Bayes Classifier using Gaussian Prior

Author: George Barrinuevo

November 2025

1 Introduction

1.1 Purpose

The purpose of this document is to implement Regime Detection for financial trading. This implementation uses the Naive Bayes Classifier using Gaussian Prior method. Both the underlining mathematics/statistics and some of the code are included. Will use both continuous and categorical calculations. But this implementation will concentrate on computing the continuous values and then converting them to categorical values.

1.2 Common Equations

In this section, you can find commonly used mathematics and statistics used in this implementation.

Compute the mean:

$$\mu = \frac{1}{N}\sum_{t=1}^{N} r_t \tag{1.2.1}$$

Compute the standard deviation:

$$\sigma = \sqrt{\frac{1}{N-1}\sum_{t=1}^{N_t}(r_t - \mu)^2} \tag{1.2.2}$$

Compute the variance:

$$\sigma^2 = \frac{1}{N-1}\sum_{t=1}^{N_t}(r_t - \mu)^2 \tag{1.2.3}$$

Compute log-returns:

$$r_t = \text{LogReturns}_t = \ln\frac{P_t}{P_{t-1}} \tag{1.2.4}$$

Create a series of log-returns:

$$\text{LogReturns}_t, \ldots, \text{LogReturns}_T = r_1, \ldots, r_T \tag{1.2.5}$$

Where:

μ: average. Usually ≈ 0 (small, near 0), so classification becomes symmetric.
σ: standard deviation (volatility)
σ^2: variance
N_t: Number of data points, time t
P_t: prices at time t, $t = 1$ is first price, $t = t$ is current price.
t: time, t=1 is first time, t=t-1 is previous, and t=t is latest time.
$r_1, r_2, \ldots, r_T$: series of log-returns. For r_t, $t = 1, \ldots, T$.

Pick a multiplier k that controls how "strict" you want to be. This is used as a threshold when converting continuous to categorical values:

$$k = \begin{cases} 0.5 & \text{More sensitive. More bars labeled positive/negative, fewer as "near 0".} \\ 1 & \text{(most common) Noise-filtered. Only bigger moves are positive/negative, more "near 0".} \\ 1.5 & \text{Very strict.} \end{cases} \tag{1.2.6}$$

The best way to select k is to do cross-validation which will try different values for k and select the k with best performance.

Calculate parameter θ:

$$\theta = k\sigma \tag{1.2.7}$$

This decision criteria converts the continuous (float numbers) to categorical values.:

$$x_f = \begin{cases} \text{Negative} & \text{if } r < \mu - \theta \\ \text{NearZero} & \text{if } \mu - \theta \leq r \leq \mu + \theta \\ \text{Positive} & \text{if } r > \mu + \theta \end{cases} \tag{1.2.8}$$

The window size w is how far back to look. Here are more examples on how to pick window w:

Time Frame	**Predict Next**	W	**Meaning**
1-minute	5-minutes	5	small micro-trend
	15-minutes	15	intraday patterns
	30-minutes	30	half-hour move
	1-hour	60	medium-term intraday move
5-minutes	30-minutes	6	small intraday signals
	1-hour	12	stable prediction window
	3-hours	36	decent short term trend
	full day ($\approx$ 6.5h)	78	long intrday prediction
1-day	1-week	5	5 trading days
	2-weeks	10	short swing
	1-month	20	typical monthly cycle
	3-month	60	quarterly cycle
	1-year	252	trading-year horizon

Table 1: Setting window w

The best way to select window W is to do cross-validation which is to try different values for W and select the W with best performance. For predicting regime change, select the time horizon to be 20-60 days (most common), must convert this to window W.

1.3 Data Set

Since this is an ML implementation of Baye's Theorem, the model must be trained. The data is divided in to 30% train & 70% test sets. This model must first be trained before it can be used for inference/predictions. The test data is used to evaluate its accuracy, reliability, and etc and to ensure it is not over fitted. Parameter tuning is recommended to select good values for w, k, θ.

1.4 Classes

The classes and their corresponding labels must be selected. An example of this is the commonly used 5-classes. In this implementation, we will use the below labels. See Table 2.

1.5 Features

Features are the inputs to the model. For example log-returns, volatility, distance from MA. They can be continuous or categorical and they encode information about the data. This table list some examples of features. See Table 3. The feature type is what to apply to prices, log log-return or distance from MA. Categorical values is like in equation 1.8, like Negative, Positive, NearZero. Continuous value is the float number, like float values of log-returns.

Class Label	Index	Abbreviation
StrongUptrend	0	SU
WeakUptrend	1	WU
SideWays	2	SW
WeakDowntrend	3	WD
StrongDowntrend	4	SD
UnCertain	5	UC

Table 2: 5-class labels

Feature Type
mean
variance
return
lognormal of return
slope
MA, RSI, MACD
volatility
distance from MA
cumulative log-return
regression slope
MA slope
volatility-adjusted returns
log-return momentum
10-day moving average slope

Table 3: Arbitrary List of Feature Type

For this implementation, we will use these features.

Feature Type	Feature Categorical Labels
Slope	positive, near 0, negative
LogNormal of Returns	negative, flat, positive
Volatility	????
Distance from MA	> 0, near 0, < 0

Table 4: Feature Type

Calculate Log-Returns

Continuous Values

From historical price data P, compute log-returns r:

$$r_t = \text{(See equation 1.2.4)} \tag{1.5.1}$$

This is needed if using float numbers for x. An example of this is: 1.428, 10.568, etc. Store the continuous (float numbers) returns like this:

$$x_{fc} = r_t \tag{1.5.2}$$

Where:

t: time $t = 1, \ldots,$ N
f: feature f

c: class c
x_{fc}: data for feature f and class c, a vector

Categorical Values

This is used to convert continuous (float numbers) to categorical (discrete labels) values. An example of categorical values are: Negative, Near 0, Positive.

Calculate the μ and σ over time $t = 1 \dots N$:

$$\mu_t = \text{mean of slope (See equation 1.2.1)} \tag{1.5.3}$$

$$\sigma_t : \text{std dev of slope (See equation 1.2.2)} \tag{1.5.4}$$

Pick a multiplier k that controls how "strict" you want to be:

$$k = \text{"strictness" parameter (See equation 1.2.6)} \tag{1.5.5}$$

Calculate the parameter θ:

$$\theta = \text{(See equations 1.2.7)} \tag{1.5.6}$$

Then your decision criteria is the following which converts the continuous (float numbers) to categorical values.:

$$x_f = \text{(See equations 1.5.2)} \tag{1.5.7}$$

Where:
x_f: data points (categorical values) for feature f, for all f

Calculate Slope

Continuous Values

Calculate log-returns:

$$r_t = \text{log-return (See equation 1.2.4)} \tag{1.5.8}$$

Where:
$r_1, r_2, \dots, r_T$: series of log-returns. For r_t, $t = 1, \dots, T$.

Calculate the regression slope that represents the average log-return per bar over the window.

Calculate log-returns:

$$r_t = \text{log-return (See equation 1.2.4)} \tag{1.5.9}$$

Calculate the cummulative log-returns:

$$R_t = \sum_{i=1}^{N} r_i \tag{1.5.10}$$

Regression Model. Fit a line:

$$R_t = \alpha + \beta \cdot t + \epsilon_t \tag{1.5.11}$$

Regression slope β:

$$\beta = \frac{Cov(t, R_t)}{Var(t)} \tag{1.5.12}$$

$$\beta = \frac{\sum_{t=1}^{N}(t - \bar{t})(R_t - \bar{R})}{\sum_{t=1}^{N}(t - \bar{t})^2} \tag{1.5.13}$$

Calculate the means:

$$\bar{t} = \frac{1}{N}\sum_{t=1}^{N} t, \quad \bar{R} = \frac{1}{N}\sum_{t=1}^{N} R_t \tag{1.5.14}$$

Where:

- t: bar index (1, 2, ..., N)
- $\bar{t}$: mean of indices
- R: cumulative log-returns
- $\bar{R}$: mean of cumulative log-return
- β: slope = average log-return per bar

Create a series of slopes:

$$slope_N = \beta \tag{1.5.15}$$

$$s_1, s_2, \ldots, s_N = slope_1, slope_2, \ldots, slope_N \tag{1.5.16}$$

This slope represents average log-return per bar over the window. There is a simpler equivalent formula, but this regression version will reduce unwanted noise; use the simple difference if you want speed.

Estimate typical slope volatility.

Compute:

$$\mu_s = \text{mean of slopes} = \text{(See equation 1.2.1)} \tag{1.5.17}$$

$$\sigma_s = \text{std dev of slopes} = \text{(See equation 1.2.2)} \tag{1.5.18}$$

Where:

- s: All slopes in the set $\{s_1, ..., s_t\}$, for all slopes

Choose a threshold using:

$$k\sigma \tag{1.5.19}$$

Pick a sensitivity constant:

$$k = \text{Multiplier to specify the strictness of the threshold (See equation 1.6)} \tag{1.5.20}$$

Calculate the parameter :

$$\theta = \text{(See equations 1.2.7)} \tag{1.5.21}$$

Categorical Values

Find the categorical values based on slope s. This is the criteria to convert continuous (float numbers) to categorical values.

$$x_f = \begin{cases} \underline{\text{if } (|\mu_s|/\sigma_s) < 0.25 \text{ (almost always true)}} & \\ \text{Negative} & \text{if } s < -\theta \\ \text{NearZero} & \text{if } |s| \leq \theta \\ \text{Positive} & \text{if } s > \theta \\ \\ \underline{\text{if } (|\mu_s|/\sigma_s) \geq 0.25 \text{ (rare)}} & \\ \text{Decision Criteria: (See equation 1.8)} & \end{cases} \tag{1.5.22}$$

1.6 Training - Find Class for Features

The data must be trained using training data before using it on test data. One of the most quant-finance friendly methods is the Log-Return Momentum Method. This method should not be used for the Testing phase since it does not use the Baye's Theorem. Only use for Training phase.

Continuous Values

The rolling cumulative log-return is computed:

$$R_t^{(w)} = \sum_{i=t-w+1}^{t} r_t \tag{1.6.1}$$

Rolling Cumulative Log-Returns

Where:

r_t: Log-Returns of prices.
w: The window w size. This is how much to look back in the data series. See Table ???.

Categorical Values

This is the α_1 and α_2 thresholds for the decision criteria to convert continuous (float numbers) to categorical values.

Parameter	**Threshold**	**Meaning**
α_1	0.3%	Weak threshold
α_2	1.0%	Strong threshold

Table 5: α Threshold Levels

Here is the class labels for class C_t. This decision criteria converts continuous (float numbers) to categorical values. This is shown in numerical index order from strong to weak trends. The index number is the python variable index number.

$$C_t = \begin{cases} \text{Class} & \text{Index} & \text{Abbreviations} & \text{Conditions} \\ \hline \\ \text{StrongUptrend} & 1 & \text{SU} & \text{if } R_t > \alpha_2 \\ \text{WeakUptrend} & 2 & \text{WU} & \text{if } \alpha_1 < R_t \leq \alpha_2 \\ \text{SideWays} & 3 & \text{SW} & \text{if } |R_t| \leq \alpha_1 \\ \text{WeakDowntrend} & 4 & \text{WD} & \text{if } -\alpha_2 \leq R_t < -\alpha_1 \\ \text{StrongDowntrend} & 5 & \text{SD} & \text{if } R_t < -\alpha_2 \\ \text{UnCertain} & 6 & \text{UC} & \text{else} \end{cases}$$

Where:

C_t: Categorical class C, time t.

1.7 Training - Find Likelihood

Calculate the Prior Gaussian Parameters for mean μ and variance σ^2. This is done on a per feature f and per class c. So, if you have 2 features and 5 classes, then you have 2 * 5 = 10 sets of μ and σ^2.

For each class and each feature, compute mean and variance:

$$\mu_{fc} = \text{mean of feature } f \text{ inside class } c \text{ (See equation 1.2.1)} \tag{1.7.1}$$

$$\sigma^2_{fc} = \text{variance of feature } f\text{inside class } c\text{(See equation 1.2.3)} \tag{1.7.2}$$

The Gaussian Probability Density Function (PDF) is:

$$\mathcal{N}(x_j \mid \mu_{jc}, \sigma^2_{jc}) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right) \tag{1.7.3}$$

Calculate the Gaussian Likelihood for all features j, given one class c:

$$\text{Likelihood} = P(x \mid y = c), \quad \text{(for 1 feature)} \tag{1.7.4}$$

$$P(x \mid y = c) = \prod_{f=1}^{N_f} P(x_j \mid y = c) = \prod_{f=1}^{N_f} \mathcal{N}(x_{fc} \mid \mu_{fc}, \sigma^2_{fc}) = \prod_{f=1}^{N_f} \frac{1}{\sqrt{2\pi}\sigma_{fc}} \exp\left(-\frac{(x_{fc}-\mu_{fc})^2}{2\sigma^2_{fc}}\right) \tag{1.7.5}$$

$$P(x \mid y = c) = \prod_{f=1}^{N_f} \frac{1}{\sqrt{2\pi}\sigma_{fc}} \exp\left(-\frac{(x_{fc}-\mu_{fc})^2}{2\sigma^2_{fc}}\right) \tag{1.7.6}$$

Where:

$\mathcal{N}$: Gaussian Distribution.
$\prod$: Product symbol. It is similar to $\sum$ but uses multiplication instead of addition.
f: Feature f.
N_f: Number of features.
c: Class c

1.8 Training - Find Gaussian Prior

The prior is simply the probability of each class before seeing any features. This method is called the 'Empirical Prior' and is the most common. It is estimated directly from the training data per class c. Most implementations (e.g., scikit-learn GaussianNB with prior=None) use this by default.

$$P(y = c) = P(C) = \frac{\text{number of training samples with class } c}{\text{total number of samples}} \tag{1.8.1}$$

Where:

<number of training samples with class c>: Filter for the one class c and count all data for all features.
<total number of samples>: Count all data without any filtering for all features and all classes.

1.9 Testing - Find Score

This is the Posterior. The $\propto$ sign means: it is proportional to. As one changes, the other changes in a consistent ratio.

The Score is calculated by multiplying the Likelihood and the Prior:

Bayes' Theorem:

$$P(C|X) = \frac{P(X|C) \cdot P(C)}{P(X)} \tag{1.9.1}$$

$$Posterior = \frac{Likelihood \cdot Prior}{Evidence} \tag{1.9.2}$$

Where:

Posterior: Means given the observed features X, what is the probability that the class is C?
Likelihood: Means given a class C, how likely is it to observe features X. How well does this class explain the data?
Prior: How likely is the class in general, before seeing the evidence.
Evidence: This is not needed here and may be removed in later equations.
c: Class c
X: feature

Posterior:

$$P(y = c \mid x) = P(C|X) \tag{1.9.3}$$

Likelihood:

$$P(x \mid y = c) = P(X|C) \tag{1.9.4}$$

Prior:

$$P(y = c) = P(C) \tag{1.9.5}$$

Calculate Score:

$$P(y = c \mid x) = \text{Prior} * \text{Likelihood} \tag{1.9.6}$$

$$P(y = c \mid x) \propto P(y = c)P(x \mid y = c) \tag{1.9.7}$$

$$P(y = c \mid x) \propto P(y = c) \prod_{f=1}^{N_f} \mathcal{N}(x_{fc} \mid \mu_{fc}, \sigma_{fc}^2) \tag{1.9.8}$$

In practice you use log-probabilities to avoid underflow. Compute LogScore(StrongUptrend), LogScore(WeakUptrend), etc, for all class c:

$$L_c = \text{LogScore}(c) = \log P(y = c) + \sum_{f=1}^{N_f} \log \mathcal{N}(x_{fc} \mid \mu_{fc}, \sigma_{fc}^2) \tag{1.9.9}$$

$$\log \mathcal{N}(x_f \mid \mu_{fc}, \sigma_{fc}^2) = -\frac{1}{2} \log(2\pi\sigma_{fc}^2) - \frac{(x_{fc} - \mu_{fc})^2}{2\sigma_{fc}^2} \tag{1.9.10}$$

$$L_c = \text{LogScore}(c) = \log P(y = c) + \sum_{f=1}^{N_f} \left[-\frac{1}{2} \log(2\pi\sigma_{fc}^2) - \frac{(x_{fc} - \mu_{fc})^2}{2\sigma_{fc}^2} \right] \tag{1.9.11}$$

Where:

$\mathcal{N}$: Gaussian Distribution.
$\prod$: Product symbol. It is similar to $\sum$ but uses multiplication
N_f: the number of features, this is the count of the categorical values instead of addition.
L_c: The LogScore(), for each class c
c: class c
f: feature f

For all L_c, find the maximum value, which is the predicted class c. Then take its python variable index, e.g. if index=0, then categorical value is 'SU', if index=4, then it is 'SD':

$$\hat{y} = \arg \max_{c \in \{\text{StrongUptrend,WeakUptrend,...}\}} L_c \tag{1.9.12}$$

If you want the actual posterior probabilities of L_c. This converts L_c to posterior probabilities, which is already calculated above:

$$P(y = c \mid x) = \text{normalize} = \frac{\exp(max(L_c))}{\sum_{c'} \exp(max(L_c))} \tag{1.9.13}$$

Where:

L_c: The LogScore(), for each clas c
$P(y = c \mid x)$: Posterior
c': All classes
c: These are categorical values in index ascending order.

Then, select the class with the largest posterior probabilities. Next, apply a threshold.

1.10 Testing - Evaluate using Threshol

Take the highest L_c and compare it with threshold values. This Threshold prevents whipsaws, false regime flips, micro noise triggering regime changes and etc. They ensure classification is stable. If the highest L_c passes the Threshold value, than the detected class was found. Else, assign its class to 'Uncertain'. Here are the recommended thresholds for Trading Regimes, see Table 6.

Threshold Meaning	**λ Threshold**
Fast switching / reactive	0.50 – 0.55
Balanced trading	0.60 – 0.70
Stable regime detection	0.70 – 0.80
Very slow regime switching	> 0.80
Uncertain	else
Most common for quants	0.65 – 0.75

Table 6: Threshold Levels for Posterior Probabilities

Compare $P(y = c \mid x)$ with the threshold level λ. If it passes this test, then the Posterior Probabilities or final solution will is accepted, else reject it.

1.11 Variables

The python variables to use is defined here. This shows how to store the data in well defined variables with descriptive names and etc.

Variable Name	**indexing**	**Example**	**Meaning**
P	[i]	11.23	Raw Price
x[i]	–	–	Feature
	['number']	32.93	Feature float number
	['categorical']	'Negative'	Feature categorical
	['feature']	'LogReturns'	Feature c
	['class']	'UpTrend'	Feature class c

Table 7: Python variables for features

Variable Name	indexing	Example	Meaning
z[i]	–	–	Feature calculations
	['mean']	26.32	mean of feature
	['std_dev']	26.32	std dev of feature
	['variance']	26.32	variance of feature
	['rolling_cum_log_returns']	26.32	rolling cumulative log-returns of feature

Table 8: Python variables for calculations

1.12 Python Code

```
import numpy as np
from enum import Enum, auto

#
class CLASS_GROUP(Enum):
    StrongUptrend = auto()
    WeakUptrend = auto()
    SideWays = auto()
    WeakDowntrend = auto()
    StrongDowntrend = auto()
    UnCertain = auto()

class FEATURE_GROUP(Enum):
    LogReturn = auto()
    Slope = auto()

class FEATURE_CATEGORICAL(Enum):
    Negative = auto()
    NearZero = auto()
    Positive = auto()

class FEATURE_COMMON_CALC(Enum):
    mean = auto()
    stdDev = auto()
    variance = auto()
    rollingCumLogReturn = auto()


print(CLASS_GROUP.SideWays)         # CLASS_GROUP.SideWays
print(CLASS_GROUP.SideWays.name)    # 'SideWays'
print(CLASS_GROUP.SideWays.value)   # 3

#
def make_array(shape, fill=0):
    return np.full(shape, fill)

x = make_array((2, 3, 4), fill=0)

x[1, 1, 2] = CLASS_GROUP.SideWays.value
print(x)
```

1.13 ToDo

- For code, create variables names.
- Divide in to Train and Test dataset.
- Create new section 'Data Set' using 30/70%. Check if have enough data items in train for each class. Add variables. Create numpy specific variables and how to search it.

- Adde section 'code'.
- Mention it uses ML.
- Tools: overleaf.com
- Combine 2 equations in 'Testing - Find Prior'.
- Combine 2 equations in 'Training - Find Likelihood'.
- Combine 2 equations in 'Training - Find Score'.
- In 'Training - Find Class for Features', add a 'where' for price P.
- Explain the need to divide the data set to Testing & Training, say 30/70%.