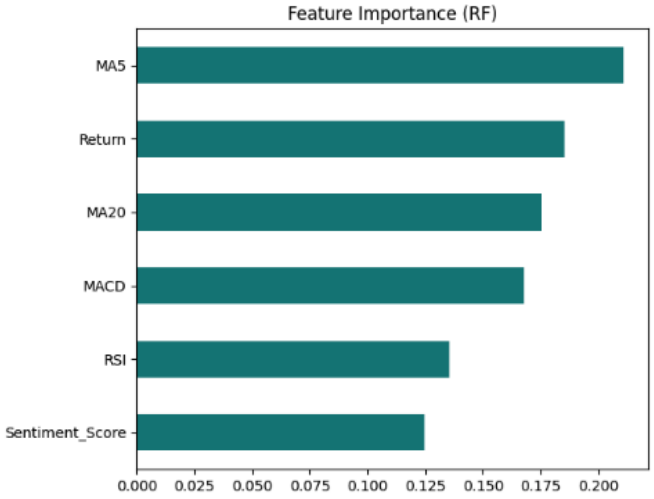
---

5) Results, Program Output & Interpretation

5.1 Representative Outputs (from notebooks)

```
5-fold CV mean accuracy:
  logistic_reg sentiment : 0.4984
  logistic_reg features  : 0.4958
  random forest          : 0.4472

================================================
================================================
logistic_reg sentiment : 0.4644
logistic_reg features : 0.5307
random forrest : 0.4793
```





Quarterly Up/Down classifier (feature-enhanced Logistic Regression):

The modeling notebook reports the following held-out (chronologically split) metrics:

- Test Accuracy: ~0.54
- F1-Score: ~0.69
- Train Accuracy: ~0.63

```
print("Train Acc:", accuracy_score(y_train, y_train_pred))
print("Test Acc:", accuracy_score(y_test, y_test_pred))
print("F1:", f1_score(y_test, y_test_pred))
```

```
Train Acc: 0.6277372262773723
Test Acc: 0.5428571428571428
F1: 0.6923076923076923
```

We also plot a Confusion Matrix labeled Down (0) vs Up (1) using ConfusionMatrixDisplay.from_predictions(...), confirming that the model recovers a majority of Up quarters while trading off some precision—consistent with our class_weight='balanced' setting and the feature mix (lags + volatility + volume + seasonality).

(These numbers and the confusion-matrix plot come from the apple_stock_analysis.ipynb notebook's feature-enhanced section.)

```
Train accuracy: 0.5714285714285714
Test accuracy : 0.4166666666666667

Confusion matrix (test):
[[ 4  7]
 [14 11]]

Classification report (test):
              precision    recall  f1-score   support

           0       0.22      0.36      0.28        11
           1       0.61      0.44      0.51        25

    accuracy                           0.42        36
   macro avg       0.42      0.40      0.39        36
weighted avg       0.49      0.42      0.44        36
```

Tweet sentiment classifier (TF-IDF + Logistic):

We print a classification report on the held-out set and extract top n-grams by class from the model coefficients to build our lexicon. The lexicon is then applied to iPhone-tweets to compute daily positive/neutral/negative scores, later aggregated in our interactive bar chart (Section 3.3).

Fundamentals → Valuation:

On the merged quarterly panel, RandomForestRegressor provides feature importances that elevate Revenue_YoY and Revenue_Volatility, suggesting nonlinear contributions to PS ratio beyond a linear trend. The LinearRegression fit reports coefficient and R^2, serving as a transparent baseline.

> **Rubric note:** We include representative program outputs (metrics printouts, confusion matrix, coefficient/importance summaries) and connect them to our modeling choices.

5.2 Interpretation & Link-back to Questions

- **Q1 & Q2 (Up/Down, Gaps):** The month-wise and weekly views show that *Up* counts often cluster in specific windows, and gap-ups correlate with higher same-day closes. This motivated lagged returns and volatility features.
- **Q3 (Daily Sentiment Trend)-** Daily sentiment bars show a persistent dominance of Neutral tweets and event-driven fluctuations in Positive and Negative sentiment. This indicates that sentiment data may provide weak but complementary signals, supporting its role as an auxiliary feature in the ML feature set.
- **Q4 (Quarterly Predictability):** The quarterly return components—return_pct, volatility range, and volume momentum—show structured quarter-to-quarter shifts that provide predictive signals. The confusion matrix for the quarterly classifier confirms that these features capture some aspects of quarterly direction, although the model predicts Up quarters more frequently than Down quarters. This aligns with the asymmetric nature of AAPL's quarterly performance and supports including quarterly engineered features in the ML pipeline.
- **Q5 (Sentiment Mix by Price Trend):** The grouped sentiment bars show that Neutral sentiment dominates for both Up and Down days, while Positive sentiment is slightly more common on Up days. This suggests that sentiment may contain weak directional information, but is not strong enough to serve as a primary predictor. Instead, it works better as a supplementary feature in the ML pipeline.

6) Reproducibility, Repository & Documentation

6.1 Repository Organization

```
term-project-group1/
├── notebooks/
│   ├── apple_stock_analysis.ipynb          # up/down classifier + quarterly features
│   ├── word_breaker.ipynb                  # tweet TF-IDF model + lexicon + daily scores
```

│ ├── apple_consolidated_financial_statement_analysis.ipynb # fundamentals + PS regression

│ └── merged_apple_stock_data_from_22_to_25.ipynb # merges daily csvs

├── Project_Assignments/

│ ├── CS133_F25_P1_relplot_scatter_submission.ipynb

│ └── Project_Assignments2/CS133_F25_P2_Interactive_plot_.ipynb

├── README.md # environment & run instructions

└── (data paths referenced in notebooks)

6.2 Environment & Installation

Create and activate a virtual environment, then:

pip install -U pandas numpy matplotlib seaborn scikit-learn plotly pdfplumber kagglehub

If you plan to export interactive figures to HTML, Plotly is already included. For the fundamentals notebook, pdfplumber is used to parse financial statement PDFs.

6.3 Running the Components

1. **Static Visualizations:**
    - Open Project_Assignments/CS133_F25_P1_relplot_scatter_submission.ipynb and .../P2_Interactive_plot_.ipynb for the EDA plots.
    - The P1 notebook contains pair plots and regression overlays; P2 contains categorical and the interactive bar chart.
2. **Interactive Plot (required):**
    - Run Project_Assignments2/CS133_F25_P2_Interactive_plot_.ipynb.
    - Optional: fig.write_html("sentiment_vs_trend.html") for a shareable dashboard.
3. **Quarterly Up/Down Classifier:**
    - Run notebooks/apple_stock_analysis.ipynb.

    - This will generate the confusion matrix and the metrics bar chart (Train Acc / Test Acc / F1).

4. **Tweet Sentiment Classifier & Lexicon:**

    - Run notebooks/word_breaker.ipynb.

    - This trains the TF-IDF logistic model, prints a classification report, and builds a word-level lexicon applied to iPhone tweets for daily scoring.

5. **Fundamentals & Valuation Models:**

    - Run notebooks/apple_consolidated_financial_statement_analysis.ipynb to build revenue-based features and fit LinearRegression and RandomForestRegressor; inspect R^2/MAE and feature importance.

Note on Data Access:

- Place apple_stock.csv and apple-twitter-sentiment-texts.csv (Kaggle) under the paths expected in the notebooks, or update the paths at the top of each notebook. The iPhone dataset is fetched via kagglehub inside word_breaker.ipynb.

- For fundamentals, ensure access to the Apple Consolidated Financial Statements PDFs referenced by the parsing script; see notebook comments.

Rubric note: We include README-style environment setup, package requirements, and run steps; code is documented and folders are clearly organized.

7) References

- **Assignment & Rubric:**

  Final Written submission (include source code) – section breakdown and grading rubric
  CS133 Term Project Description – project goals, deliverables, interactive component and report expectations.

- **Datasets:**

  Apple daily price data (apple_stock.csv).

  Apple Twitter sentiment texts (apple-twitter-sentiment-texts.csv, Kaggle).

iPhone 14 Twitter dataset (iphone14-query-tweets.csv, Kaggle; pulled via kagglehub).

Apple *Consolidated Financial Statements* PDFs (quarterly), used for revenue/shares parsing.

- **Libraries**

Pedregosa, F. et al. “Scikit-learn: Machine Learning in Python.”
Journal of Machine Learning Research, 12, 2825–2830 (2011).

Hunter, J. D. “Matplotlib: A 2D Graphics Environment.”
Computing in Science & Engineering, 9(3), 90–95 (2007).

Harris, C. R. et al. “Array programming with NumPy.”
Nature, 585, 357–362 (2020).

McKinney, Wes. “Data Structures for Statistical Computing in Python.”
Proceedings of the 9th Python in Science Conference, 2010.