EUR/USD Forex Pair Next Hourly Bar Classification

01418362 Introduction to Machine Learning

6610401985 ไชยวัตน์ หนูวัฒนา

Data Loading

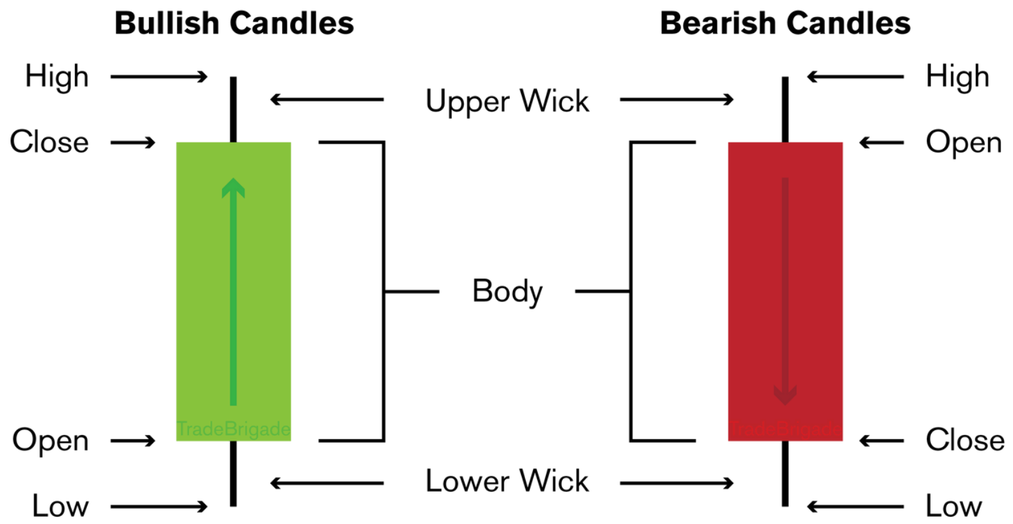
EUR/USD Hourly Price Historical Data

Raw Data Shape

100,000 x 5

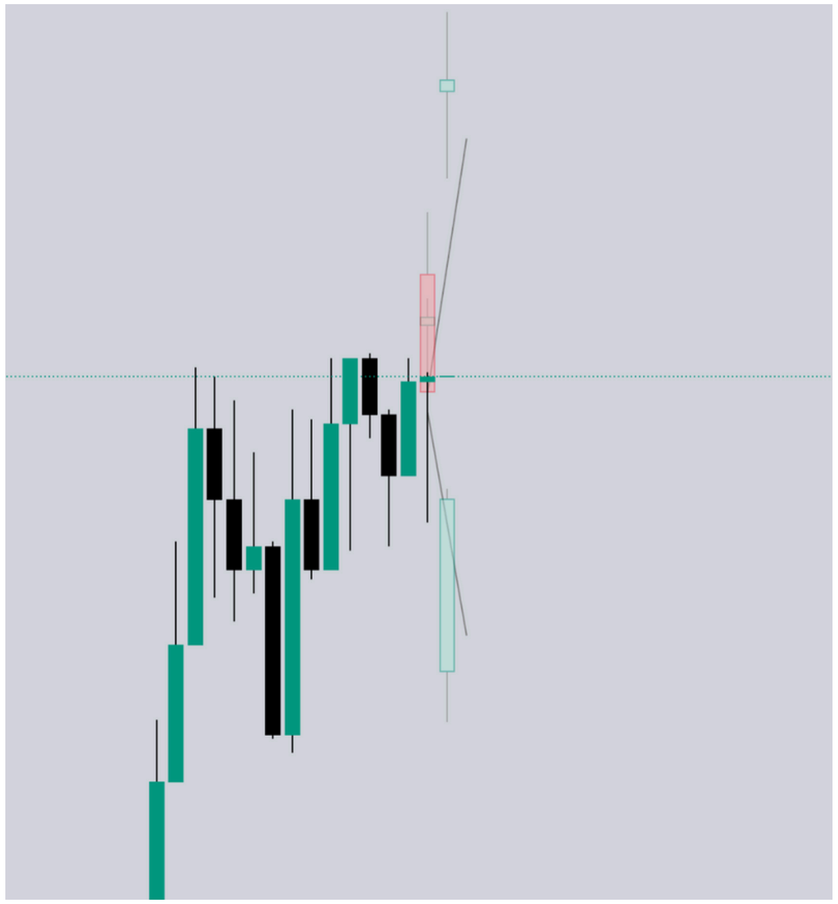
Features

- Open
- High
- Low
- Close
- Volume



From : https://forexsb.com/historical-forex-data

Problem Exploration

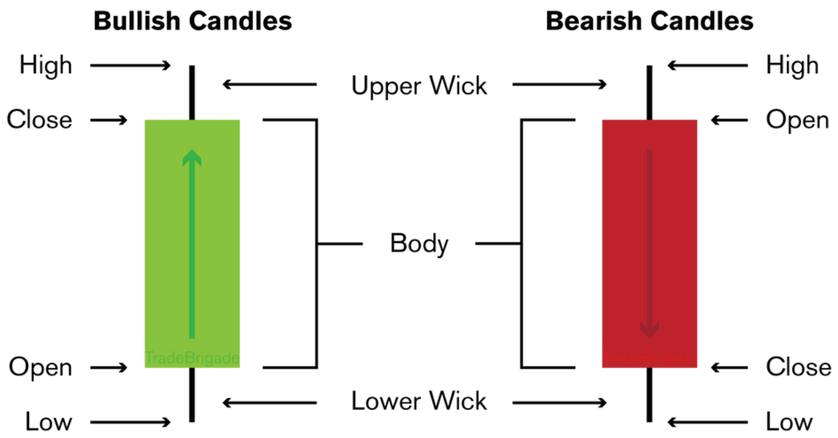


The problem is to predicted the next Bar that either

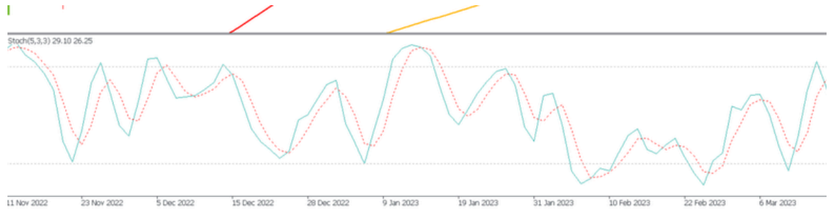
- Bullish (Up Candle)
- Bearish (Down Candle)

Base on information from the past

Feature Engineering



Financial Metrics
&
Technical Indicators



Feature Engineering (cont.)

Some of the Financial Metrics and Technical Indicators

- Price Change Percentage
- Price Range
- Volatility
- Momentum
- Simple Moving Average
- Exponential Moving Average
- Stochastic
- ETC.

Engineered Data Shape

99945 x 35

Modelling

1. Decision Tree
   a. Why ? → features is around 40 features tree-based algorithm have the impurity function to select the best features
2. Random Forest
   a. Why ? → Since the DT often overfit due to high variance the Bagged DT (Random Forest) can fix this problem
3. AdaBoost (Decision Tree Based)
   a. Why ? → In the experimental, I try to use stumps tree instead of full tree and stumps tree is weak classifier and high bias boosting technique can fix this problem
4. Support Vector Machine
   a. Why ? → Since all the algorithm above is tree-based algorithm, I can only select the best features by extract the feature importances and apply SVM or Kernelized SVM to find the linear decision boundary to classify the data points

Model Selection

How I choose the best model ?

```
df.next_bars_bullish.value_counts() / len(df)
```

✓ 0.0s

```
next_bars_bullish
Bearish    0.500355
Bullish    0.499645
Name: count, dtype: float64
```

Performing better than BASELINE

- Since the ratio of Bearish and Bullish is equal
- Random guess would achieve 50% accuracy
- the model must outperform baseline

Variance and Bias Tradeoff

- Not too Complex
- Not too Simple
- Balance Between Complex and Simple
- or Best of the setting possible

Model Selection Process

01 Split Train Test Split

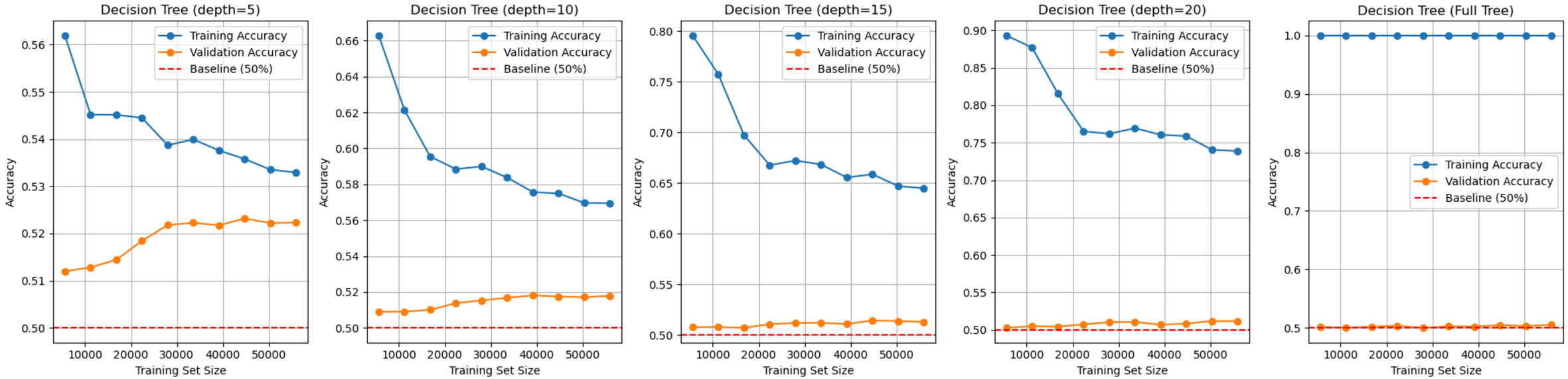
02 K-Fold CV

03 Observe Learning Curve

04 Fit and Evaluate

Decision Tree

Learning Curve below is show the **Decision Tree** with different Depth and compare Training Accuracy and Validation Accuracy



Decision Tree (cont.)

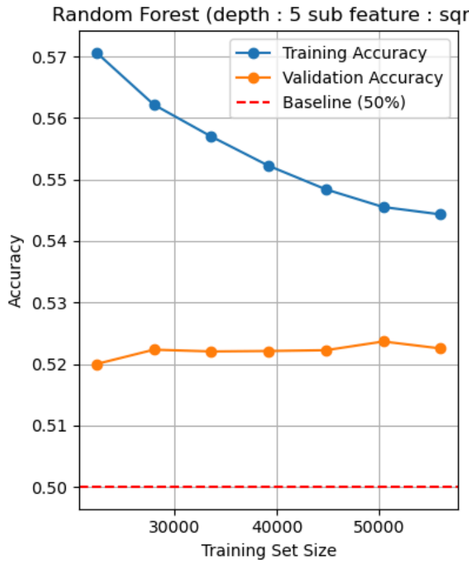
The Best DT out there is Decision Tree with Depth = 10 , I pick it and fit then evaluate

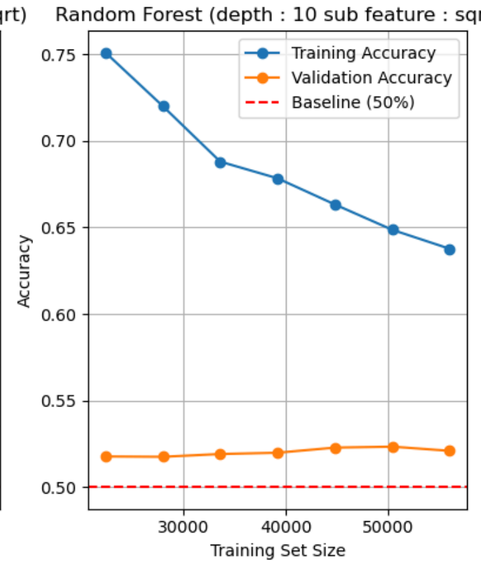
Metric	Precision	Recall	F1-Score	Support
Bearish	0.52	0.55	0.53	15049
Bullish	0.52	0.49	0.50	14935
Accuracy			0.52	29984
Macro Avg	0.52	0.52	0.52	29984
Weighted Avg	0.52	0.52	0.52	29984

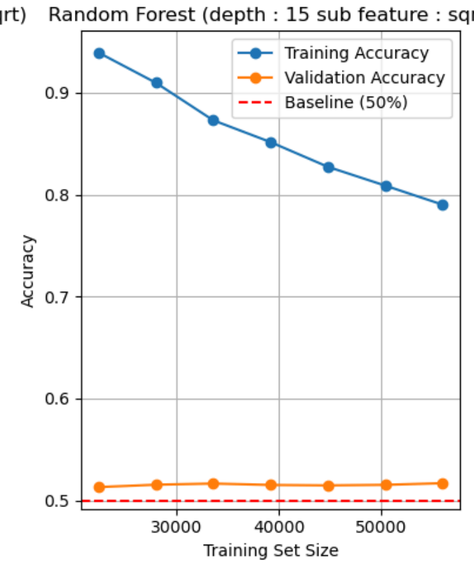
But we can see the learning curve and see that models still **overfitted (High Variance)**

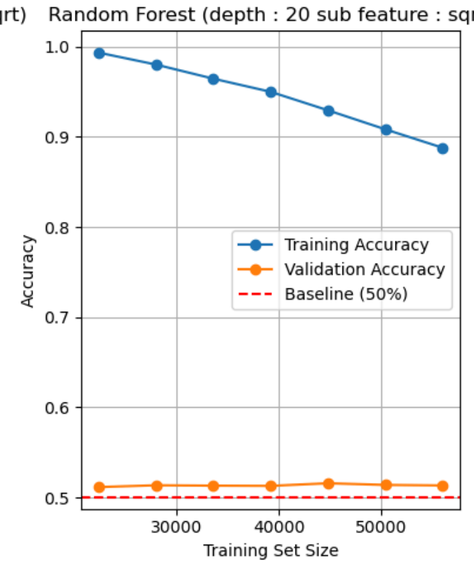
Random Forest

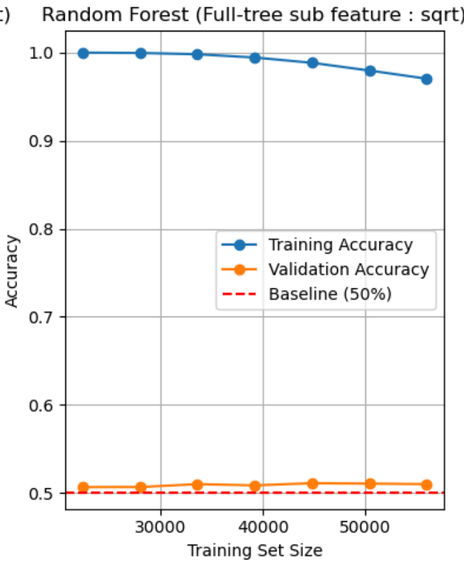
Learning Curve below is show the **Random Forest** with different Depth and compare Training Accuracy and Validation Accuracy











Random Forest (cont.)

The Best RF out there is Random Forest with Depth = 15 , I pick it and fit then evaluate. Why Bagging but still not full tree ? → when its full tree it better than single DT but still overfitted

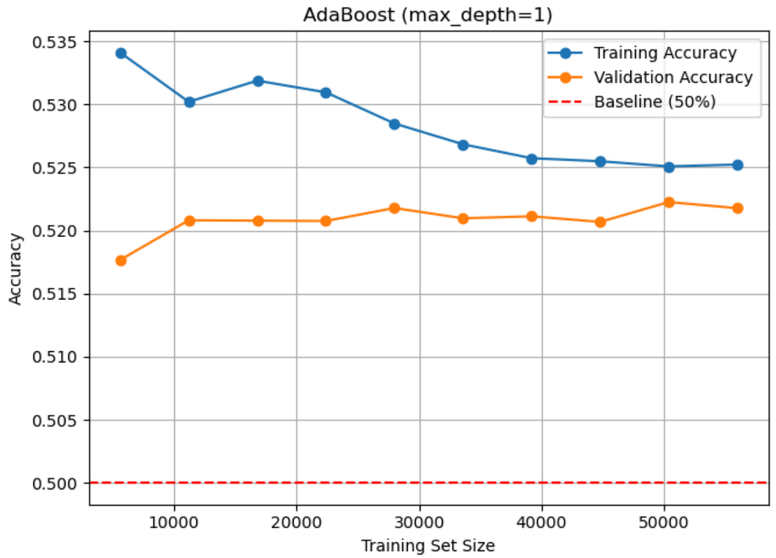
Metric	Precision	Recall	F1-Score	Support
Bearish	0.52	0.55	0.53	15049
Bullish	0.52	**0.52**	**0.52**	14935
Accuracy			0.52	29984
Macro Avg	0.52	0.52	0.52	29984
Weighted Avg	0.52	0.52	0.52	29984

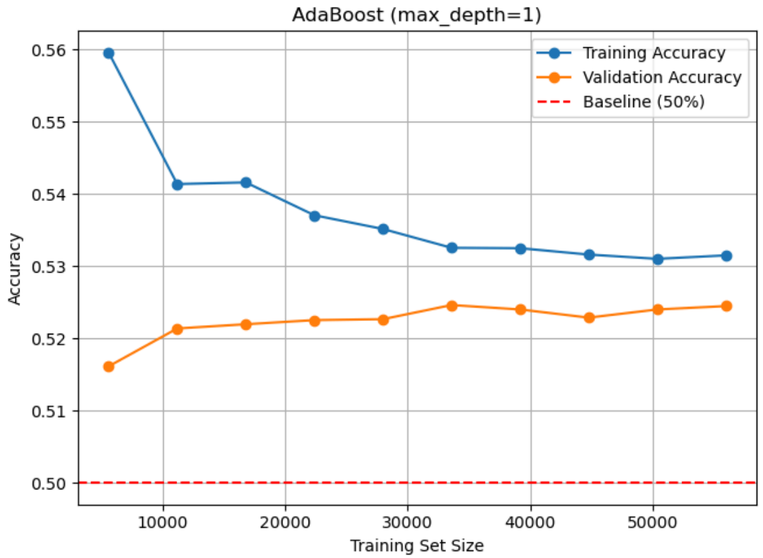
The RF give a very minimal improvement from single DT.
And we can see the learning curve and see that models still **overfitted** (again😩)

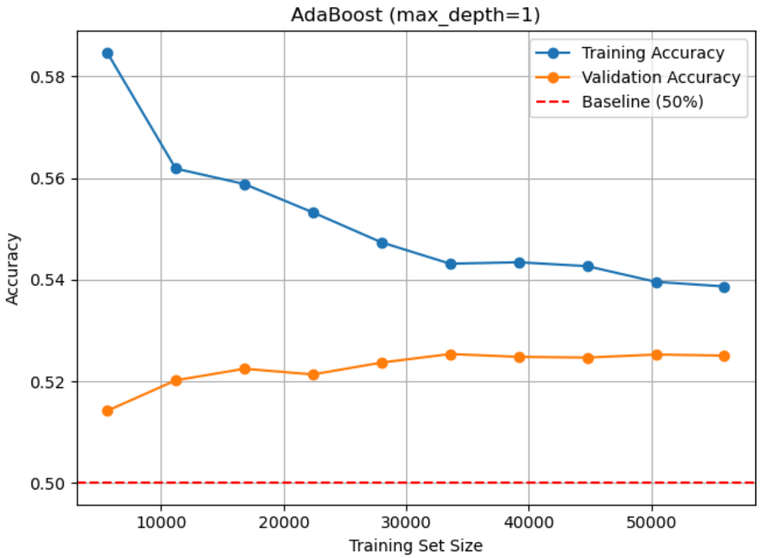
AdaBoost

Next, I try using **stumps tree** and **boosting techniques** to model the data. I use **AdaBoost with Decision Tree Based**

Learning Curve below is show the AdaBoost with different Depth of BaseLearner and compare Training Accuracy and Validation Accuracy







AdaBoost (cont.)

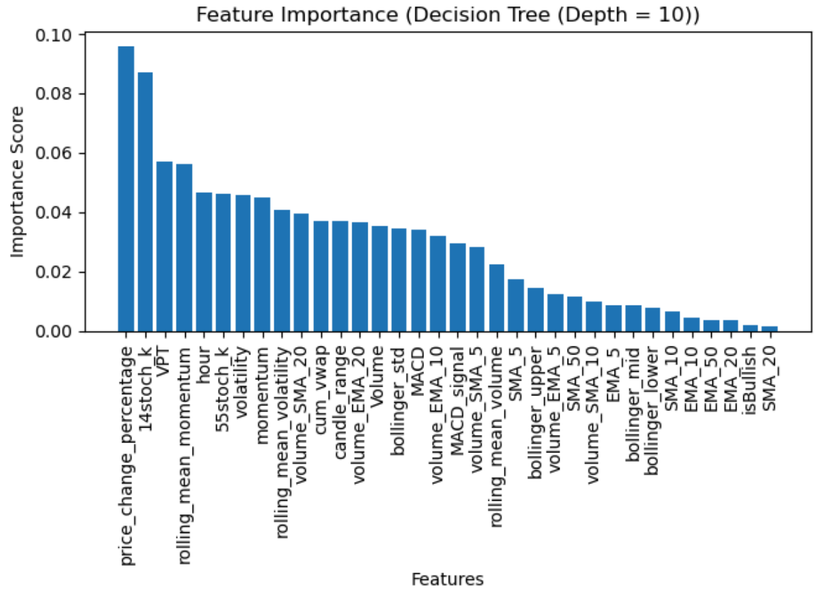
We can see that DT base with any depth setting offer the same performance, So I choose the AdaBoost with DT depth of 5

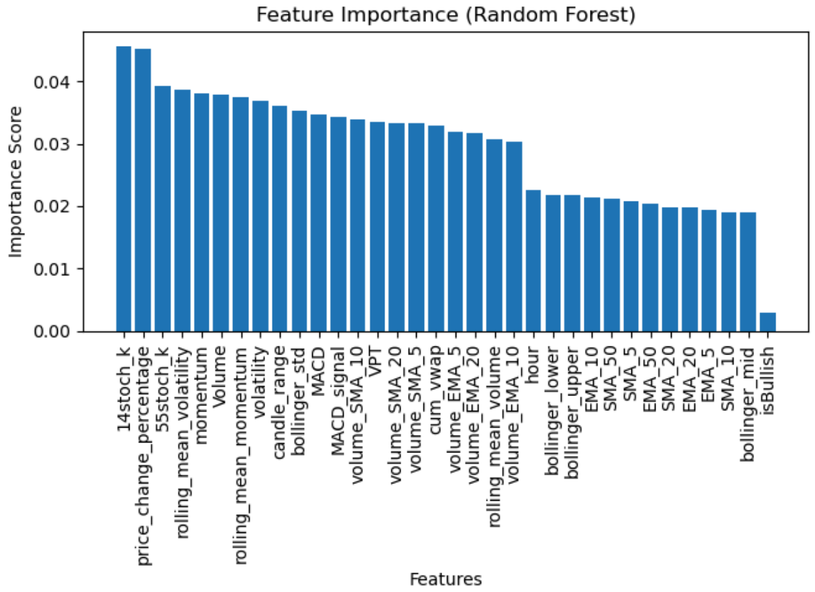
Metric	Precision	Recall	F1-Score	Support
Bearish	0.52	0.55	0.53	15049
Bullish	0.52	**0.54**	**0.53**	14935
Accuracy			**0.53**	29984
Macro Avg	**0.53**	**0.53**	**0.53**	29984
Weighted Avg	**0.53**	**0.53**	**0.53**	29984

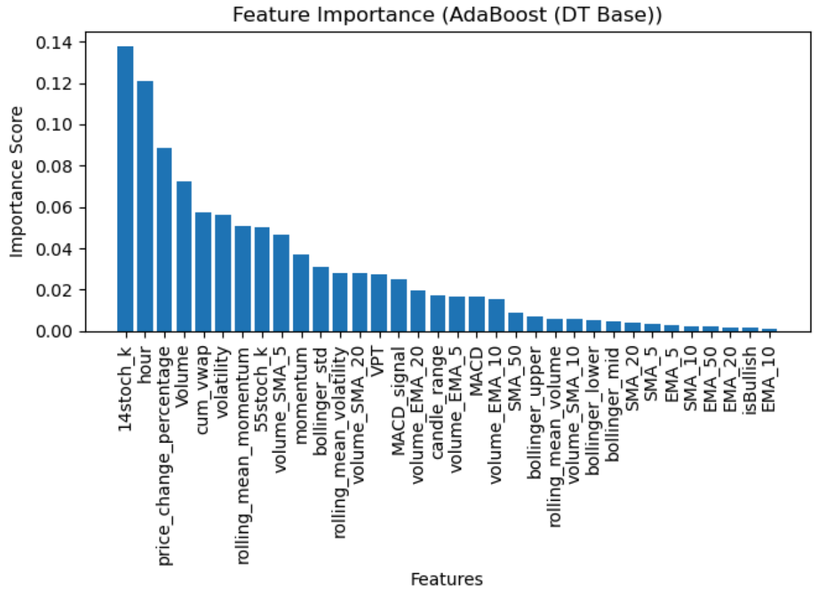
We can see that AdaBoost give a **small improvement**.
And from the learning Curve we can see that training and validation **more converged** but still very **poor performance** 🤯

Support Vector Machine

Since I'm using the tree-based models so far, I can extract only the best features by look at **feature importance** from all of 3 models and then use it in **Support Vector Machine**







Support Vector Machine (cont.)

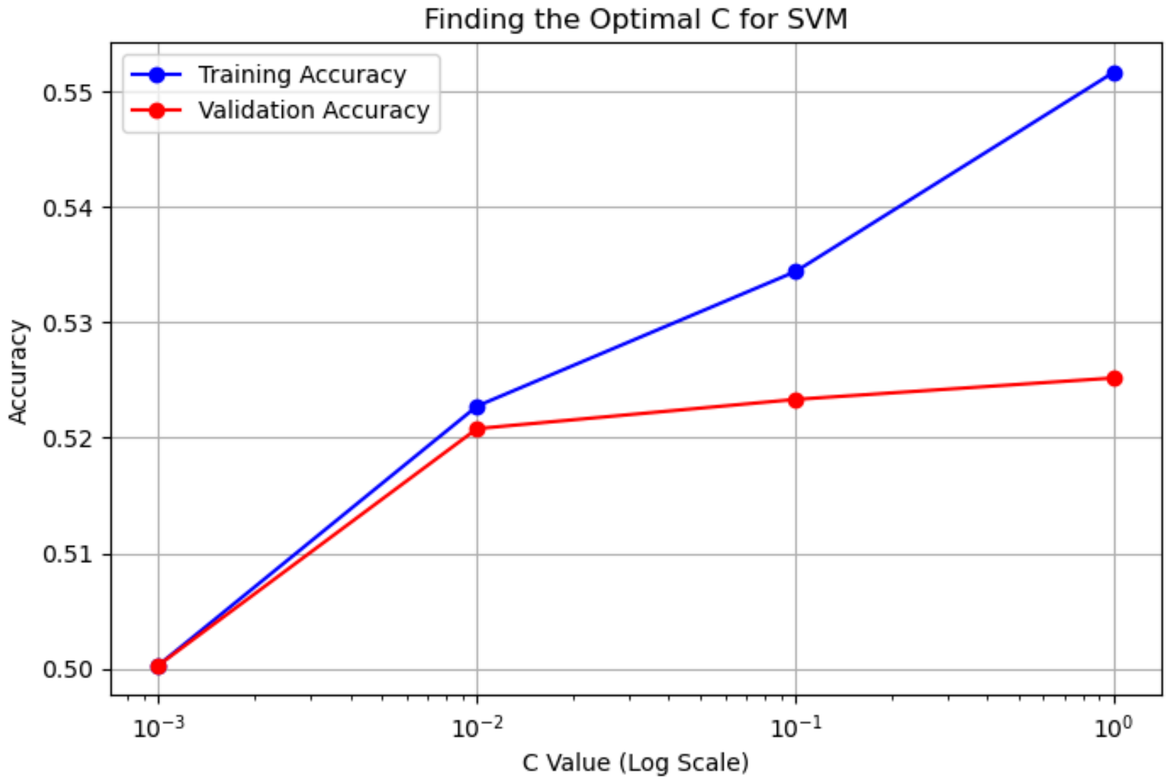
Below is the classification reports of **SVM linear kernel** of C = 1, which is the best C parameter I try so far

Metric	Precision	Recall	F1-Score	Support
Bearish	0.52	0.52	0.52	15049
Bullish	0.52	0.52	0.52	14935
Accuracy			0.52	29984
Macro Avg	0.52	0.52	0.52	29984
Weighted Avg	0.52	0.52	0.52	29984

compare to other C setting of Linear SVM, when C is very low the algorithm sacrifice all data points and only predicted the Bullish class since the Bullish class is slight ly higher than Bearish class (50.1% vs 49.9%)

Support Vector Machine (cont.)

Since Linear Kernel SVM not improve any performance, I try to use **RBF Kernel** to tranforms the features space to higher dimensions. Below is the graph of C value between Training and Validation accuracy



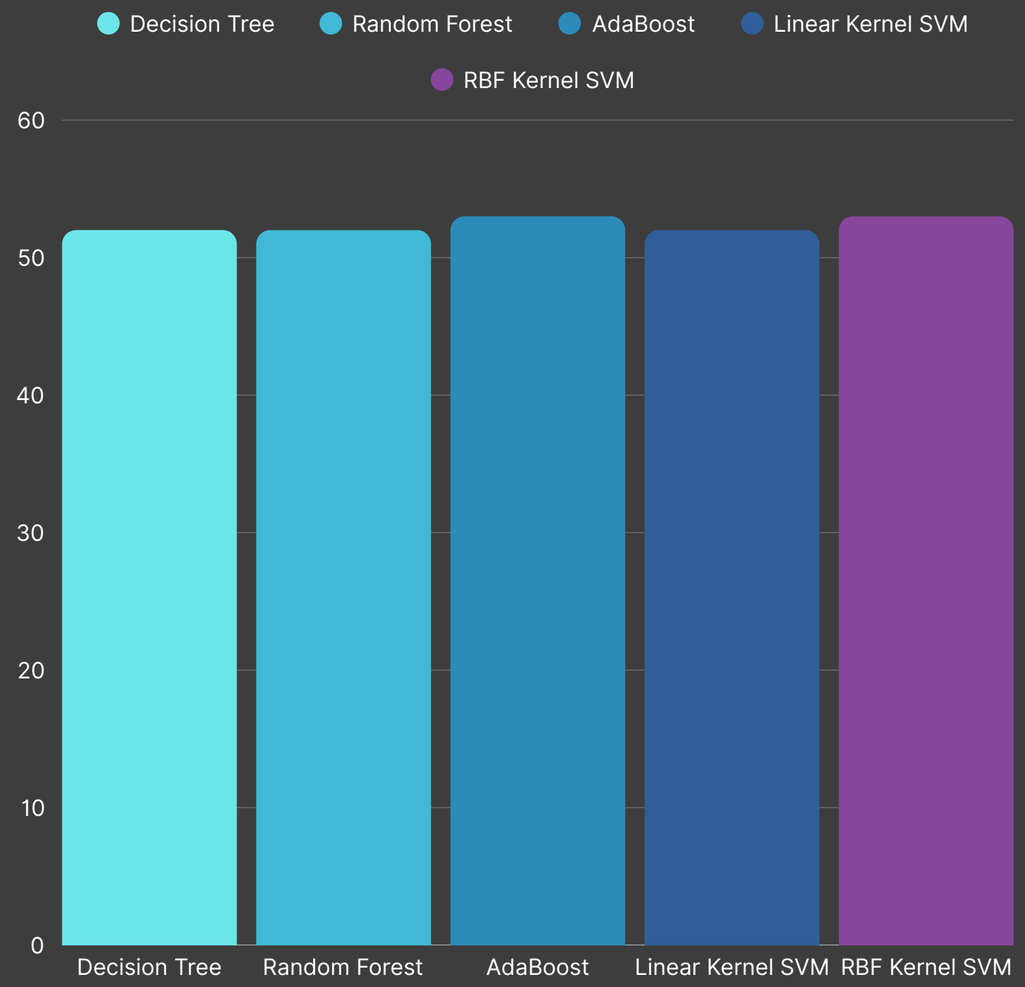
We can see that the optimal value of C in **SVM RBF Kernel** is 0.01 and 1

Support Vector Machine (cont.)

Below is the classification reports of **SVM RBF kernel** of C = 1, which is the best C parameter I try so far

Metric	Precision	Recall	F1-Score	Support
Bearish	**0.53**	**0.54**	0.53	15049
Bullish	**0.53**	**0.51**	0.52	14935
Accuracy			**0.53**	29984
Macro Avg	0.53	0.53	0.53	29984
Weighted Avg	0.53	0.53	0.53	29984

We can see that the SVM with RBF Kernel still not improve the performance very much



Model Coparision

Graph of Accuracy between the models

- Decision Tree (Depth 10)
- Random Forest (Depth 10)
- AdaBoost (DT Depth 5 Base)
- Linear Kernel SVM
- RBF Kernel SVM

We can see that every model give the same performance on test set but when considner on training set and validation set AdaBoost give the best performance

How to improve ?

From the results, we see that the model performs only slightly better than random guessing (~6% improvement).
(Random guessing: 50% vs. Model accuracy: 53%), but the performance is still poor.
How can we improve?
Since the features were engineered manually, they might not be effective enough in capturing market trends.
To improve the model, we can focus on:

- Enhancing feature engineering by selecting more relevant indicators.
- Exploring additional technical or fundamental factors that may better represent market movements.
- Refining existing features to remove noise and improve signal quality.

Better feature engineering could lead to a more informative representation of the market, ultimately improving model performance.

Thank You For Watching my Presentation

01418362 Introduction to Machine Learning

6610401985 ไชยวัตน์ หนูวัฒนา