

# Final Report: S&P 500 Index Movement Prediction with Classical Machine Learning

Name	Student ID
Isak Rabin	4466624P
Woo Ka Keung Alex	8148468Y

## 1. Project Overview *(Contributor: Isak Rabin, Woo Ka Keung Alex)*

Our project addresses the challenge faced by Stock traders and market analysts in making predictions on how to make investment and trading decisions based on vast amounts of stock data and technical indicators. This analysis is typically manual and labour intensive, and most of the time influenced by emotions or subjective judgements.

We aim to develop a machine learning model that is able to predict the **market movement direction of S&P 500 index**, whether it will go **up**, **down**, or remain **sideways**, using **historical price data** and **technical indicators**. By predicting market movement direction instead of forecasting the exact closing price, it makes the output prediction become more interpretable and practically useful for general analysis and decision making.

## 2. ML Formulation *(Contributor: Isak Rabin, Woo Ka Keung Alex)*

This project formulates the task of predicting S&P 500 index movement direction as a **supervised multiclass classification problem**. Instead of forecasting the exact closing price, we categorize next-day market movement into three classes:

- **Up**: When the closing Index Price, compared to previous day, increased above defined threshold
- **Down**: When the closing Index Price, compared to previous day, decreased below defined threshold
- **Sideways**: When the closing Index Price, compared to previous day, falls within a defined threshold range.

The model will learn from historical price data and technical indicators to predict the movement label for the following trading day.

Machine Learning Problem Formulation	
Problem Type	Supervised Multiclass classification. (3 classes: <b>up</b> , <b>down</b> , <b>sideways</b> )
Ideal Outcome	1. The model reliably predicts the correct movement direction for the S&P 500 index with high accuracy and interpretability. 2. The deployed application is usable and reflects predictions based on recent data and potentially real-time data.
Success Metrics (non-technical):	1. Achieve a minimum of <b>65%</b> accuracy, with a target of <b>70%</b> . 2. False positive rate for “ <b>Up</b> ” and “ <b>Down</b> ” predictions remains below <b>25%</b> . 3. Consistent <b>F1-score</b> above <b>0.60</b> for each class (Up, Down, Sideways). 4. The deployed application is usable and reflects predictions based on recent or live data.

Machine Learning Problem Formulation	
	With the above criteria, the model will be able to help to eliminate human emotions and subjective judgements in stock market prediction.
Failure Conditions:	<ol style="list-style-type: none"> <li>1. Accuracy consistently below <b>33%</b> (i.e., worse than random guessing for 3 classes)</li> <li>2. False positive rate for majority “<b>Up</b>” and “<b>Down</b>” predictions is exceeding <b>40%</b>.</li> <li>3. One or more classes are poorly predicted (F1-score &lt; <b>0.4</b>).</li> <li>4. The model suffers from data leakage or overfitting despite tuning.</li> <li>5. Final models cannot generalize to unseen time periods (poor temporal robustness)</li> </ol>
Model Output	A predicted class label: <b>Up</b> , <b>Down</b> , or <b>Sideways</b> for the next trading day.
Use of Output:	The output model can be used and integrated into <b>market prediction tools or applications</b> . This application allows users to input recent market data and receive a predicted <b>movement direction</b> (Up, Down, or Sideways) for the next trading day.
Heuristic Baseline (if no ML)	<p>If we did not use machine learning, a rule-based system could classify movement using the difference between today's and yesterday's closing prices, as an example:</p> <ul style="list-style-type: none"> <li>• If % index price change <math>\geq</math> <b>Upper Threshold</b> <math>\rightarrow</math> <b>Up</b></li> <li>• If % index price change <math>\leq</math> <b>Lower Threshold</b> <math>\rightarrow</math> <b>Down</b></li> <li>• Else <math>\rightarrow</math> <b>Sideways</b></li> </ul> <p>This basic rule will serve as a benchmark for evaluating whether ML models are truly adding value.</p>

### 3. Data Preparation and Feature Engineering

#### 3.1. Dataset Overview *(Contributor: Isak Rabin, Woo Ka Keung Alex)*

We obtained historical data of S&P 500 Index Market from **Yahoo Finance** as our primary dataset. We used the Python **yfinance** library to help us download the data.

Dataset Description		
Source	Yahoo Finance	
Time Period	02-01-1970 to 30-06-2025	
Frequency	Daily	
Raw Columns		
	Column Name	Description
	Date	Trading Date
	Open	Opening Price
	High	Highest Price
	Low	Lowest Price

Dataset Description			
	Close	Closing Price	
	Adj Close	Adjusted Close Price	
	Volume	Transaction Volume	

### 3.2 First EDA (Before Feature Engineering) *(Contributor: Isak Rabin, Woo Ka Keung Alex)*

We need to understand our data to gain some insight before conducting further processing. This step helped us identify data characteristics, detect any anomalies in the dataset, guide the data cleaning and pre-processing before the actual model training.

First, we would like to understand the structure and characteristics of our dataset.

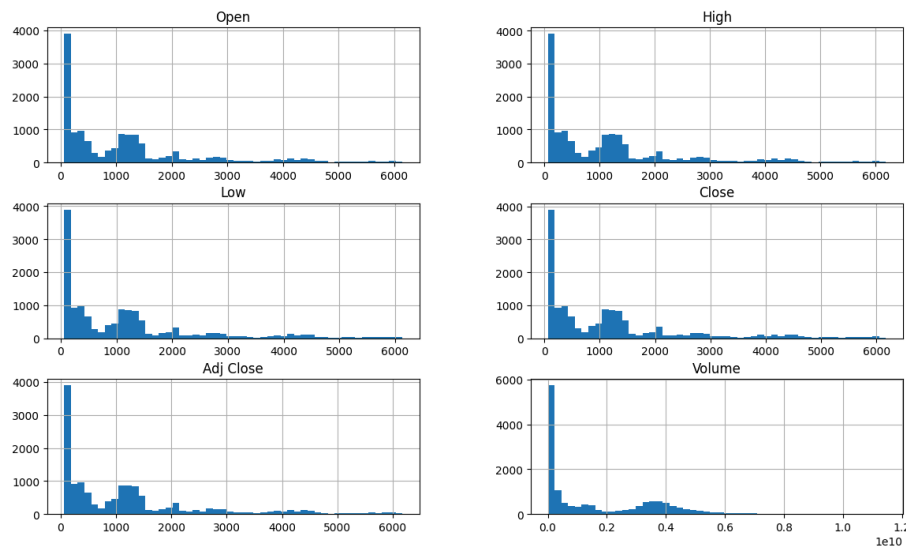
Number of Rows	13993																																								
Structure of Data	<div>Data columns (total 6 columns):</div> <table><thead><tr><th>#</th><th>Column</th><th>Non-Null Count</th><th>Dtype</th></tr></thead><tbody><tr><td>0</td><td>Adj Close</td><td>13993 non-null</td><td>object</td></tr><tr><td>1</td><td>Close</td><td>13993 non-null</td><td>object</td></tr><tr><td>2</td><td>High</td><td>13993 non-null</td><td>object</td></tr><tr><td>3</td><td>Low</td><td>13993 non-null</td><td>object</td></tr><tr><td>4</td><td>Open</td><td>13993 non-null</td><td>object</td></tr><tr><td>5</td><td>Volume</td><td>13993 non-null</td><td>object</td></tr></tbody></table>	#	Column	Non-Null Count	Dtype	0	Adj Close	13993 non-null	object	1	Close	13993 non-null	object	2	High	13993 non-null	object	3	Low	13993 non-null	object	4	Open	13993 non-null	object	5	Volume	13993 non-null	object												
#	Column	Non-Null Count	Dtype																																						
0	Adj Close	13993 non-null	object																																						
1	Close	13993 non-null	object																																						
2	High	13993 non-null	object																																						
3	Low	13993 non-null	object																																						
4	Open	13993 non-null	object																																						
5	Volume	13993 non-null	object																																						
Statistics of Data	<div>Summary statistics:</div> <table><thead><tr><th></th><th>Adj Close</th><th>Close</th><th>High</th></tr></thead><tbody><tr><td>count</td><td>13993</td><td>13993</td><td>13993</td></tr><tr><td>unique</td><td>12740</td><td>12740</td><td>12738</td></tr><tr><td>top</td><td>102.08999633789062</td><td>102.08999633789062</td><td>105.41000366210938</td></tr><tr><td>freq</td><td>7</td><td>7</td><td>7</td></tr></tbody></table> <table><thead><tr><th></th><th>Low</th><th>Open</th><th>Volume</th></tr></thead><tbody><tr><td>count</td><td>13993</td><td>13993</td><td>13993</td></tr><tr><td>unique</td><td>12747</td><td>12761</td><td>12368</td></tr><tr><td>top</td><td>100.3499984741211</td><td>102.08999633789062</td><td>4223740000</td></tr><tr><td>freq</td><td>6</td><td>8</td><td>12</td></tr></tbody></table>		Adj Close	Close	High	count	13993	13993	13993	unique	12740	12740	12738	top	102.08999633789062	102.08999633789062	105.41000366210938	freq	7	7	7		Low	Open	Volume	count	13993	13993	13993	unique	12747	12761	12368	top	100.3499984741211	102.08999633789062	4223740000	freq	6	8	12
	Adj Close	Close	High																																						
count	13993	13993	13993																																						
unique	12740	12740	12738																																						
top	102.08999633789062	102.08999633789062	105.41000366210938																																						
freq	7	7	7																																						
	Low	Open	Volume																																						
count	13993	13993	13993																																						
unique	12747	12761	12368																																						
top	100.3499984741211	102.08999633789062	4223740000																																						
freq	6	8	12																																						
Data Quality Checks	<div><div><div>No missing value</div><pre>1 # Check for missing values 2 raw_data.isnull().sum()</pre><div>0</div><div><div>Adj Close</div><div>0</div></div><div><div>Close</div><div>0</div></div><div><div>High</div><div>0</div></div><div><div>Low</div><div>0</div></div><div><div>Open</div><div>0</div></div><div><div>Volume</div><div>0</div></div></div></div> <div>Since there is no missing value, it can also imply that the</div>																																								

dataset also does not include non-trading days data.

- **No duplicate records were found**  
Duplicate check is important to avoid model memorizing instead of generalising.

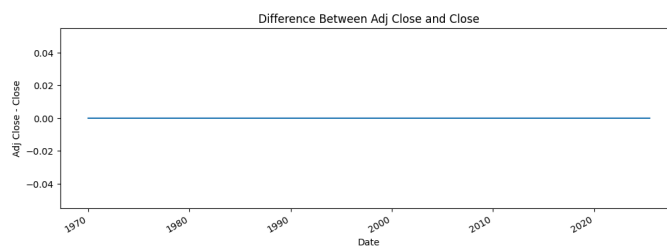
Next, we would like to understand the distribution of our raw features: **Open, High, Low, Close, Adjusted Close, and Volume**.

S&P 500 - Raw Feature Distributions



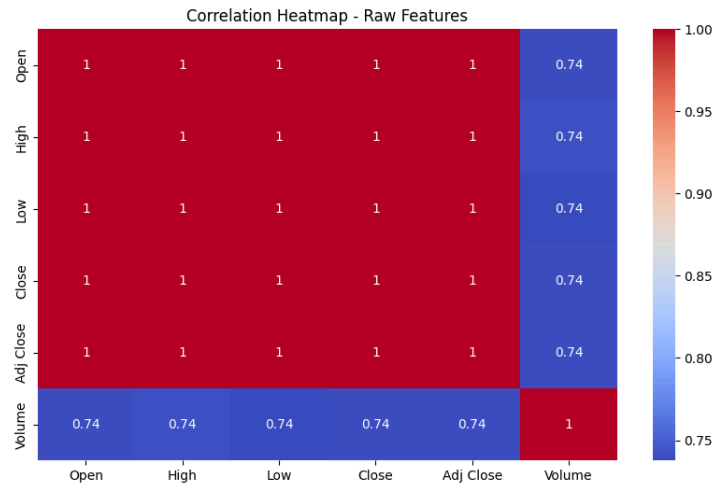
Based on above histogram, we analysed that

- **The dataset is a right skewed distribution.**  
This confirms that our raw dataset contains non-uniform distribution and potential outliers, which need to be addressed through scaling, normalization, or transformation during preprocessing to improve model performance.
- **Adjusted Close price closely mirroring the Close price.**  
To confirm this, we conducted further analysis, and we confirmed that both actually contain the same value.



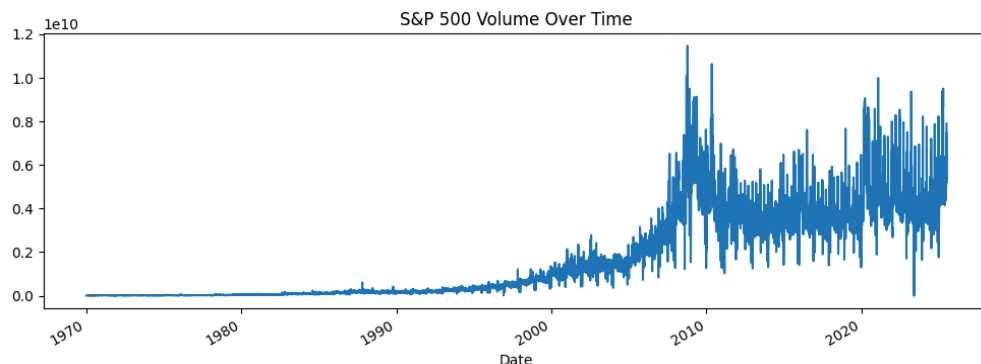
This is not common in the actual Index Market, so probably our dataset has been adjusted from the sources. **To avoid redundancy, we can drop one of them.**

We would also want to understand the correlation between raw features in our dataset. To do so, we plotted a heatmap diagram to illustrate the relationship between raw features in our dataset.

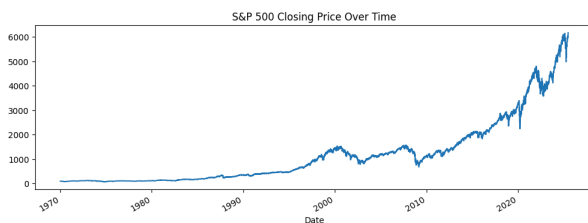


Referring to the heatmap diagram above, it shows almost perfect correlation (value of 1.0) among all prices related features, except for the **Volume**. This strong correlation among the price features needs to be considered in **selecting the feature columns for model training to avoid multicollinearity** associated with the regression model.

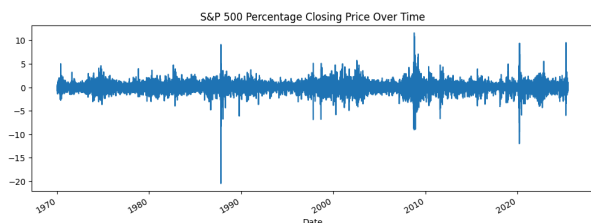
**Volume shows a moderate positive correlation with price related features (0.74), indicating that the increase in trading activity (volume) may cause the increase in price, and less likely to cause the price to decrease.**



Above graph showing S&P 500 trading volume over our dataset. It showed the scale of trading volume change dramatically over decades. It had relatively low and stable volumes in the 1970s-1990s, followed by significant growth from the 2000s onwards. This non-stationary behavior means that **using raw volume directly as a feature would bias the model towards recent years**, where the volumes are naturally larger. We need to **apply log transformation or normalize volume** to make it comparable across the time periods.



**S&P 500 Closing Price Trend**

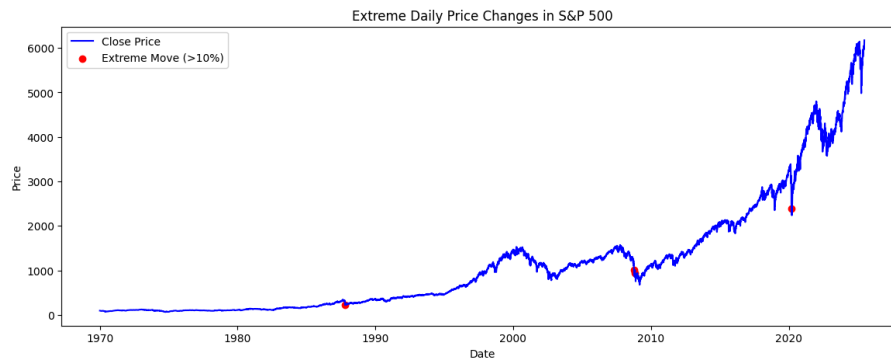


**S&P 500 Closing Price Percentage Change Trend**

Similar to the volume, the Closing Price of S&P 500 also shows a clear long-term upward trend. It means that their **mean and variance increase over time**, and using them directly as a feature would bias the model toward recent years. For example, an increase of 50 points in the 1970s might be highly significant, but not in recent years. For the training, we need to use **relative market price movement** instead of the

raw Closing Price.

We also analysed the price changes between two consecutive trading days to identify whether there were any extreme price changes. This would help us to identify the outliers in our dataset.



The above chart shows that there were only a few instances of extreme price movement throughout the entire dataset period. This means, **we do not need to exclude any outliers in our dataset.**

### 3.3. Feature Engineering

#### 3.3.1. Label Creation *(Contributor: Isak Rabin, Woo Ka Keung Alex)*

To convert the problem into a classification task, we created a new column **movement\_label**, which is our **target variable** for classification.

We calculated the value of **movement\_label** by comparing the difference between today's and tomorrow's closing prices with a certain **threshold** percentage.

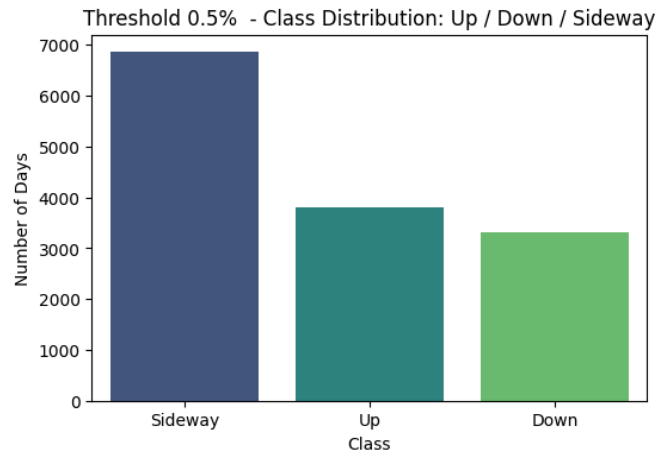
The percentage change in closing price between the current day (t) and the next trading day (t+1) is calculated by using this formula:

$$\text{Percentage Change} = \frac{\text{Close}(t+1) - \text{Close}(t)}{\text{Close}(t)} * 100$$

This **threshold** percentage value is important because it may impact the balance distribution of **movement\_label** from our dataset, thus we experimented with different threshold percentages to find the most balanced distribution.

First, we tried with the following threshold definition:

- Up → % price change  $\geq +0.5\%$ .
- Down → % price change  $\leq -0.5\%$ .
- Sideway → % price change is between **-0.5% to +0.5%**

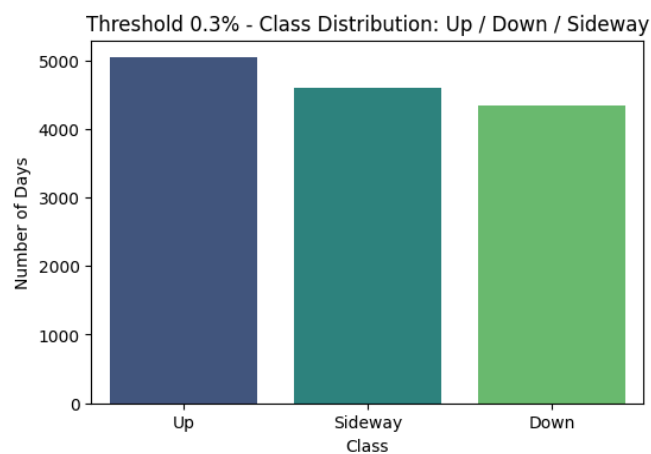


With the threshold of **+/- 0.5%**, the dataset gave an imbalance distribution of the target variable (majority class is Sideway, about +/- 60% of the dataset).

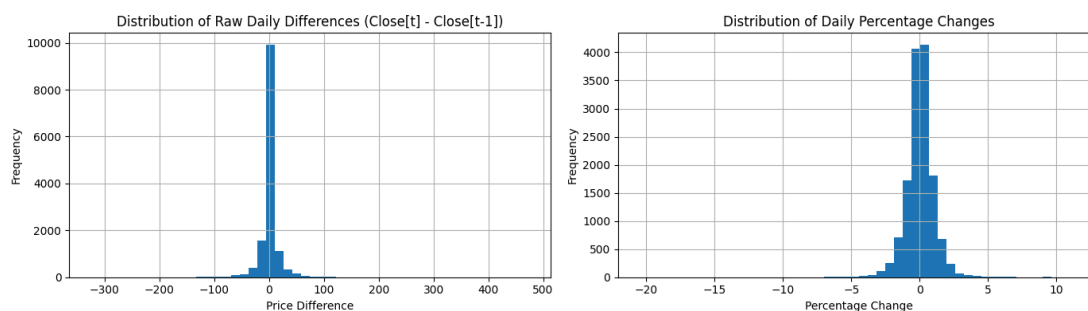
To overcome this imbalance data, we considered two options:

- Modify the percentage points to find more balance dataset
- Apply imbalance handling, such as SMOTE, on the training data to make training data more balanced

We tried to modify our threshold definition to **+/-0.3%**, and we found fewer days get classified as “Sideways.”, make more **balanced the distribution** of classes, making training more effective (avoiding a classifier that just predicts “Sideways” most of the time).



To confirm this threshold, we conducted further analysis on Close Price change distributions (daily difference between consecutive trading days).



As can be observed from the above graph, the distribution is heavily centered around 0, meaning that most daily changes are small, the bulk of it fall within **-100 to + 100 index points**.

Distribution - Percentage Change of Closing Price																	
<div> <div>Close</div> <table> <tr><td>count</td><td>13991.000000</td></tr> <tr><td>mean</td><td>0.035904</td></tr> <tr><td>std</td><td>1.084822</td></tr> <tr><td>min</td><td>-20.466931</td></tr> <tr><td>25%</td><td>-0.460576</td></tr> <tr><td>50%</td><td>0.048271</td></tr> <tr><td>75%</td><td>0.559791</td></tr> <tr><td>max</td><td>11.580037</td></tr> </table> </div>	count	13991.000000	mean	0.035904	std	1.084822	min	-20.466931	25%	-0.460576	50%	0.048271	75%	0.559791	max	11.580037	<p>This is also aligned with the typical behavior of the S&amp;P 500 Index. As we can see on the left table</p> <ul style="list-style-type: none"> <li>On average, the S&amp;P 500 Index increases by <b>~0.0359%</b> per day.</li> <li><b>50% of the days</b> have a <b>price range change between -0.4605% to +0.5597%</b>.</li> <li>25% of the days had returns <b>worse than -0.46%</b>. (25th percentile)</li> <li>25% of the days had returns <b>better than +0.56%</b>. (75th percentile)</li> </ul>
count	13991.000000																
mean	0.035904																
std	1.084822																
min	-20.466931																
25%	-0.460576																
50%	0.048271																
75%	0.559791																
max	11.580037																

This pattern justifies our adjustment from a **+/-0.5%** to a **+/-0.3%** threshold for defining Up/Down/Sideways movements. It reflects meaningful price movements across both calm and volatile market conditions, resulting in a more balanced label distribution.

### 3.3.2. Technical Indicators *(Contributor: Isak Rabin, Woo Ka Keung Alex)*

The model will use the combination of raw price data and engineered technical indicators to make predictions. This combination will enable the model to learn rich patterns associated with market direction changes while remaining interpretable and grounded in traditional financial analysis.

We selected Technical Indicators that provide a balanced representation of:

- Trend strength and direction
- Momentum indicators that often precede price changes
- Volatility patterns that reflect uncertainty and risk

Here is the list of various Technical Indicators that are mostly in use for analysing stock market movement.

Features	Type	Reason for Selection
MA5, MA10, MA20	Trend	Moving Averages smooth out noise and indicate price trends
RSI (Relative Strength Index)	Momentum	Helps identify overbought or oversold conditions
MACD (Moving Average Convergence Divergence)	Momentum/Trend	Captures shifts in momentum and possible trend reversals
EMA10, EMA20	Trend	Assigns more weight to recent prices; captures short-term shifts
Bollinger Band Width	Volatility	Measures standard deviation-based volatility
ATR (Average True Range)	Volatility	Measures daily range and market uncertainty
Stochastic Oscillator	Momentum	Compares closing price to price range over time; captures turning points



### 3.3.3. Additional Features

*(Contributor: Isak Rabin)*

Referring to our initial analysis of our raw dataset, we have identified log transformation or normalisation is required, such as on Volume and Closing Price. Hence, we introduce a new features, such as:

Additional Features	
<b>Log Volume</b>	<ul style="list-style-type: none"> <li>Normalising the volume, due to increasing trading volume over the decade, may impact model learning.</li> </ul>
<b>20D Moving Average Volume</b>	<ul style="list-style-type: none"> <li>Daily volume can spike, to capture moving averages over 20 days.</li> </ul>
<b>Price Change Percentage</b>	<ul style="list-style-type: none"> <li>Normalising the Closing Price, due to increasing price over the decade</li> <li>To avoid leakage, the percentage is calculated with between <math>(\text{Close}(t) - \text{Close}(t-1)) / \text{Close}(t)</math>, a slightly different formula used to assign the movement label.</li> </ul>
<b>Daily Range</b>	<ul style="list-style-type: none"> <li>To detect intraday sentiment / fluctuation (High - Low)</li> </ul>
<b>Close to Open Gap</b>	<ul style="list-style-type: none"> <li>To detect overnight sentiment / fluctuation (Today Close and Tomorrow Open)</li> </ul>

*(Contributor: Woo Ka Keung Alex)*

Additional features that are specifically useful for directionality prediction, as our model focuses on directions rather than the actual values.

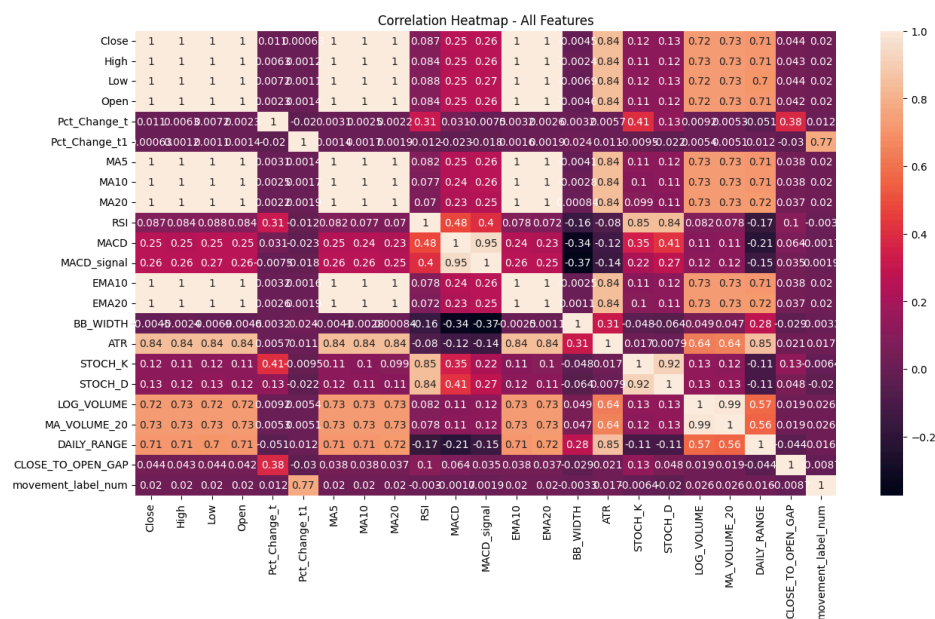
Directionality Feature	Explanation
MACD Histogram ( <b>MACD_Hist</b> )	Measures the momentum between the MACD line and the Signal line. Positive = Bullish momentum. Negative = Bearish momentum.
<b>MACD_Above_Zero</b>	Binary feature indicating whether the MACD line is above or below zero. 1 = Bullish trend (MACD > 0). 0 = Bearish trend (MACD < 0).
Average Directional Index ( <b>ADX_Strong_Trend</b> )	ADX measures the strength of a trend, regardless of uptrend or downtrend. 1 if ADX > 25: Indicates a strong trend is in place.
<b>ADX_Uptrend_DI</b>	ADX positive directional indicator (+DI). 1 if +DI > -DI.
<b>ADX_Downtrend_DI</b>	ADX negative directional indicator (-DI). 1 if -DI > +DI.
On-Balance Volume ( <b>OBV</b> ) and New High ( <b>OBV_New_High</b> )	A cumulative indicator that links volume to price changes. If the S&P 500 is in an uptrend and the OBV also making new highs, this confirms that the uptrend is being supported by strong buying volume. A divergence (price making new highs while OBV does not) can signal a weak trend.
Stochastic Crossover ( <b>Stoch_Crossover</b> )	The faster-moving %K line crossing above the slower-moving %D line is a bullish signal, and vice-versa. 1 → bullish crossover (%K goes from below %D to above %D)

	-1 → bearish crossover (%K goes from above %D to below %D) 0 → no bullish crossover that day
Moving Averages Crossing ( <b>MA_Cross</b> )	1 → Golden Cross: 50-day MA crosses above 200-day MA. This is a key bullish signal, suggesting a new uptrend is beginning. -1 → Death Cross: 50-day MA crosses below 200-day MA. This is a bearish signal, indicating that a downtrend may be starting. 0 → no crossover on that day.

### 3.4 Second EDA (After Feature Engineering)

(Contributor: Isak Rabin)

To identify the optimal set of features before proceeding to model training, we performed another Exploratory Data Analysis to understand the importance of newly added features (engineered features).



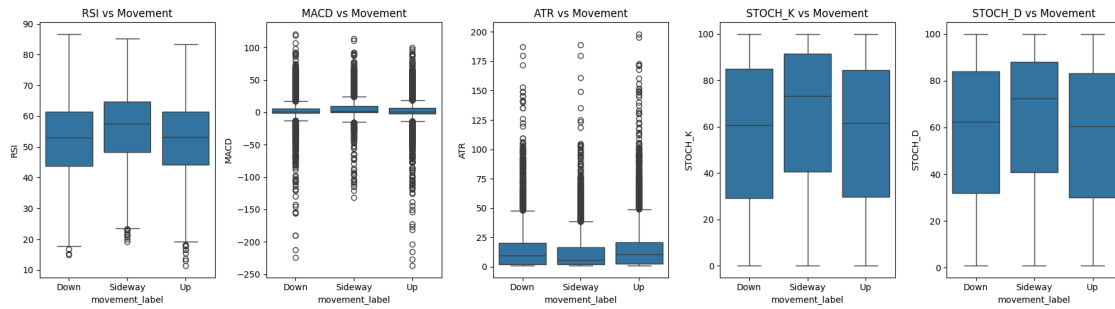
Based on above heatmap chart, we identified that:

- There is no feature showing strong correlation to our movement label (our target label). It means, there is no single feature that strongly predicts the next day's movement on its own.
- Strong Correlation
  - MA5, MA10, and MA20 are perfectly correlated
  - EMA10 and EMA20 are perfectly correlated
  - There are similar correlations between Stoch\_D and Stoch\_K indicators (0.92), and same for MACD and MACD\_Signal (0.95) indicators.

Similar to earlier EDA on raw data, we may need to consider **dropping these features** because they are identical or highly correlated. This redundancy can lead to **multicollinearity problems** in our Logistic Regression model, making coefficients unstable and less interpretable. Even in non-linear models such as Decision Trees, strong feature correlations may reduce the diversity of splits and thus limit the model's ability to generalize effectively.

- Less Correlation
  - BB\_width is less correlated with most other features (low absolute values). This suggests it could provide unique information for volatility that is not redundant.

Next, we want to illustrate the distribution of technical indicators (RSI, MACD, ATR, Stochastic K, and Stochastic D) across the three movement classes (Down, Sideways, Up).

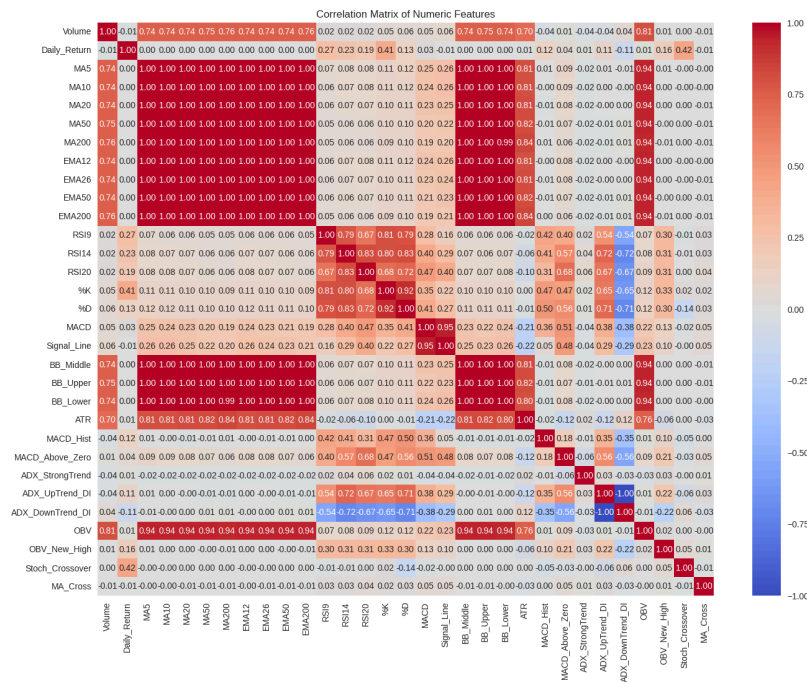


Most indicators (RSI, MACD, Stochastic K/D) display overlapping ranges and similar medians, offering little separation between classes. MACD shows strong outliers, reflecting noise. ATR is the most informative: Sideways markets have lower ATR, while trending markets (Up/Down) show higher volatility, consistent with financial intuition.

Overall, we can see that there is no single indicator that is predictive on its own, suggesting that effective performance depends on combining features and capturing nonlinear interactions

(Contributor: *Woo Ka Keung Alex*)

**Correlation Analysis** - basic and additional directionality features (total features: 27):



We refined the feature set using domain knowledge and correlation analysis. Feature selection rationale:

- Short-term trend: **MA50, EMA12, EMA26** – captures near-term momentum.
- Long-term trend: **MA200, EMA50, EMA200** – captures overall market direction.
- Momentum / Oscillator: **RSI14, %K, %D, MACD, Signal\_Line** – for overbought/oversold and crossover signals.
- Volatility: **ATR** – trend strength indicator.
- Volume: **Volume** – to confirm trend strength.

- Directional / Event-based features: **ADX\_StrongTrend**, **ADX\_UpTrend\_DI**, **ADX\_DownTrend\_DI**, **OBV\_New\_High**, **Stoch\_Crossover**, **MA\_Cross**, **MACD\_Hist**, **MACD\_Above\_Zero**.

Removed features:

- Highly redundant or highly correlated features: **MA20**, **BB\_Middle**, **BB\_Upper**, **BB\_Lower**, **RSI9**, **RSI20**. These are either already captured by other features or have strong correlation with included features.

This subset (final selected features: 24) balances predictive power with reducing multicollinearity.

### 3.5 Data Validation *(Contributor: Isak Rabin, Woo Ka Keung Alex)*

After feature engineering, we performed data validation:

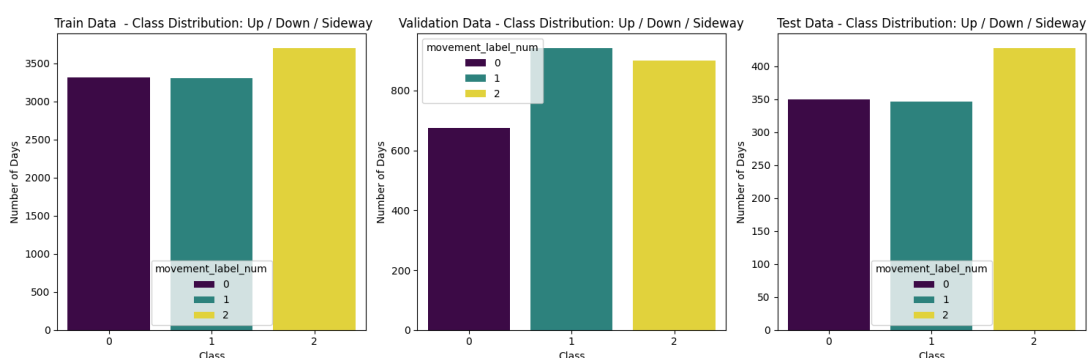
- To fill up records with missing values (eg: 20D Moving Average of Volume, etc) with **forward fill** and **backward fill** (if still any).
- To remove records with missing values (e.g., the last rows that had no movement labels due to the labeling logic)

### 3.5 Splitting the Data *(Contributor: Isak Rabin, Woo Ka Keung Alex)*

To ensure robust model development and unbiased evaluation, we split the dataset **chronologically** into three subsets.

	Period	# Rows	Percentage	
Training Set	1970-01-02 to 2010-12-31	10315	~73%	Used for Training and Tuning
Validation Set	2011-01-01 to 2014-12-31	2517	~18%	
Test Set	2015-01-01 to 2025-06-30	1124	~8%	Used for Final Testing

Above split gave us a balance dataset for each class (Up, Sideway, Down)



## 4. Modelling and Experiments

### 4.1. Model Selection *(Contributor: Isak Rabin, Woo Ka Keung Alex)*

Our problem is a **supervised multiclass classification** problem with three possible outcomes: **Up**, **Down**, or **Sideways**, and we have selected the following **classical supervised machine learning models**:

Model Name	Explanation
Logistic Regression	We have selected Logistic Regression as our baseline model. Its inherent interpretability will be crucial for understanding the foundational relationships between financial indicators and market direction in our dataset.
Decision Tree Classifier	We have chosen it for its <b>rule-based, interpretable structure</b> . It handles both numerical and categorical features, and can model <b>non-linear decision boundaries</b> . It also provides insights into which features are most important in decision making.
Random Forest Classifier	Random Forest will be employed as an <b>ensemble model</b> . Its ability to combine multiple decision trees will enhance predictive accuracy and mitigate overfitting, which is a common challenge when modeling volatile financial time series data like the S&P 500.
xgBoost Classifier (optional)	xgBoost is well known for its best performance in classification problems.

We will choose Decision Tree and Random Forest so that we can compare the impact of ensembles in improving model performance (if any) for our problem (stock movement prediction).

We will evaluate all models based on consistent performance metrics to determine which approach performs best. Hyperparameter tuning will also be applied to optimize each model's behavior based on training performance and generalization.

## 4.2. Output Features *(Contributor: Isak Rabin, Woo Ka Keung Alex)*

The output feature for this project is a categorical label representing the next-day directional movement of the S&P 500 index, which classified into one of three discrete classes:

- **Up** – When the percentage of closing index price (compared to previous day)  $\geq +0.3\%$
- **Down** – When the percentage of closing index price (compared to previous day)  $\leq -0.3\%$
- **Sideways** – When the percentage of closing index price (compared to previous day) is fall between  $-0.3\%$  and  $+0.3\%$

## 4.3. Experiment Tracking *(Contributor: Isak Rabin, Woo Ka Keung Alex)*

We utilized **MLflow** for comprehensive experiment tracking and stored it in google drive. To visualise the result, we integrated with **ngrok** libraries.that created a network tunnel between jupyter notebook to ngrok dashboard.

## 4.4. Experiment Result

*(Contributor: Isak Rabin)*

### Input Features

- We limit the input features to improve the interpretability, avoid multicollinearity, reduce computation cost, and hopefully help the model focus on the most informative signals.
- 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

### Dataset

- Experimenting with different splitting (still maintain chronological time), eg:
  - Multiple train with rolling window, example
    - Train: 1970 - 2000, Validate: 2001-2020, Test: 2021-2025
    - Train: 1970 - 2010, Validate: 2011-2020, Test: 2021-2025
    - and so on
  - Expanding the window with certain focus
    - Train: 2000 - 2020, Validate: 2021-2023, Test: 2023-2025
    - Train: 1970 - 1990, Validate: 1991-1995, Test: 1996-1999
    - and so on

### Training

- 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

	Model	n_features	Train macro-F1 (base)	Train acc (base)	Val macro-F1 (base)	Val acc (base)
0	Logistic	9	0.379573	0.402230	0.378884	0.386969
1	Random Forest	11	1.000000	1.000000	0.371362	0.370679
2	Decision Tree	11	1.000000	1.000000	0.352197	0.354390
3	XGBoost	11	0.949184	0.949103	0.354207	0.353993

Across all four baseline models, we observe **clear overfitting** for tree-based models: training scores are very high (Decision Tree and Random Forest even reached perfect F1 = 1.0), but validation performance drops sharply. Logistic Regression, despite being the simplest, gives the most realistic baseline (avoids memorization). Tree-based models (DT, RF, XGB) capture patterns too well on training but fail to generalize, highlighting the need for **regularization, feature refinement, or better tuning** to improve robustness.

- Decision Tree & Random Forest:
  - Both models achieved perfect training accuracy and macro-F1 (1.0), which clearly indicates overfitting
  - Their validation performance dropped significantly (macro-F1 ~0.35–0.37), showing poor generalization.
- XGBoost:
  - Achieved very high training accuracy (~0.94) but validation macro-F1 remained low (~0.35).
  - This suggests better regularization than Random Forest, but still prone to overfitting.
- Logistic Regression:



- Most balanced compared to other models, validation performance was similar to training (~0.38).
- Training accuracy (~0.37) and macro-F1 (~0.40) were much lower, this indicates underfitting, the model is too simple to capture the complex relationships in the data.
- Performs better on **Sideway** detection but struggles with Down/Up classes.

### Hyperparameter Tuning

- Time series aware
- Use same dataset as initial Training to see improvements
- Use GridSearchCV to tune parameters to mitigate overfitting and optimize performance.
  - 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

Tuning and retraining helped reduce overfitting for tree-based models, but gains were marginal. Overall, despite tuning, the models struggled to exceed the **0.35–0.39 macro-F1** range on unseen data. The weak signal in financial data limited the extent of improvement possible.

	Model	n_features	Train macro-F1 (tuned)	Train acc (tuned)	Val macro-F1 (tuned)	Val acc (tuned)
0	20250828-163440-tuning-tree	11	0.402027	0.426175	0.374680	0.427096
1	20250828-163440-tuning-rf	11	0.939419	0.939699	0.398917	0.398888
2	20250828-163440-tuning-linear	9	0.379573	0.402230	0.378884	0.386969
3	20250828-163440-tuning-xgb	11	0.947079	0.947164	0.350540	0.350020

- Decision Tree (tuned)
  - Validation macro-F1 improved slightly from 0.35 to 0.37, showing better generalization.
  - Stronger at detecting Up (high recall), but struggles on Down.
  - Still prone to underfitting/instability despite tuning.
- Random Forest (tuned):
  - Validation macro-F1 improved to ~0.39, suggesting slightly reduced overfitting compared to baseline model.
- XGBoost (tuned):
  - Validation macro-F1 slightly decreased (0.3542 to 0.3505), showing limited gain from tuning.
- Logistic Regression:
  - Results remained consistent with base training, with macro-F1 around **0.37**, as expected from its simpler design.
  - Performing relatively better on *Sideway* detection but weaker on *Up* and *Down*.
  - Shows no overfitting, but overall predictive power is limited.

### Test Results

All models fix around **0.30–0.38 macro-F1** on test data, which is significantly below the target success metrics set in the project proposal. Logistic Regression remains the simplest and most stable baseline, while **XGBoost achieves the best balance across classes**, though still limited by data complexity and imbalance. Further improvements will require **feature engineering, class balancing, and stronger regularization**.

	Model	n_features	Test macro-F1 (test)	Test acc (test)
0	XGBoost	11	0.383773	0.395018
1	Decision Tree	11	0.363659	0.431495
2	Random Forest	11	0.354931	0.405694
3	Logistic	9	0.300999	0.401246

- **XGBoost:**
  - Best macro-F1 (0.38), test accuracy ~0.39.
  - Balanced across classes, giving moderate recall for *Down* (0.31) and better trade-off overall.
- **Decision Tree:**
  - Macro-F1 ~0.36, but highest test accuracy (~0.43).
  - Best at detecting *Sideway* and *Up*, but *Down* remains very poor.
  - Slightly more balanced than Logistic.
- **Random Forest:**
  - Macro-F1 ~0.35, accuracy ~0.40.
  - Moderate performance. Good recall on *Up* (0.73) but weak on *Down* and *Sideway*.
  - Overfitting is still evident despite tuning.
- **Logistic Regression:**
  - Lowest macro-F1 (0.30), test accuracy ~0.40.
  - Performs well on *Up* (recall 0.83), but fails badly on *Down* (recall 0.02).
  - Stable but weak overall (struggled to capture non-linear signals).

### Overall Experiment Summary

- **Overfitting was a major issue** for tree-based models (Decision Tree, Random Forest, XGBoost), seen from perfect/near-perfect training scores but weaker validation/test results.
- **Logistic Regression**, while simple, was the most stable model, though its predictive ceiling is low. It struggles heavily with the **Down** class.
- **XGBoost achieved the best overall balance on the test set**, but still far below project success metrics (target F1 > 0.65).
- Across all models, **Down** movements were consistently the hardest to predict, while **Sideway** and **Up** were detected more reliably.
- **Macro-F1 scores plateaued at 0.30 - 0.38 on test**, showing models capture some patterns but fail to generalize strongly to unseen data.

(Contributor: *Woo Ka Keung Alex*)

Random Forest pre-tuned hyperparameters (based on general best practices for financial datasets with moderate sample size):

- n\_estimators=300 (number of trees)
- max\_depth=20 (control overfitting)
- min\_samples\_split=5
- min\_samples\_leaf=3
- random\_state=42 (reproducibility)
- class\_weight="balanced" (address any class imbalance)

**Training Accuracy:** 0.9483 (~94.8%)

- Accuracy: 94.8% (very strong performance on training data).
- The model correctly predicts ~94.8% of the labels on the training set.



- This is reasonable for financial data, where perfect accuracy is rare due to market noise.
- It indicates the model has learned meaningful patterns without severe overfitting (i.e., accuracy is not 100%).

**Validation Accuracy:** 0.5936 (~59%)

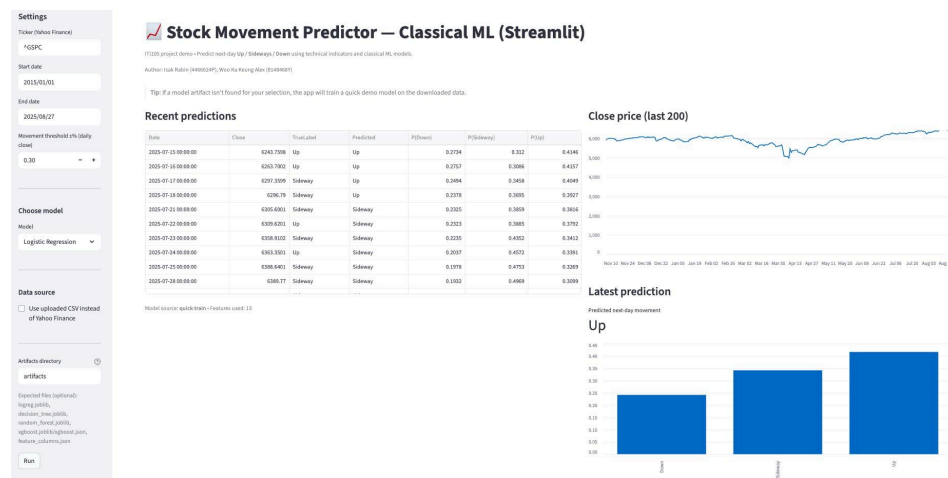
- Accuracy drops from ~94.8% (training) to ~59% (validation).
- This indicates the model is overfitting the training data: it performs very well on training but struggles to generalize to unseen data.
- Overfitting is common in Random Forests when trees are deep or features are highly correlated.

**Test Accuracy:** 0.5654 (~56.5%)

- Overall Accuracy: 0.5654 (56.5%)
- The Random Forest model correctly predicted the label for just over half of the test samples.
- This is notably lower than the training accuracy (94.8%), confirming overfitting—the model learned training patterns too well but failed to generalize.

## 5. Deployment *(Contributor: Isak Rabin, Woo Ka Keung Alex)*

We developed a **low-code interactive web application** with **Streamlit** to simulate **market prediction tool applications**. The application allows users to input recent market data (or upload it), and receive a predicted **movement direction** (Up, Down, or Sideways) for the next day from the developed model



The application is deployed into the **HuggingFace** platform, and accessible on the following URL: <https://huggingface.co/spaces/sgirabin/iti105-machine-learning-project>.

## 6. Summary *(Contributor: Isak Rabin, Woo Ka Keung Alex)*

- **Overall** we have achieved our project goal - the set of models that developed outperform random guessing (33%) although our target (65%) has not been reached. This set of models will be a valuable resource to help market practitioners make informed decisions systematically.
- **Financial data is hard to predict**, no single model found strong signals due to the semi-efficient, noisy nature of daily S&P 500 movements.
- **Overfitting was a major issue for Decision Tree and Random Forest**. Tuning helped reduce it, but performance gains were small.

- *(Isak Rabin)* **XGBoost performed best overall in terms of F1-macro**, meaning it balanced predictions across Up, Down, and Sideways better than the others.
- *(Woo Ka Keung Alex)* **Random Forest** can achieve reasonably high accuracy but still exhibits overfitting symptoms despite reduction in highly correlated features and avoiding deep trees. Class prediction accuracy is also consistent. Hyper-parameter tuning improved performance but to a limited extent. On the other hand, the addition of directionality features improved model performance compared to just using the basic technical indicators.
- **Logistic Regression underfit badly and performed the worst**, showing that simple linear models cannot capture the weak, non-linear signals in the data.
- **The results highlight that ensemble and boosting methods are more robust** for this task, but still limited in accuracy due to inherent data difficulty.

## 7. Future Works *(Contributor: Isak Rabin, Woo Ka Keung Alex)*

- **Imbalance Handling:** Although the dataset does not show any imbalance, the results show similarity with imbalance dataset. So, we may want to consider applying SMOTE or class weighting to see if any improvement can be made.
- **Feature Engineering:**
  - Add more financial indicators (e.g., volatility clustering, sector indices, volume shocks).
  - Explore lag features and rolling windows to better capture temporal dependencies.
  - Incorporate macroeconomic indicators (interest rates, inflation, sentiment indices) to enrich predictive context. These provide additional dimensions beyond pure technical analysis, Cyclical patterns (calendar effects), market timing (month-end, quarter-end) effects, and sentiment proxies (greed vs fear) can significantly improve prediction accuracy.
  - Use higher-frequency intraday data to capture finer patterns if feasible.
- **Regularization and Tuning**
  - Apply stronger regularization to tree-based models (max depth, min samples per split/leaf, learning rate schedules).
  - Use early stopping in XGBoost to prevent overfitting.
- **Model Expansion:**
  - Try ensemble methods (stacking or soft voting) to combine strengths of Logistic and XGBoost.
  - Consider temporal CV (rolling window validation) to reflect real-world market prediction more closely.

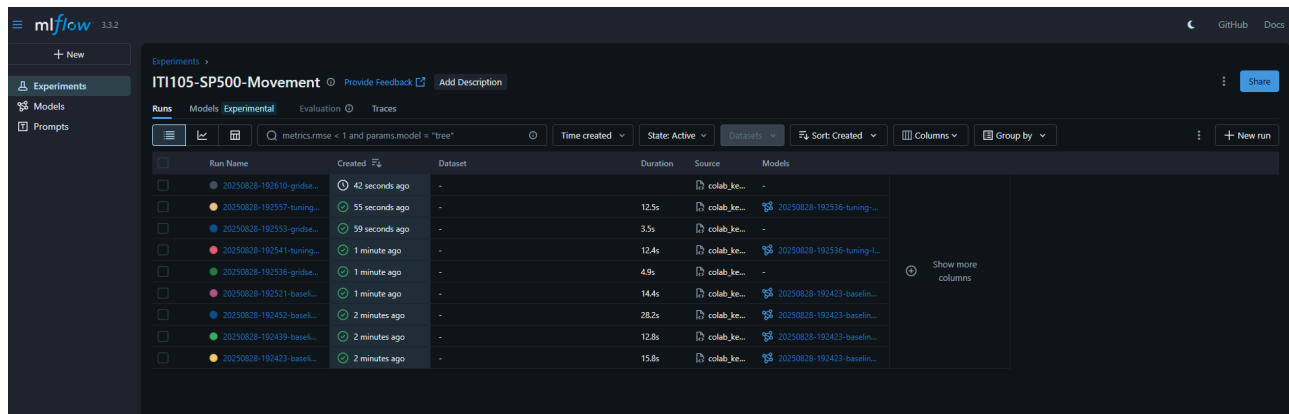
## 8. References

- **Bakary Bah.** (2023). *Predicting the Movement of the S&P 500 Index using Machine Learning*. Master's Thesis, Lund University.  
<https://lup.lub.lu.se/luur/download?func=downloadFile&recordId=9137039&fileId=9137040>
- **Giovanni Campisi, Silvia Muzzioli, Bernard De Baets** (2024). *A comparison of machine learning methods for predicting the direction of the US stock market on the basis of volatility indices*, International Journal of Forecasting, Volume 40, Issue 3,  
<https://www.sciencedirect.com/science/article/pii/S0169207023000729>
- **Luckyson Khaidem, Snehanshu Saha, Sudeepa Roy Dey.** (2016). *Predicting the direction of stock market prices using random forest*.  
<https://arxiv.org/pdf/1605.00003>
- Recent 3 Months (Mar - June 2025) Data for S&P 500 Index Market,  
<https://sg.finance.yahoo.com/quote/%5EGSPC/history/?period1=1743379200&period2=1751587200>
- Stock Market Prediction: [https://en.wikipedia.org/wiki/Stock\\_market\\_prediction](https://en.wikipedia.org/wiki/Stock_market_prediction)

## Subject: ITI105 - Machine Learning Project

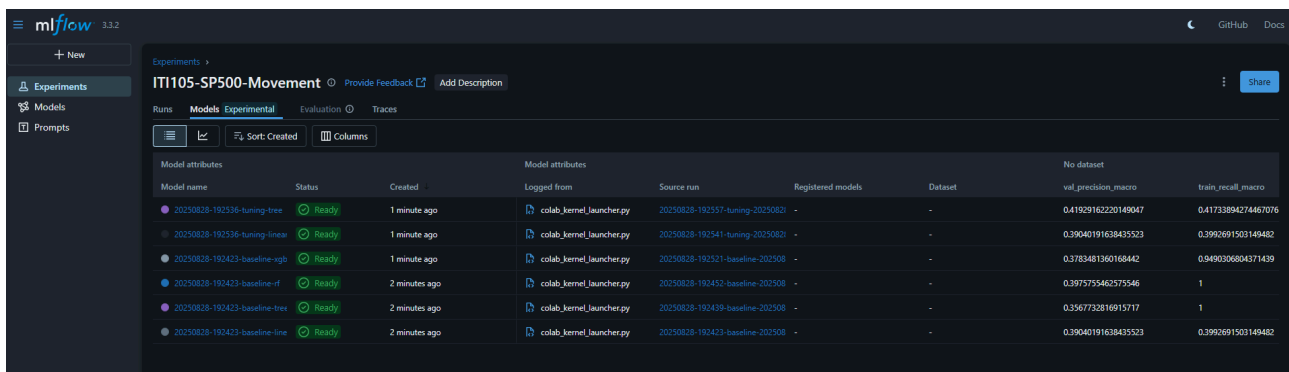
- Understanding Indicators in Technical Analysis, [https://www.fidelity.com/bin-public/060\\_www\\_fidelity\\_com/documents/learning-center/Understanding-Indicators-TA.pdf](https://www.fidelity.com/bin-public/060_www_fidelity_com/documents/learning-center/Understanding-Indicators-TA.pdf)
- Yahoo Finance Python Libraries, <https://github.com/ranaroussi/yfinance>

## Appendix - MLFlow Screenshots (Contributor: Isak Rabin)



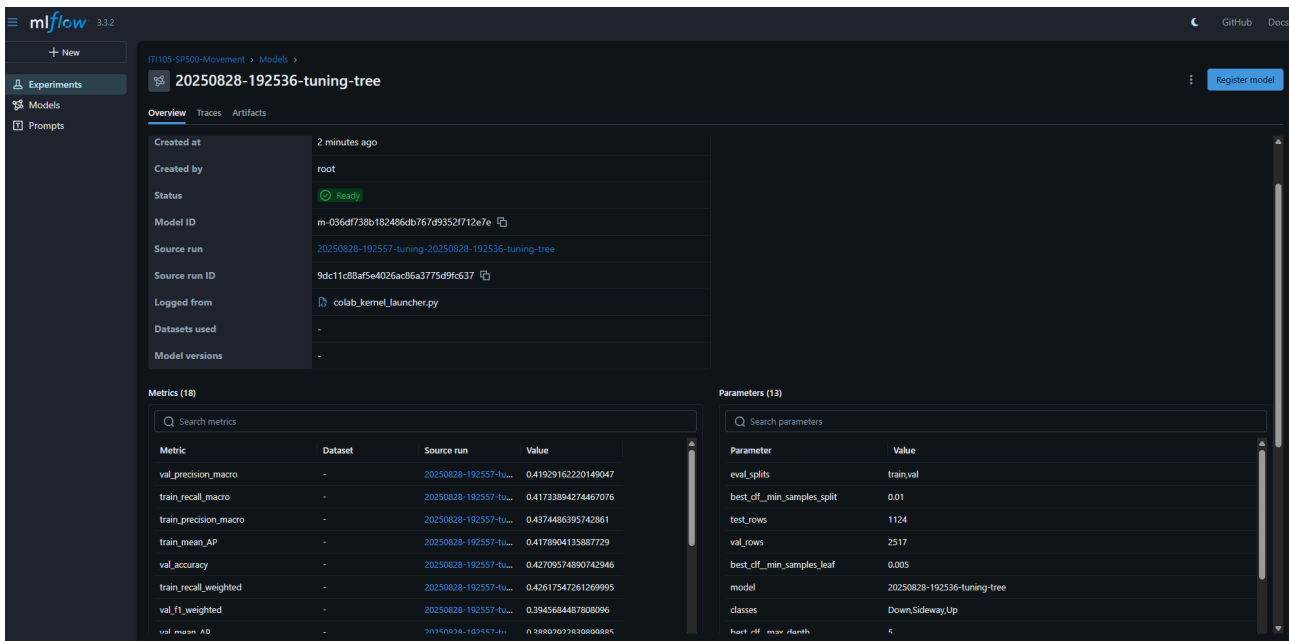
The screenshot shows the MLFlow Experiments interface for the experiment 'ITI105-SP500-Movement'. The 'Runs' tab is selected, displaying a table of runs. The table columns are Run Name, Created, Dataset, Duration, Source, and Models. The runs are sorted by 'Time created' and are in 'Active' state. A search bar at the top shows the query 'metrics.rmse < 1 and params.model = "tree"'. A '+ New run' button is in the top right.

Run Name	Created	Dataset	Duration	Source	Models
20250828-192610-gridse...	42 seconds ago	-	-	colab_ke...	-
20250828-192557-tuning...	55 seconds ago	-	12.5s	colab_ke...	20250828-192536-tuning...
20250828-192553-gridse...	59 seconds ago	-	3.5s	colab_ke...	-
20250828-192541-tuning...	1 minute ago	-	12.4s	colab_ke...	20250828-192536-tuning-L...
20250828-192536-gridse...	1 minute ago	-	4.9s	colab_ke...	-
20250828-192521-basel...	1 minute ago	-	14.4s	colab_ke...	20250828-192423-basel...
20250828-192452-basel...	2 minutes ago	-	28.2s	colab_ke...	20250828-192423-basel...
20250828-192439-basel...	2 minutes ago	-	12.8s	colab_ke...	20250828-192423-basel...
20250828-192423-basel...	2 minutes ago	-	15.8s	colab_ke...	20250828-192423-basel...



The screenshot shows the MLFlow Models interface for the experiment 'ITI105-SP500-Movement'. The 'Models' tab is selected, displaying a table of registered models. The table columns are Model name, Status, Created, Logged from, Source run, Registered models, Dataset, No dataset, val\_precision\_macro, and train\_recall\_macro. The models are sorted by 'Sort: Created'.

Model name	Status	Created	Logged from	Source run	Registered models	Dataset	No dataset	val_precision_macro	train_recall_macro
20250828-192536-tuning-tree	Ready	1 minute ago	colab_kernel_launcher.py	20250828-192557-tuning-20250828	-	-	-	0.41929162220149047	0.41733894274467076
20250828-192536-tuning-linear	Ready	1 minute ago	colab_kernel_launcher.py	20250828-192541-tuning-20250828	-	-	-	0.39040191638435523	0.3992691503149482
20250828-192423-baseline-rgb	Ready	1 minute ago	colab_kernel_launcher.py	20250828-192521-baseline-20250828	-	-	-	0.3783481360168442	0.3490306804371439
20250828-192423-baseline-rl	Ready	2 minutes ago	colab_kernel_launcher.py	20250828-192452-baseline-20250828	-	-	-	0.397575462575546	1
20250828-192423-baseline-tree	Ready	2 minutes ago	colab_kernel_launcher.py	20250828-192439-baseline-20250828	-	-	-	0.3567732816915717	1
20250828-192423-baseline-line	Ready	2 minutes ago	colab_kernel_launcher.py	20250828-192423-baseline-20250828	-	-	-	0.39040191638435523	0.3992691503149482

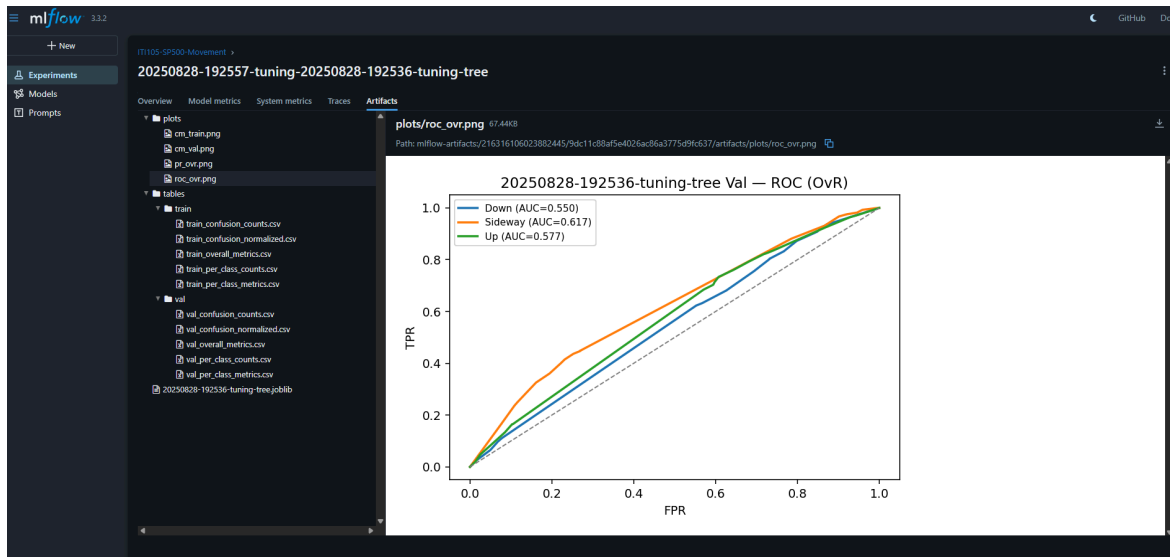
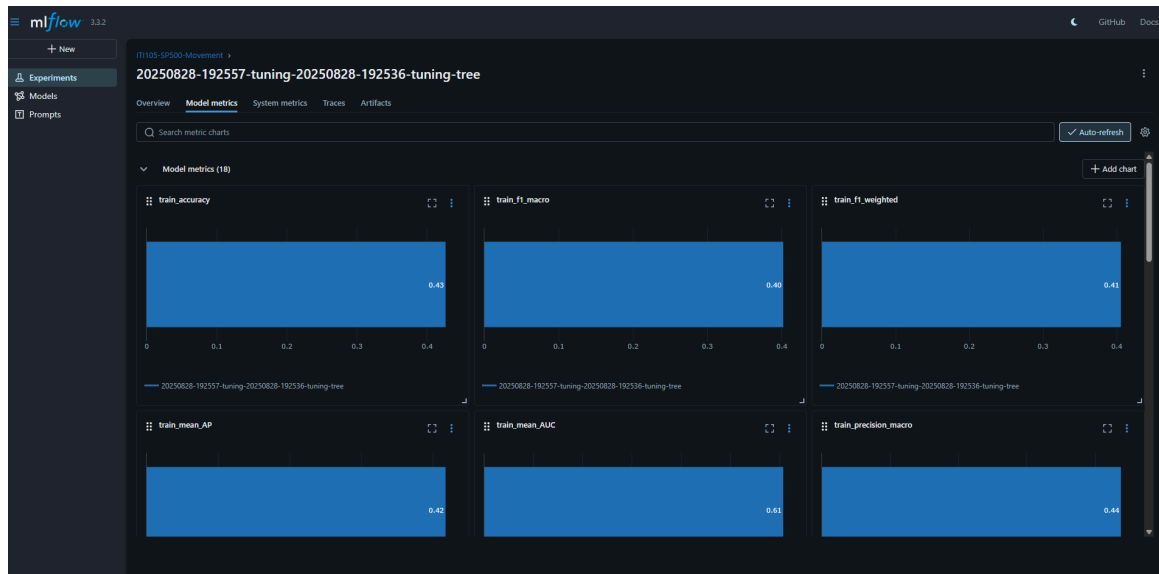


The screenshot shows the MLFlow Model interface for the model '20250828-192536-tuning-tree'. The 'Overview' tab is selected, displaying a table of model details. The table columns are Created at, Created by, Status, Model ID, Source run, Source run ID, Logged from, Datasets used, and Model versions. The model is in 'Ready' status. Below the table, there are sections for 'Metrics (18)' and 'Parameters (13)'.

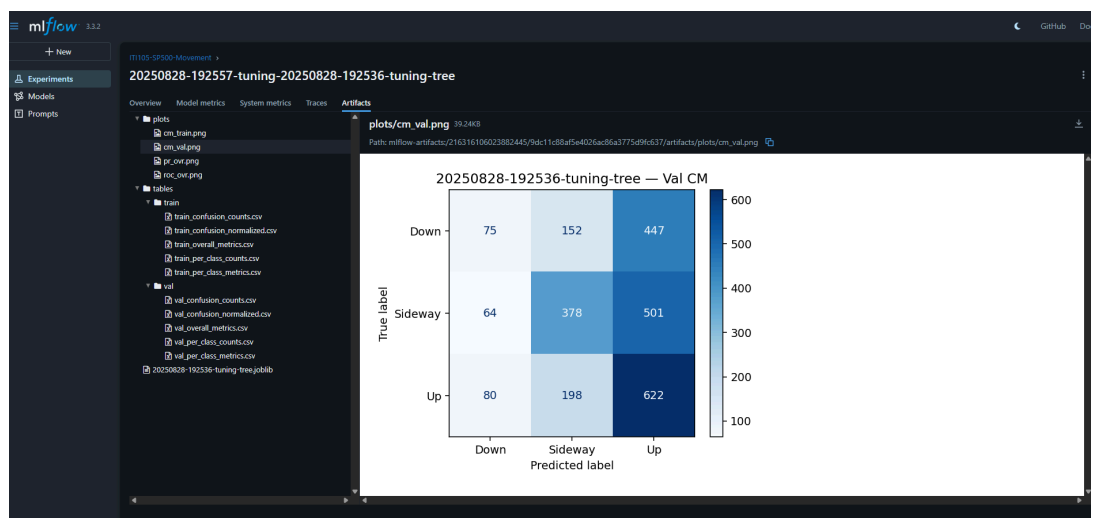
Metric	Dataset	Source run	Value
val_precision_macro	-	20250828-192557-tu...	0.41929162220149047
train_recall_macro	-	20250828-192557-tu...	0.41733894274467076
train_precision_macro	-	20250828-192557-tu...	0.4374486395742861
train_mean_AP	-	20250828-192557-tu...	0.4178904135887729
val_accuracy	-	20250828-192557-tu...	0.42709574890742946
train_recall_weighted	-	20250828-192557-tu...	0.42617547261269995
val_f1_weighted	-	20250828-192557-tu...	0.394568448780896
val_mean_AU	-	20250828-192557-tu...	0.3880707383000086

Parameter	Value
eval_splits	train, val
best_of_min_samples_split	0.01
test_rows	1124
val_rows	2517
best_of_min_samples_leaf	0.005
model	20250828-192536-tuning-tree
classes	Down, Sideway, Up
best_of_min_samples	5

## Subject: IT1105 - Machine Learning Project



Subject: ITI105 - Machine Learning Project



mlflow 3.3.2

ITI105-SF500-Movement

20250828-192557-tuning-20250828-192536-tuning-tree

Overview Model metrics System metrics Traces Artifacts

plots

- cm\_train.png
- cm\_val.png
- pr\_ov.png
- roc\_ov.png

tables

- train
  - train\_confusion\_counts.csv
  - train\_confusion\_normalized.csv
  - train\_overall\_metrics.csv
  - train\_per\_class\_counts.csv
  - train\_per\_class\_metrics.csv
- val
  - val\_confusion\_counts.csv
  - val\_confusion\_normalized.csv
  - val\_overall\_metrics.csv
  - val\_per\_class\_counts.csv
  - val\_per\_class\_metrics.csv

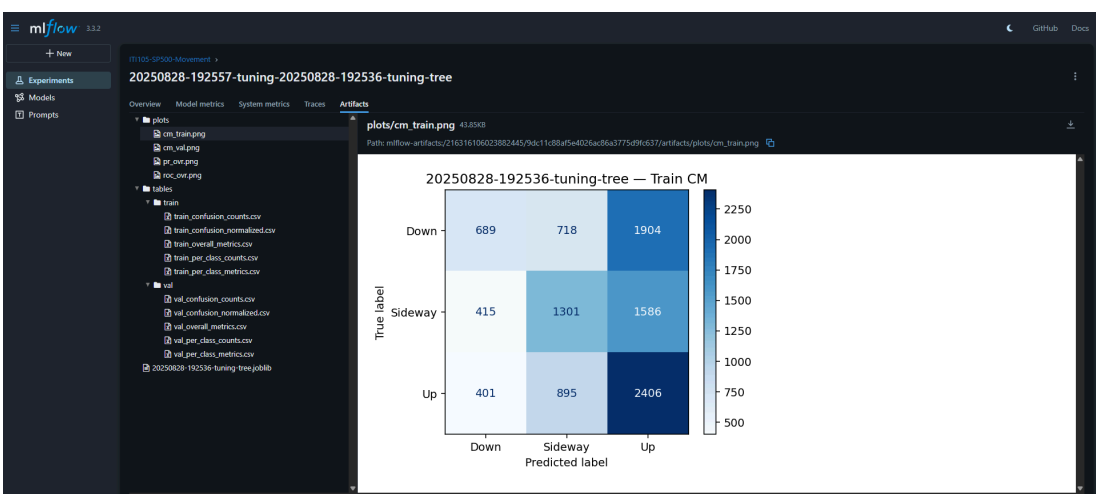
20250828-192536-tuning-tree.joblib

tables/train/train\_confusion\_counts.csv 73B

Path: mlflow-artifacts/216316106023882445/9dc11c88af5e4026ac86a3775d96c37/artifacts/tables/train/train\_confusion\_counts.csv

Previewing the first 3 rows

	Down	Sideway	Up
Down	689	718	1904
Sideway	415	1301	1586
Up	401	895	2406



Subject: ITI105 - Machine Learning Project

