S&P 500 Index Movement Prediction with Classical Machine Learning

ITI105 - Machine Learning Project - Group 04

- Isak Rabin (4466624P)
- Woo Ka Keung Alex (8148468Y)

Overview

Problem Statement

The Challenge:

- Stock traders and market analysts face difficulty making investment and trading decisions based on vast amounts of stock data and technical indicators
- Current analysis is typically manual and labor-intensive
- Decisions are often influenced by emotions or subjective judgments

Our Objective

Project Goal:

- Develop an efficient machine learning model to predict the market movement direction of the S&P 500 index

Prediction Focus:

- Directionality: Determine if the market will go Up, Down, or remain Sideways
- Model accuracy: Goal is 65% all 3 directions; target to achieve 70% after optimization
- Make use of industry widely adopted technical analysis indicators; this helps our model filter the valuable signals out of the noise

Key Benefits:

- Output prediction is more interpretable and practically useful for general analysis and decision-making
- Aims to eliminate human emotions and subjective judgments in stock market prediction

ML Formulation

Machine Learning Framing:

- Problem Type: Supervised Multiclass Classification
- 3 Classes: Up, Down, Sideways

Input and Output:

- **Input:** Model learns from historical price data and engineered technical indicators
- **Output:** A predicted class label for the next trading day
    - Up: Closing Index Price, compared to the previous day, increased above a defined threshold (e.g., ≥ +0.3%)
    - Down: Closing Index Price, compared to the previous day, decreased below a defined threshold (e.g., ≤ -0.3%)
    - Sideways: Closing Index Price, compared to the previous day, falls within a defined threshold range (e.g., between -0.3% and +0.3%)

ML Formulation (cont'd)

Type of ML Models Used (Classical):

- **Logistic Regression**: Selected as a baseline for interpretability
- **Decision Tree Classifier**: Chosen for its rule-based, interpretable structure, handling non-linear decision boundaries, and insights into feature importance
- **Random Forest Classifier**: Employed as an ensemble model to enhance predictive accuracy and mitigate overfitting
- **XGBoost Classifier** (optional): Planned for exploration, known for strong performance in classification problems

Data Preparation and Analysis

Dataset

Source Data:

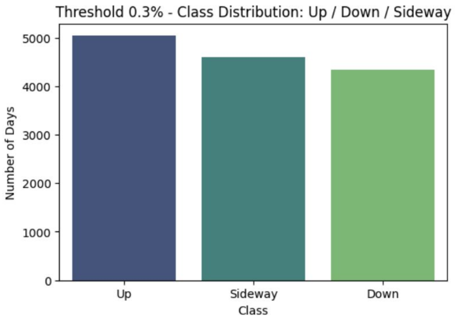
- S&P 500 Index Market historical data obtained from Yahoo Finance using the Python yfinance library
  - Time Period: 1970-01-02 to 2025-06-30, with daily frequency
  - Raw Columns: Date, Open, High, Low, Close, Adj Close, Volume
  - Total Records: 13993
  - Data Quality: No missing values or duplicate records; ensured no non-trading days; and 'Adjusted Close' was redundant with 'Close'

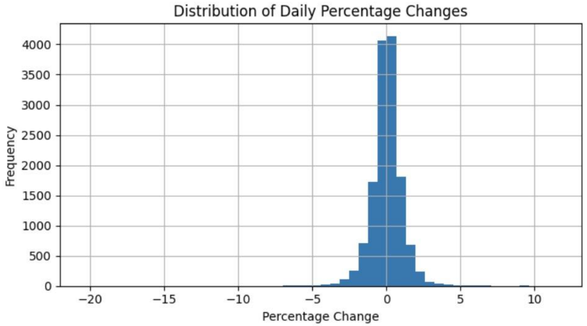
Final Prepared Data (Chronological Split):

- Training Set (73%); e.g.: ~1970 to 2010
- Validation Set (18%); e.g.: 2011 to 2020
- Test Set (8%); e.g.: 2021 to 2025

Feature Engineering

- **Target Label (movement_label):**
  Calculated based on the % change in Closing Price, classified as Up, Down, or Sideways
  - The threshold was adjusted from +/-0.5% to +/-0.3% to achieve a more balanced distribution of classes and better reflect meaningful price movements





	Close
count	13991.000000
mean	0.035904
std	1.084822
min	-20.466931
25%	-0.460576
50%	0.048271
75%	0.559791
max	11.580037

Feature Engineering (cont'd)

Final Feature Set (Engineered Columns):

- **Technical Indicators:** Selected to provide a balanced representation of trend, momentum, and volatility

Trend:

- Moving Averages (MA): 5-day, 10-day, 20-day.
- Exponential Moving Averages (EMA): EMA10, EMA20.
- Moving Average Convergence Divergence (MACD).

Momentum:

- Relative Strength Index (RSI).
- Stochastic Oscillator (%K and %D).

Volatility:

- Bollinger Bands Width.
- Average True Range (ATR)

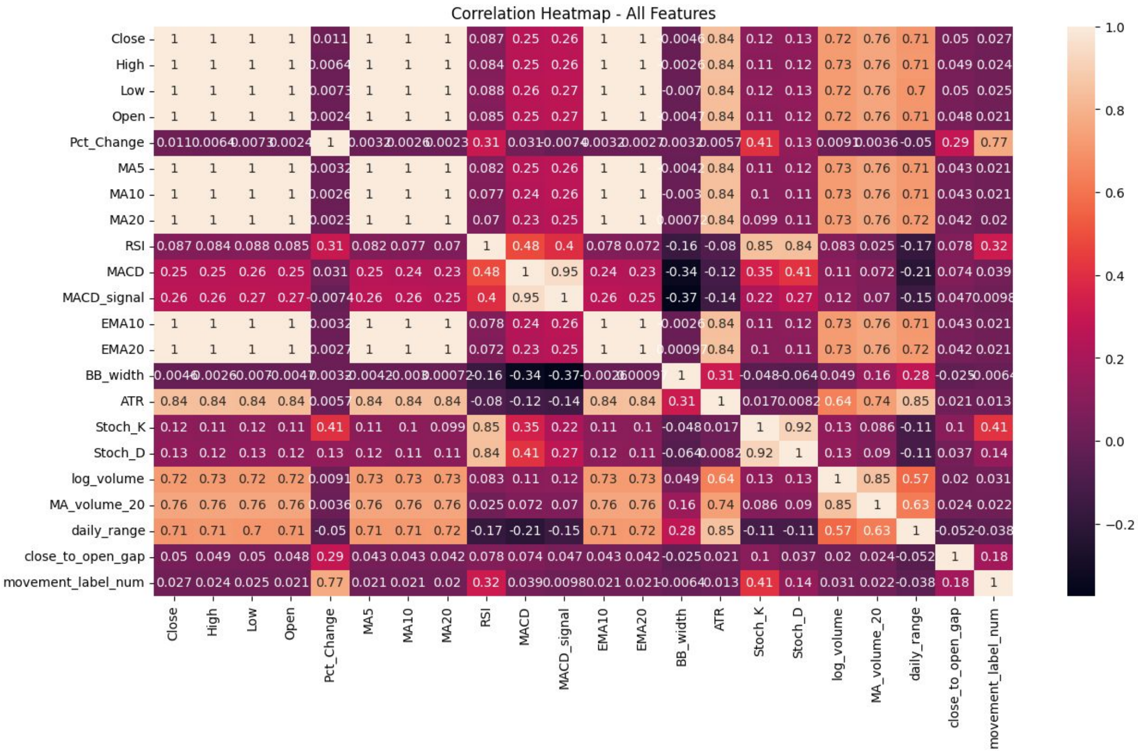
Feature Engineering (cont'd)

Additional Features:

- **Log Volume**
  - Normalising the volume, due to increasing trading volume over the decade, may impact model learning.
- **20D Moving Average Volume**
  - Daily volume can spike, to capture moving average over 20 days.
- **Daily Range**
  - To detect intraday sentiment / fluctuation (High - Low)
- **Close to Open Gap**
  - To detect overnight sentiment / fluctuation (Today Close and Tomorrow Open)

Exploratory Data Analysis

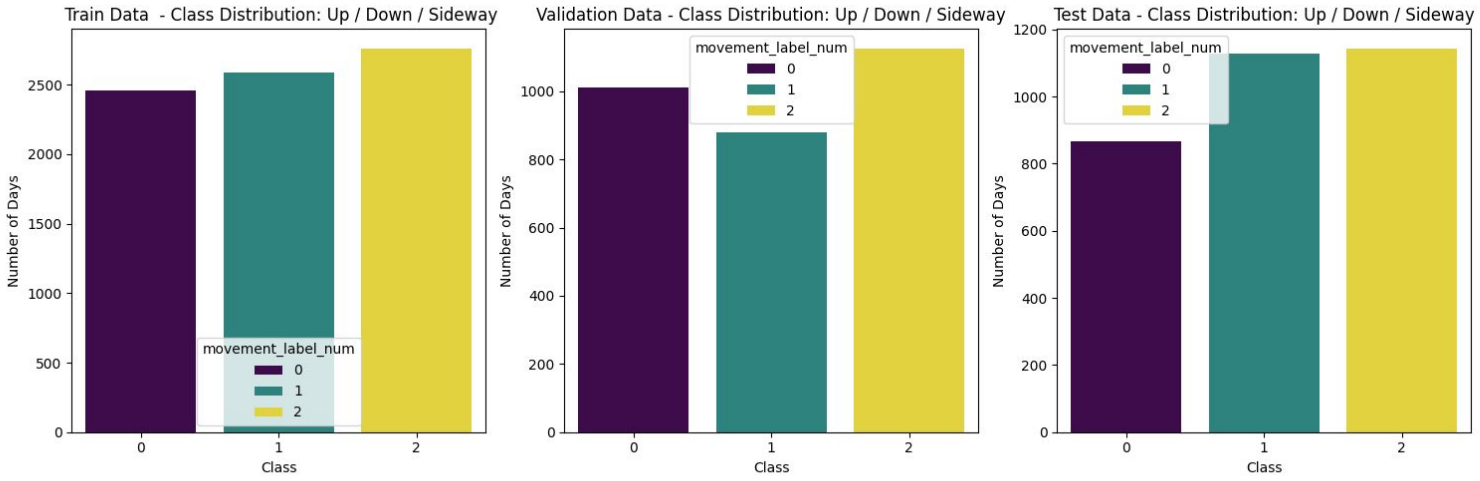
- **Feature Insights:** Identified high correlation among some engineered features (e.g., MA5, MA10, MA20 are perfectly correlated) which may cause redundancy. ATR was identified as the most informative single feature for differentiating Sideways from trending markets



Our Experiments

Experiment 1: Preliminary Steps

- Check if any imbalance in our training dataset



No imbalance data handling is required

Experiment: Input Features

- Logistic Regression (9 features)
  - Raw Features: -
  - Engineered Features: PCT_CHANGE_T, LOG_VOLUME, DAILY_RANGE, CLOSE_TO_OPEN_GAP,
  - Technical Indicators: RSI, MACD, BB_WIDTH, ATR, STOCH_K
- Decision Tree, Random Forest, XGBoost (11 features)
  - Raw Features: -
  - Engineered Features: PCT_CHANGE_T, LOG_VOLUME, DAILY_RANGE, CLOSE_TO_OPEN_GAP,
  - Technical Indicators: **MA20**, **EMA20**, RSI, MACD, BB_WIDTH, ATR, STOCH_K

Logistic Regression is sensitive to multicollinearity. MA20 and EMA20 is highly correlated to Close.

MA20, EMA20 added to Decision Tree, Random Forest, xgBoost to capture trends.

Experiment: Training & Validation

- Define each model pipeline as a baseline

```
pipe_linear = Pipeline([
    ("scaler", StandardScaler()),
    ("clf", LogisticRegression(
        max_iter=2000,  random_state=42
    ))
])

pipe_tree = Pipeline([
    ("clf", DecisionTreeClassifier(random_state=42))
])

pipe_rf = Pipeline([
    ("clf", RandomForestClassifier(random_state=42, n_jobs=-1))
])

pipe_xgb = Pipeline([
    ("clf", XGBClassifier(
        objective="multi:softprob",
        num_class=3,
        eval_metric="mlogloss",
        tree_method="hist",
        random_state=42,
        n_jobs=-1,
        subsample=1.0,
        colsample_bytree=1.0,
        n_estimators=150
    ))
])
```

- Model Training
  - Logistic Regression: StandardScaler(), 2000 iteration
  - XGBoost: 3 class, evaluation metrics mlogloss
- Evaluate Training Results with
  - Confusion Matrix, Precision, Recall, **f1-score**, **Accuracy**
- Logs the experiment with MLFlow

Experiment: Tuning

- Using GridSearchCV

```
grid_lr = {
    "clf__C": [0.01, 0.1, 1, 10],
    "clf__penalty": ["l2"],
    "clf__solver": ["lbfgs", "saga"]
}

grid_dt = {
    "clf__max_depth": [3, 5, 7],
    "clf__min_samples_split": [0.01, 0.02],
    "clf__min_samples_leaf": [0.005, 0.01],
    "clf__criterion": ["gini"],
}

grid_rf = {
    "clf__n_estimators": [200, 300],
    "clf__max_depth": [None, 12],
    "clf__max_features": ["sqrt"],
    "clf__min_samples_leaf": [1, 2, 5],
    "clf__min_samples_split": [2, 5, 10]
}

grid_xgb = {
    "clf__n_estimators": [200, 400],
    "clf__learning_rate": [0.03, 0.1],
    "clf__max_depth": [4, 6],
    "clf__subsample": [0.8, 1.0],
    "clf__min_child_weight": [1, 3],
    "clf__colsample_bytree": [0.8, 1.0],
    "clf__reg_lambda": [0.5, 1.0, 2.0],
    "clf__reg_alpha": [0, 0.1]
}
```

- Time series aware
- Use same dataset as initial Training to see improvements
- Tuning Parameters
  - Logistic Regression: C, solver
  - Decision Tree, Random Forest: max_depth, min_samples_split, min_samples_leaf
  - xgBoost: learning_rate, max_depth,min_child_weight, subsample, colsample_bytree, reg_lambda/alpha
- Use same evaluation metrics with initial Training
- Logs the experiment with MLFlow

Experiment: Results

- Training Results

	Model	n_features	Train macro-F1 (base)	Train acc (base)	Val macro-F1 (base)	Val acc (base)
0	Logistic	9	0.370095	0.39715	0.379742	0.415177
1	XGBoost	11	0.932868	0.93292	0.364253	0.363528
2	Random Forest	11	1.000000	1.00000	0.350652	0.358760
3	Decision Tree	11	1.000000	1.00000	0.312244	0.315058

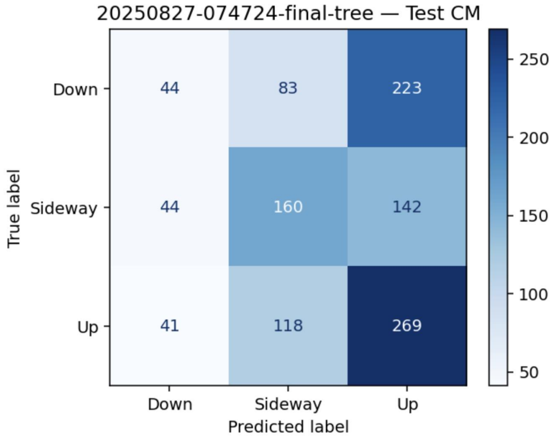
- After Tuning

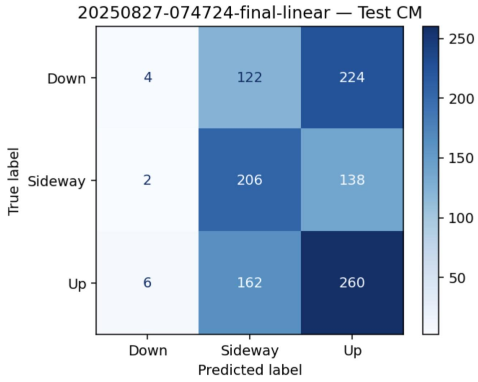
	Model	n_features	Train macro-F1 (tuned)	Train acc (tuned)	Val macro-F1 (tuned)	Val acc (tuned)
0	20250827-074445-tuning-linear	9	0.370066	0.396762	0.379742	0.415177
1	20250827-074445-tuning-rf	11	0.737778	0.739531	0.389706	0.389352
2	20250827-074445-tuning-tree	11	0.436117	0.438639	0.357445	0.374652
3	20250827-074445-tuning-xgb	11	0.933921	0.933792	0.344953	0.347239

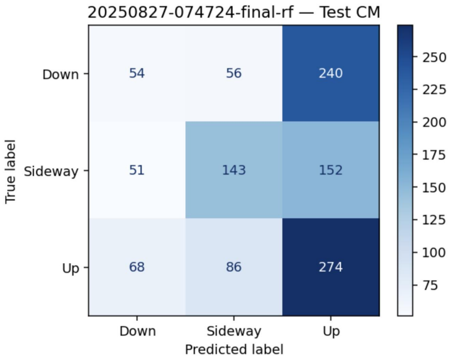
Experiment: Results (cont’d)

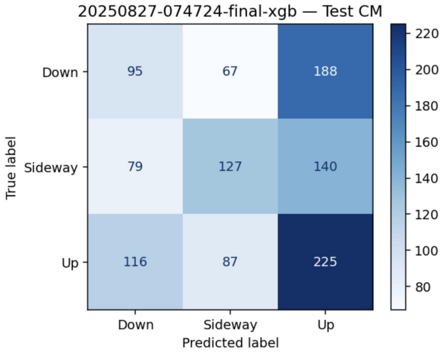
- Test Results

	Model	n_features	Test macro-F1 (test)	Test acc (test)
0	XGBoost	11	0.386898	0.397687
1	Random Forest	11	0.386888	0.419039
2	Decision Tree	11	0.380975	0.420819
3	Logistic	9	0.336720	0.418149









Experiment: Summary

- Training
  - **Trees massively overfit**: Decision Tree & Random Forest show **Train F1/Acc = 1.00**, but **Val macro-F1 ≈ 0.3122–0.3506** ; showing high-variance behavior.
  - **XGBoost** also overfits (Train F1 = 0.9328) with Val F1 = 0.3642
  - **Logistic Regression** is stable but underfits: Train/Val macro-F1 = **0.3700–0.3797**, Acc ≈ **0.3971–0.4151** (no overfitting, but limited capacity).
- After Tuning
  - **RF improves the most on validation**: Val macro-F1 **0.3507** to **0.3897** (+0.039), Val acc **0.3588** to **0.3893** (+0.030). Training scores drop (which is good, less overfit).
  - **Decision Tree** also improves: Val macro-F1 **0.3122** to **0.3574** (+0.045).
  - **XGBoost** validation slightly **drops** (0.3642 to 0.3449), still high train scores, meaning some overfit remains.
  - **Logistic** unchanged (tuning didn't move the needle).

Experiment: Summary (cont'd)

- Test
  - **Macro-F1 (primary)**
    - **XGBoost = 0.3868, Random Forest 0.3868** > Decision Tree 0.3809 > Logistic 0.3367.
      Best balanced performance: **XGB or RF (essential tie)**.
  - **Accuracy:**
    - Decision Tree **0.4208** ≥ Random Forest **0.4190** ≥ Logistic **0.4181** > XGBoost **0.3978**.
      DT likely predicts the majority class a bit more, boosting accuracy but not macro-F1.
  - **Gaps are small**: all models land in a narrow band; improvements over chance are modest.

Implementation Walkthrough
Demo

https://huggingface.co/spaces/sgirabin/iti105-machine-learning-project

Settings

Ticker (Yahoo Finance)

^GSPC

Start date

2015/01/01

End date

2025/08/27

Movement threshold ±% (daily close)

0.30

Choose model

Model

Logistic Regression

Data source

Use uploaded CSV instead of Yahoo Finance

Artifacts directory

artifacts

Expected files (optional): logreg.joblib, decision_tree.joblib, random_forest.joblib, xgboost.joblib/xgboost.json, feature_columns.json

Run

📈 Stock Movement Predictor — Classical ML (Streamlit)

ITI105 project demo • Predict next-day **Up** / **Sideways** / **Down** using technical indicators and classical ML models.

Author: Isak Rabin (4466624P), Woo Ka Keung Alex (8148468Y)

Tip: If a model artifact isn't found for your selection, the app will train a quick demo model on the downloaded data.

Recent predictions

Date	Close	TrueLabel	Predicted	P(Down)	P(Sideway)	P(Up)
2025-07-15 00:00:00	6243.7598	Up	Up	0.2734	0.312	0.4146
2025-07-16 00:00:00	6263.7002	Up	Up	0.2757	0.3086	0.4157
2025-07-17 00:00:00	6297.3599	Sideway	Up	0.2494	0.3458	0.4049
2025-07-18 00:00:00	6296.79	Sideway	Up	0.2378	0.3695	0.3927
2025-07-21 00:00:00	6305.6001	Sideway	Sideway	0.2325	0.3859	0.3816
2025-07-22 00:00:00	6309.6201	Up	Sideway	0.2323	0.3885	0.3792
2025-07-23 00:00:00	6358.9102	Sideway	Sideway	0.2235	0.4352	0.3412
2025-07-24 00:00:00	6363.3501	Up	Sideway	0.2037	0.4572	0.3391
2025-07-25 00:00:00	6388.6401	Sideway	Sideway	0.1978	0.4753	0.3269
2025-07-28 00:00:00	6389.77	Sideway	Sideway	0.1932	0.4969	0.3099

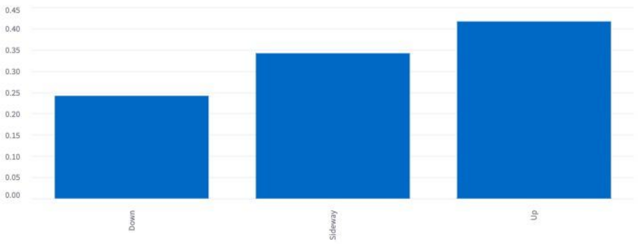
Model source: **quick-train** • Features used: 13

Close price (last 200)

Latest prediction

Predicted next-day movement

Up



Key Takeaways
and Future Improvements

Conclusions

- **XGBoost, Random Forest, and Decision Tree** achieved macro-F1 ~ 0.38 on the held-out test set, modestly above a 0.33 baseline (random guessing)
- Tree ensembles (RF/XGBoost) generalized best on balanced metrics, while a constrained Decision Tree had the highest accuracy, likely due to majority-class bias.
- Tuning primarily reduced overfitting in tree models; Logistic Regression remained stable but underfit.
- Based on our experiment, we recommend **Random Forest** (or **XGBoost**) as the primary model, reporting macro-F1 as the main metric and accuracy as secondary.

Future Works

- Tune further with different parameters
- Add analysis to identify feature importance
- Implement Stacking or Voting to check whether it can help to improve model performances
- Trying out with other Machine Learning, such SVM

Thank You