Machine Learning Techniques to develop Quant Trading strategies

Quant Trading strategies with ML

A trading strategy is a set of decisions

1. What **frequency** should we trade at?

   Daily/Weekly/Monthly

2. In each period, **which stocks** should we trade?

3. For each stock, should we **go long or short?**

Quant Trading strategies with ML

We can make these decisions the output of an ML model

Historical data → **Model** → Trading strategy

Quant Trading strategies with ML

To approach any problem using Machine learning

We can follow a standard set of steps

Quant Trading strategies with ML

Step 1: Define the problem statement

Step 2: Identify the class of ML problems it falls into

Step 3: Represent your data using features

Step 4: Use the data to train an ML algorithm

Step 5: Use the trained model as your trading strategy

Quant Trading strategies with ML

Step 1: Define the problem statement

Step 2: Identify the class of ML problems it falls into

Step 3: Represent your data using features

Step 4: Use the data to train an ML algorithm

Step 5: Use the trained model as your trading strategy

Let's say we want to find a daily trading strategy a security

Quant Trading strategies with ML

Step 1: Define the problem statement

Step 2: Identify the class of ML problems it falls into

Stock, Date → Go long/Short?

Step 3: Represent your data using features

Step 4: Use the data to train an ML algorithm

Step 5: Use the trained model as your trading strategy

Quant Trading strategies with ML

Step 1: Define the problem statement

Step 2: Identify the class of ML problems it falls into

Stock, Date → One of [-1, 1]

Step 3: Represent your data using features

Step 4: Use the data to train an ML algorithm

Step 5: Use the trained model as your trading strategy

Quant Trading strategies with ML

Step 1: Define the problem statement

Stock, Date → **One of [-1, 1]**

Step 2: Identify the class of ML problems it falls into

Step 3: Represent data using features

This is a classic example of

Step 4: Use the data to train an ML algorithm

a Machine Learning

Step 5: Use the trained model as your trading strategy

Classification Problem

Classification Problem

Stock, Date → One of [-1, 1]

Given a particular date, we want to classify it as an Up Day or Down Day for the stock

Classification Problem

Stock, Date → One of [-1, 1]

Up day -> Stock price will increase, Returns are positive

Classification Problem

Stock, Date → One of [-1, 1]

Down day -> Stock price will decrease, Returns are negative

Classification Problem

Stock, Date → One of [-1, 1]

Instead of just Up/Down, we can have more categories

Classification Problem

Stock, Date → One of [-3, -2, -1, 0, 1, 2, 3]

The sign -> up /down

The magnitude -> Conviction in the signal

Classification Problem

Stock, Date → One of [-3, -2, -1, 0, 1, 2, 3]

The categories here are called labels

Classification Problem

Stock, Date → One of [-3, -2, -1, 0, 1, 2, 3]

This signal tells us how big the long/short position on the stock should be

Classification Problem

Stock, Date → One of [-3, -2, -1, 0, 1, 2, 3]

We can choose any ML Classification algorithm to solve this problem

Classification Problem

Stock, Date → One of [-3, -2, -1, 0, 1, 2, 3]

Random Forests, Support Vector Machines, Gradient Boosted Classifiers are a few examples

Quant Trading strategies with ML

Step 1: Define the problem statement

Step 2: Identify the class of ML problems it falls into

This is a classic example of a Machine Learning Classification Problem

Quant Trading strategies with ML

These 3 are standard steps to solve any ML problem

Step 3: Represent your data using features

Step 4: Use the data to train an ML algorithm

Step 5: Use the trained model as your trading strategy

Quant Trading strategies with ML

Let's see how these would work for our ML classification problem

Step 3: Represent your data using features

Step 4: Use the data to train an ML algorithm

Step 5: Use the trained model as your trading strategy

Quant Trading strategies with ML

Step 1: Define the problem statement

For classification, these 2 steps form the training phase

Step 2: Identify the class of ML problems it falls into

Step 3: Represent your data using features

Step 4: Use the data to train an ML algorithm

Step 5: Use the trained model as your trading strategy

Quant Trading strategies with ML

Step 1: Define the problem statement

For classification, these 2 steps form the training phase

Step 2: Identify the class of ML problems it falls into

Steps 3, 4: Training phase

Step 5: Use the trained model as your trading strategy

Quant Trading strategies with ML

Step 1: Define the problem statement

Step 2: Identify the class of ML problems it falls into

Steps 3, 4: Training phase

In the training phase, the algorithm will use past data to identify relationships/patterns

Step 5: Use the trained model as your trading strategy

Quant Trading strategies with ML

Step 1: Define the problem statement

Step 2: Identify the class of ML problems it falls into

Steps 3, 4: Training phase

This is called the test phase

Step 5: Use the trained model as your trading strategy

Quant Trading strategies with ML

Step 1: Define the problem statement

Step 2: Identify the class of ML problems it falls into

Steps 3 4: Training phase

In the test phase we use the patterns previously identified to find the strategy on a given day

Step 5: Test phase

Quant Trading strategies with ML

Step 1: Define the problem statement

Let's understand in detail what happens in each phase

Step 2: Identify the class of ML problems it falls into

Steps 3, 4: Training phase

Step 5: Test phase

Training phase

In the training phase we start with a lot of historical data

This data has to be in a certain format

Stock, Date, Features, Label

Training phase

Stock, Date, Features, Label

The label is the category this stock, date belong to

One of
[-3, -2, -1, 0, 1, 2, 3]

Training phase

Stock, Date, Features, Label

[-3, -2, -1, 0, 1, 2, 3]

The label will depend on the actual returns that we saw on that date

Training phase

Stock, Date, Features, Label

[-3, -2, -1, 0, 1, 2, 3]

Returns between -0.5% and 0.5%

Training phase

Stock, Date, Features, Label

[-3, -2, -1, 0, 1, 2, 3]

[-1.5%, -0.5%) and (0.5%,1.5%]

Training phase

Stock, Date, Features, Label

[-3, -2, -1, 0, 1, 2, 3]

[-2.5%, -1.5%) and (1.5%,2.5%]

Training phase

Stock, Date, Features, Label

[-3, -2, -1, 0, 1, 2, 3]

(-inf, -2,5%) and (2,5%,inf)

Training phase

Stock, Date, Features, Label

This is a set of attributes that describe this datapoint

Training phase

Stock, Date, Features, Label

It can include calendar features

Month of year

Day of week

Day of the month

Trading day in Month

Training phase

Stock, Date, Features, Label

Calendar features will capture any seasonal effects in the price movements

Training phase

Stock, Date, Features, Label

We would also want to include price movements relative to this date

Momentum

Jump

Training phase

Stock, Date, Features, Label

For the NIFTY, we can include the P/E ratio as a factor

Training phase

Stock, Date, Features, Label

We can use the Momentum, Jump of other related stocks/indices as features

For example, NIFTY could use BANKNIFTY as an input

Training phase

Stock, Date, [CalendarFeatures, Momentum, Jump, Features of Related stocks], Label

We can already see the advantages of using ML over Excel

Training phase

Stock, Date, [CalendarFeatures, Momentum, Jump, Features of Related stocks], Label

The number of features that we can use is huge

Training phase

Stock, Date, [CalendarFeatures, Momentum, Jump, Features of Related stocks], Label

It's upto the classification algorithm to figure out the relationship between the features and the label

Training phase

Stock, Date, [CalendarFeatures, Momentum, Jump, Features of Related stocks], Label

The data in this form used to train an ML classifier

Training phase

Stock, Date, [CalendarFeatures, Momentum, Jump, Features of Related stocks], Label

ML Classifier

Training phase

Trained ML Classifier

Stock, Date,
[CalendarFeatures,
Momentum, Jump,
Features of Related
stocks], Label

By looking at the training data, this classifier has "learnt" how to classify any Stock, Date

Test phase

Stock, Date,
[CalendarFeatures,
Momentum, Jump,
Features of Related
stocks]

Trained ML
Classifier

Label

One of
[-3, -2, -1, 0, 1, 2, 3]

Test phase

Trained ML Classifier

The inner workings of this Trained classifier are somewhat opaque to us

Test phase

Trained ML Classifier

With most algorithms, it's difficult to articulate how the input factors are being combined to decide the label

Quant Trading strategies with ML

Step 1: Define the problem statement

The training and test phases are implemented in Python using a library called Scikit learn

Step 2: Identify the class of ML problems it falls into

Steps 3, 4: Training phase

Step 5: Test phase

Scikit Learn

Scikit learn contains a large number of built -in ML classifiers

RandomForestClassifier

SVC

GradientBoostedClassifier

Scikit Learn

Each classifier has 2 methods

fit, predict

Scikit Learn

The fit method is used for the training step

Stock, Date, [CalendarFeatures, Momentum, Jump, Features of Related stocks], Label

ML Classifier.fit()

Scikit Learn

The fit method is used for the training step

Trained ML Classifier

Scikit Learn

The predict method is used for the training step

Stock, Date,
[CalendarFeatures, Momentum, Jump, Features of Related stocks]

→

Trained ML Classifier .predict()

→

Label

One of
[-3, -2, -1, 0, 1, 2, 3]

Scikit Learn

fit, predict

Knowing these 2 methods, we can use any classifier in the Scikit learn library

Scikit Learn

fit, predict

However, there are some parameters which can be tuned for individual classifiers

Scikit Learn

fit, predict

Let's start with a RandomForestClassifier

RandomForestClassifier

Insert Decision trees and Random forests theory

RandomForestClassifier

A decision tree is a good way to visualize a trading strategy

A trading strategy is nothing but a set of rules

RandomForestClassifier

Each rule would be of the form

If Condition1 (Feature1) {

If Condition2 (Feature2){

If Condition3 (Feature3)......

{ Label}

RandomForestClassifier

These rules can be collectively visualized as a decision tree

If Condition1 (Feature1) {

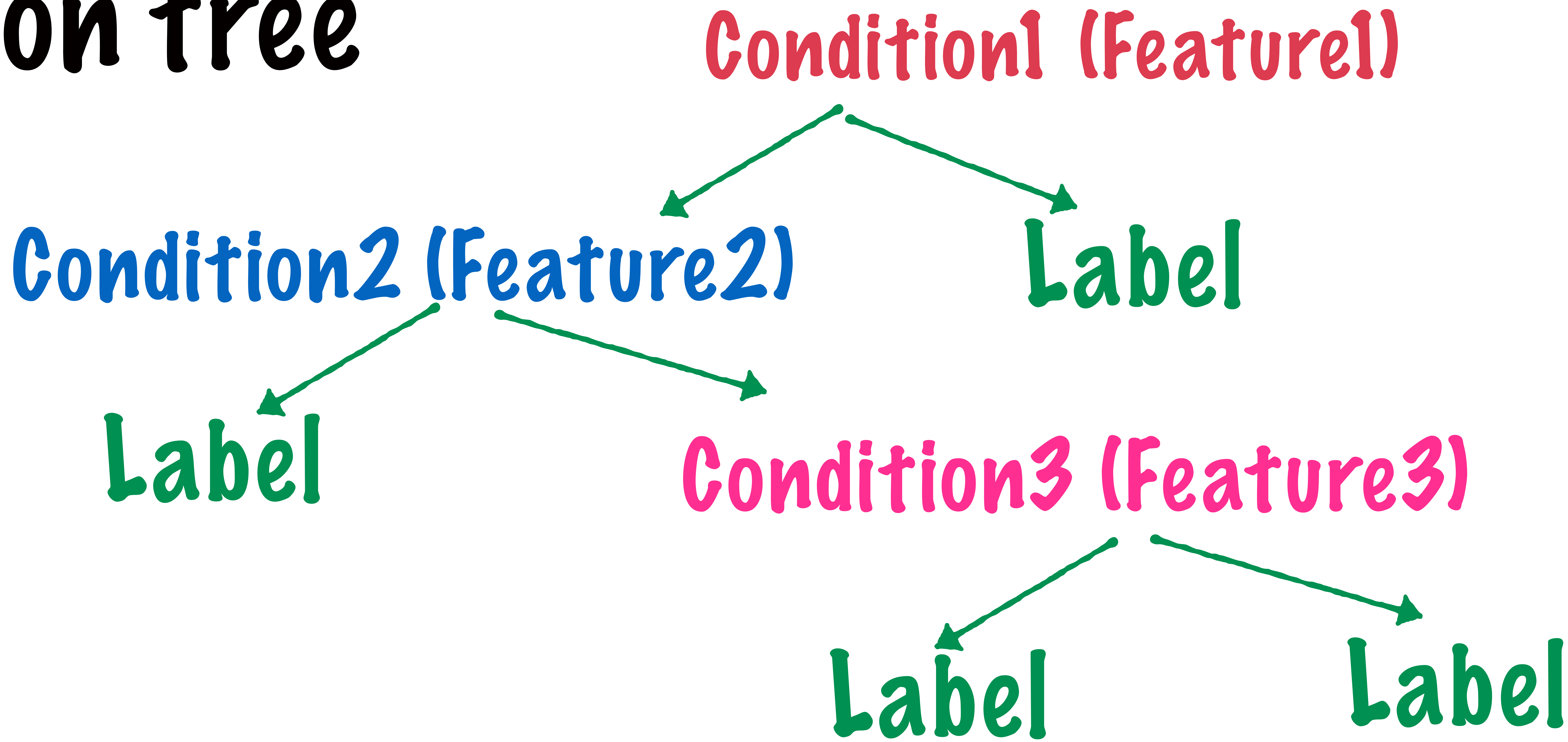
If Condition2 (Feature2) {

If Condition3 (Feature3)......

{ Label}

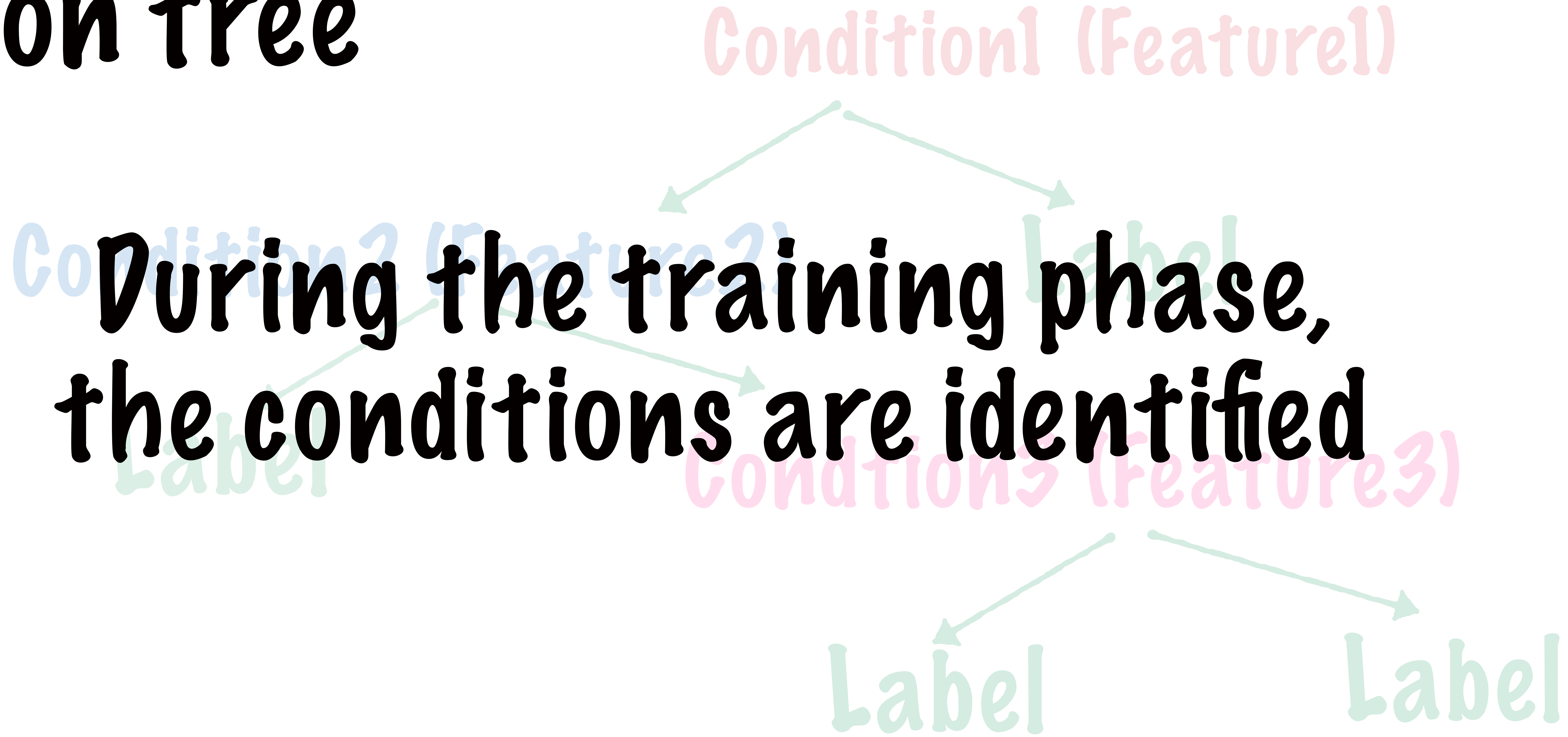
RandomForestClassifier

Decision tree



RandomForestClassifier

Decision tree



During the training phase,
the conditions are identified

RandomForestClassifier

A random forest builds a collection of decision trees

Each will have a different set of conditions

RandomForestClassifier

Each tree is identified using a

Random subset of the training data

Random subset of the features

RandomForestClassifier

The final label is a majority vote of all the trees in the forest

RandomForestClassifier

In ScikitLearn

There are 2 important parameters used to tune the RandomForestClassifier

RandomForestClassifier

In ScikitLearn

n_estimators This is the number of trees the Random forest will build

random_state This is a seed used for the random selection of the training data and feature subsets

RandomForestClassifier

In ScikitLearn

n_estimators The more the number of features, the higher this should be

random_state

RandomForestClassifier

In ScikitLearn

n_estimators

100 is a standard choice for the number of trees

random_state

RandomForestClassifier

In ScikitLearn

n_estimators

random_state

If this is not explicitly set, the classifier will give different results each time it's run

RandomForestClassifier

In ScikitLearn

n_estimators

random_state

This should be chosen at the outset of the exercise and kept constant

Running a backtest on RandomForestClassifier

Backtest on RandomForestClassifier

1. Train a RandomForest Classifier using data from the period 2009-2013

2. To this trained classifier, pass the 2013-2016 data for backtesting

Backtest on RandomForestClassifier

From the trained classifier we get

Date, Signal

If we had used this trading strategy

Strategy Returns = Signal*Returns

Backtest on RandomForestClassifier

Here are a few different metrics we should look at from the backtest

Average returns

Risk (ie Standard Deviation)

Sharpe Ratio

These measures tell us how good the strategy is

Backtest on RandomForestClassifier

Here are a few different metrics we should look at from the backtest

Average returns

Risk (ie Standard Deviation)

Sharpe Ratio

We can use them to compare multiple classifiers (each using different features)

Backtest on RandomForestClassifier

Average returns

Risk (ie Standard Deviation)

Sharpe Ratio

In addition to these, we can also look at

Skewness of Returns

%Up, %down

These measures tell us how robust the strategy is

Backtest on RandomForestClassifier

Average returns

Risk (ie Standard Deviation)

Sharpe Ratio

Skewness of Returns

%Up, %down

Skewness tells us if there is any tail risk to the strategy

ie. is there a risk of a large downside with a very low probability

Backtest on RandomForestClassifier

Average returns

Risk (ie Standard Deviation)

Sharpe Ratio

Skewness of Returns

%Up, %down

This tells us how often our model is actually right

Improving the Backtest with expanding training period

Backtest on RandomForestClassifier

In our first backtest

We trained a RandomForest Classifier using data from the period 2009-2013

To this trained classifier, we passed the 2013-2016 data for backtesting

Backtest on RandomForestClassifier

In effect

We were predicting the signal in 2016 , using only history till 2013

This is not a realistic backtest

Backtest on RandomForestClassifier

In our new backtest

For each datapoint in 2013-2016

1. Train a classifier using data from 2009-[CurrentDate-1]

2. Use this classifier to find the signal for CurrentDate

Backtest on RandomForestClassifier

In effect

We are **recalibrating the classifier** with the most current data

Using a categorical variable as a feature

Using a categorical variable as a feature

One important advantage of using machine learning is the ability to use categorical variables as features

Using a categorical variable as a feature

Let's **incorporate a categorical variable** into our RandomForestClassifier

We have used **Momentum and Jump** as features in our classifier

Using a categorical variable as a feature

Momentum and Jump

These features capture the overall trend across the last few weeks/months

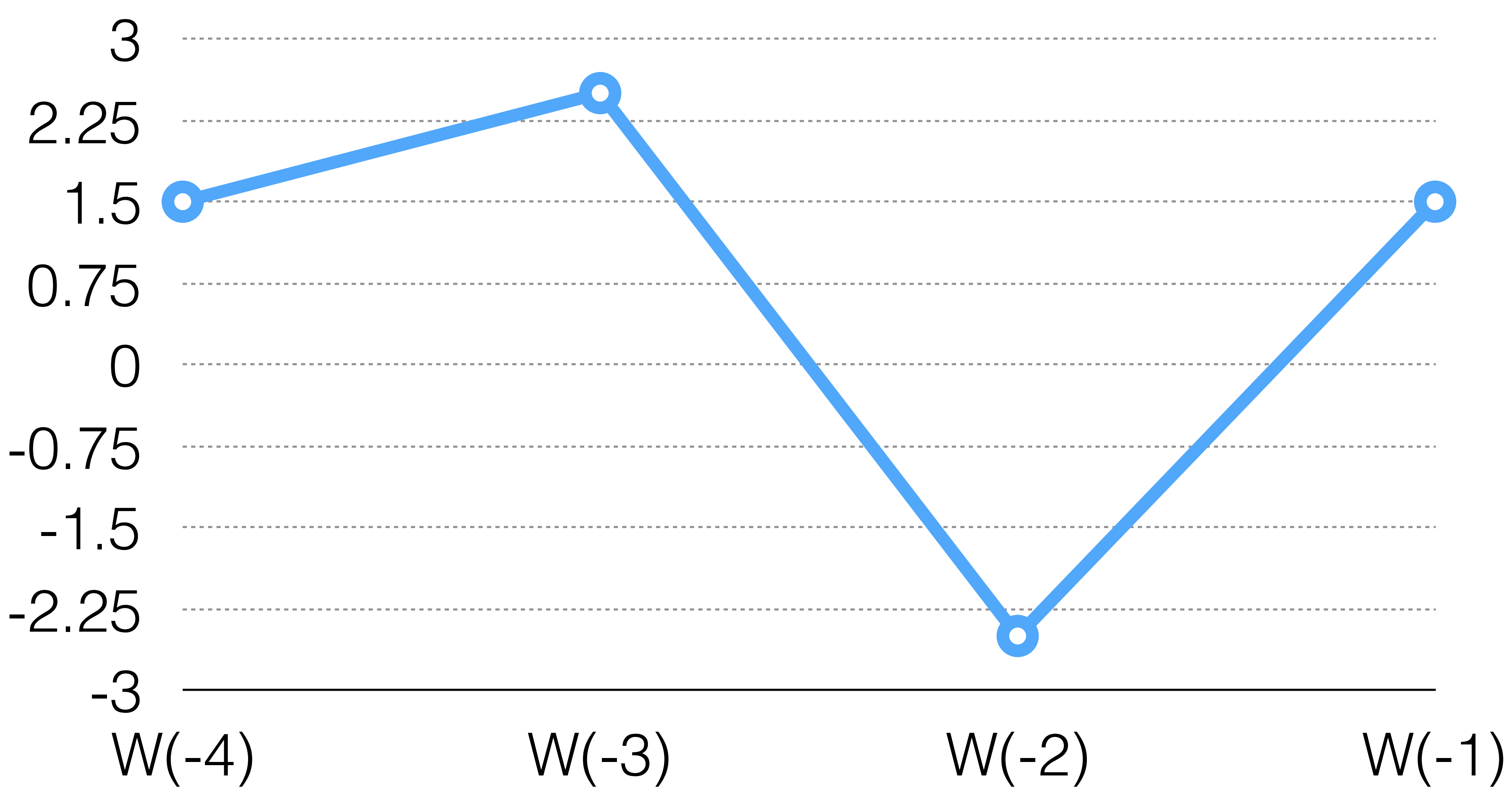
Using a categorical variable as a feature

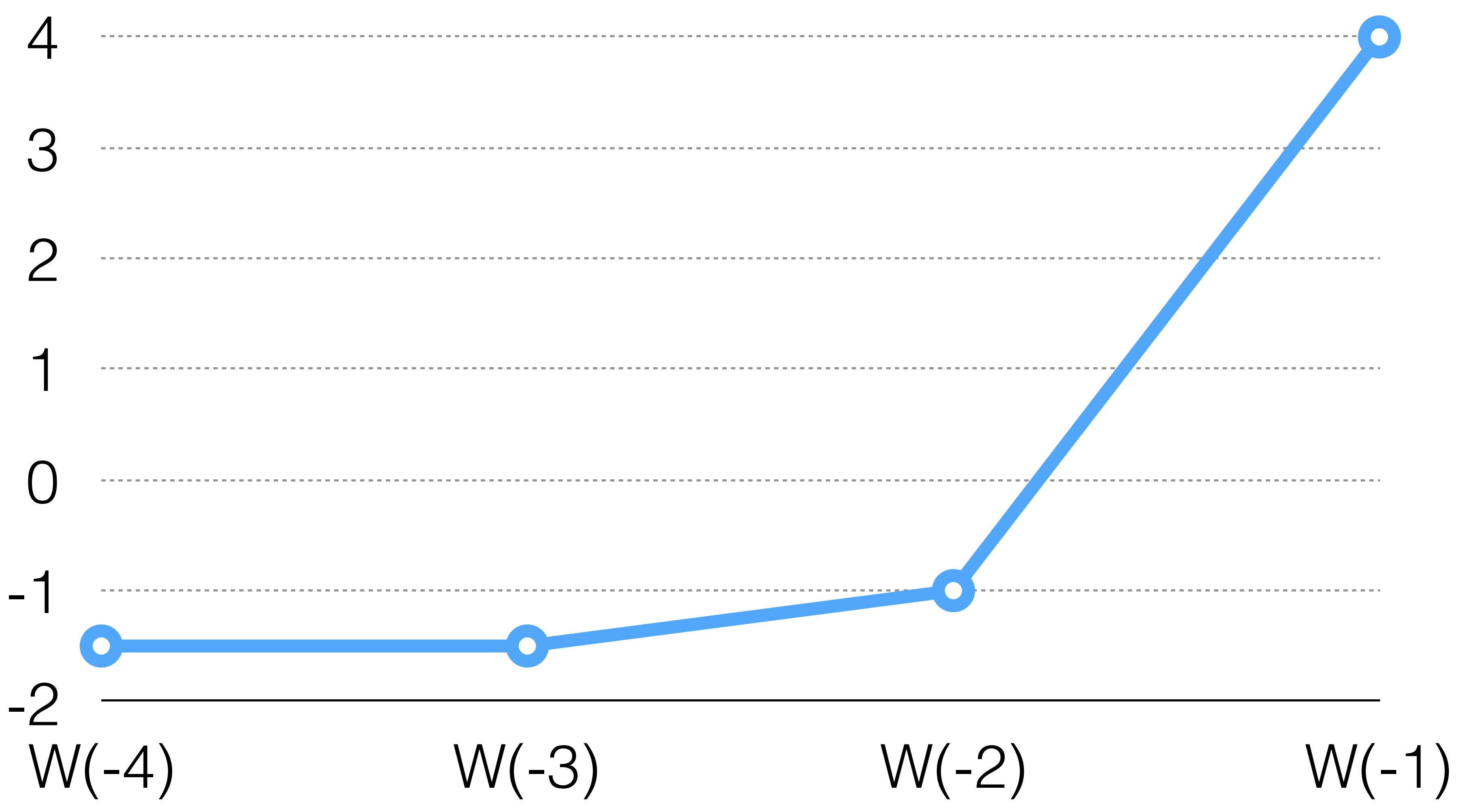
Momentum and Jump

Could we incorporate a feature which captures the trend in each of the prior 4 weeks?

prevWeeks feature

Here is the trend in past 4 weeks for 2 dates

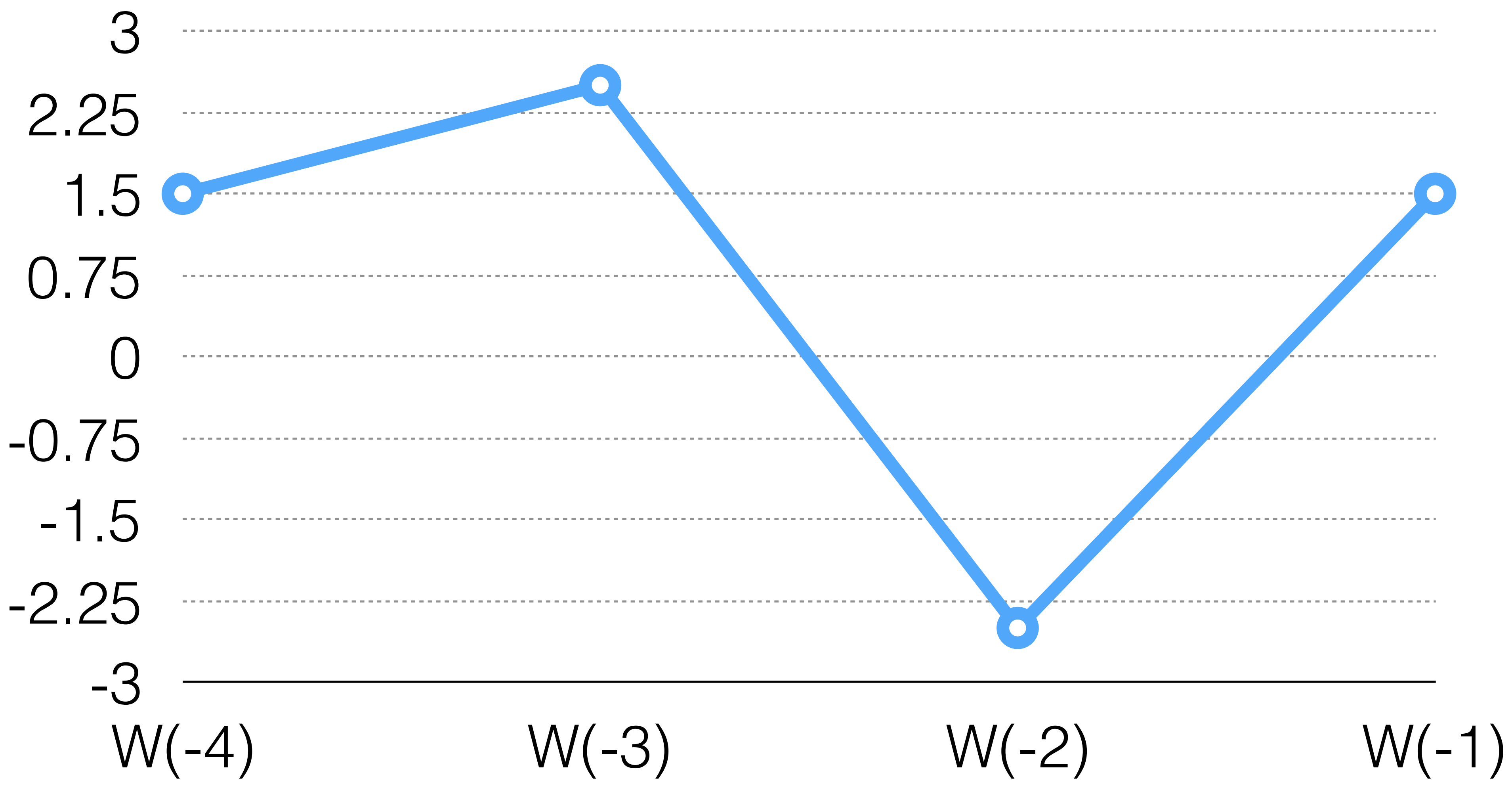


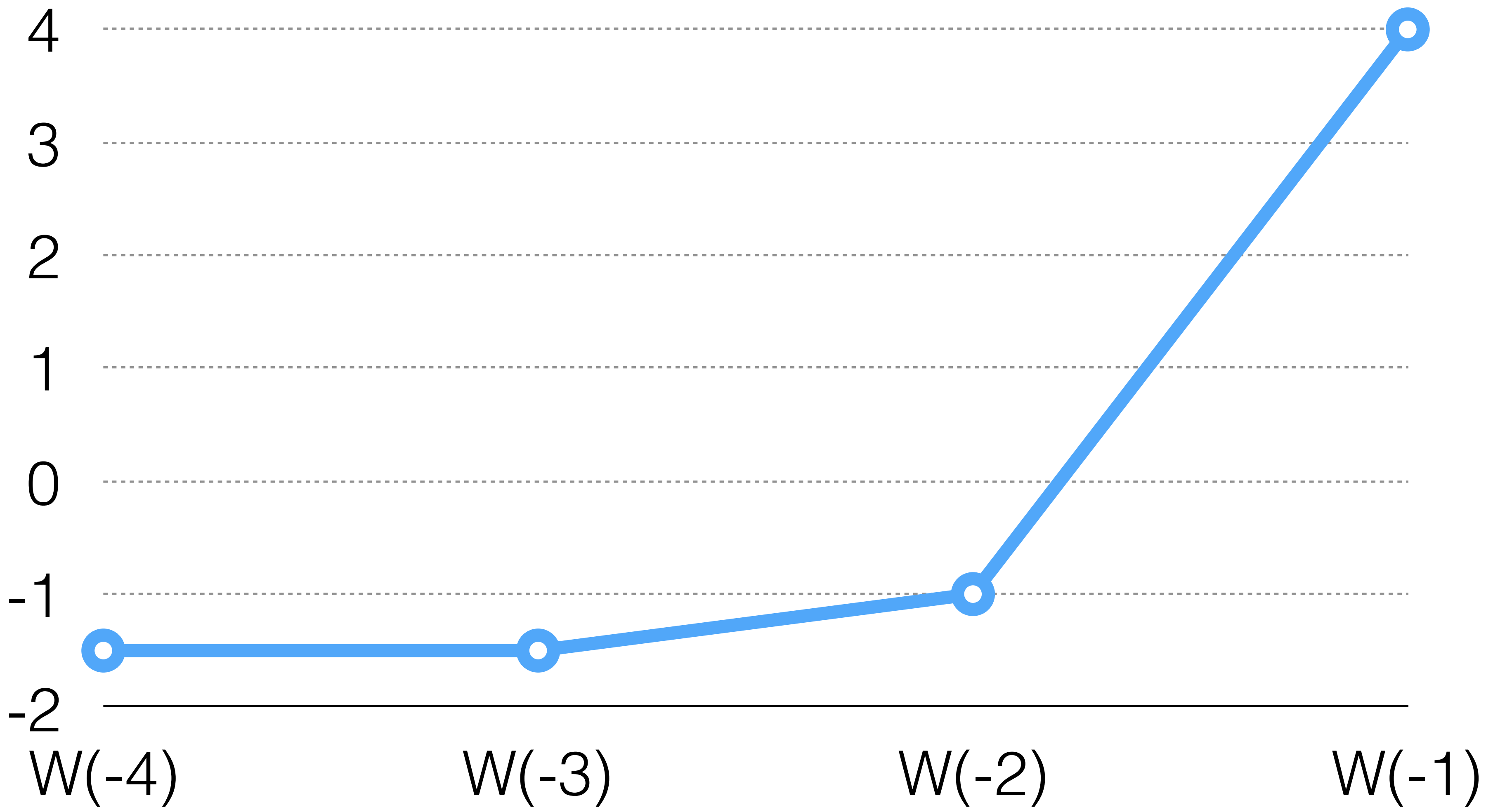


For both, Sum of returns for 4 weeks = 3%

prevWeeks feature

Here is the trend in past 4 weeks for 2 dates





Up, Up, Down, Up

Down, Down, Down, Up

prevWeeks feature

Up, Up, Down, Up

Down, Down, Down, Up

Our categorical variable needs to capture this pattern

prevWeeks feature

Compute a 4 digit number

$X_1X_2X_3X_4$

Each X_i represents the trend in the corresponding week

prevWeeks feature

Each X_i is one of [1,2,3]

1, if Returns in [-1.5%,1.5%]

2, if Returns >1.5%

3, if Returns <1.5%