Generative and Discriminative Classification Models

A Mathematical And Visual Analysis

Sanidhya Tyagi
Indian Institute of Technology Kharagpur
Kharagpur, West Bengal, India

February 2026

Abstract

This report provides a robust mathematical and graphical analysis of different generative (Naive Bayes, Gaussian Mixture Models) and discriminative models (Logistic Regression, Support Vector Machines) with probabilistic derivations and interpretations for each. It also compares how they reach posterior probabilities using different hard or soft assumptions. This report also examines the final decision boundaries predicted by these models and concludes with guidance on how to choose an appropriate classification model for a given dataset.

Contents

1 INTRODUCTION

Generative classifiers are modeled around the probability $p(\boldsymbol{x}|y)$ (and consequently $p(y)$) if we consider the input features as $\boldsymbol{x}$ and the output labels as y. With these two values, a model can hence compute the posterior probability $p(y|\boldsymbol{x})$ using the standard Bayes Theorem given by:

Theorem 1.1 (Bayes Theorem)**.** Given joint distribution $p(\boldsymbol{x}, y)$, the posterior is:

$$p(y|\boldsymbol{x}) = \frac{p(\boldsymbol{x}|y)p(y)}{p(\boldsymbol{x})} = \frac{p(\boldsymbol{x}, y)}{\sum_{y'} p(\boldsymbol{x}, y')}, \tag{1}$$

where $p(\boldsymbol{x}|y)$ is learned explicitly and $p(\boldsymbol{x})$ serves as a normalizing constant.

Discriminative Classifiers on the other hand learn directly the decision boundary directly from $\boldsymbol{x}$ ($p(y|\boldsymbol{x})$ in case of logistic regression) given by binomial labels $\{0, 1\}$ (in our consideration). This makes them easier to train, and more insensitive to distribution assumptions.

2 GENERATIVE MODELS

2.1 Naive Bayes

2.1.1 Complete Set of Assumptions

Assumption 2.1 (Binary Features)**.** Each feature $x_j^{(i)} \in \{0, 1\}$, $j = 1, \ldots, n$.

Assumption 2.2 (Class Prior)**.** $P(y = 1) = \phi_y \in (0, 1)$, $P(y = 0) = 1 - \phi_y$.

Assumption 2.3 (Naive Independence)**.** $p(\boldsymbol{x} \mid y) = \prod_{j=1}^{n} p(x_j \mid y)$ (conditional independence given class).

2.1.2 Probabilistic Derivation

Step 1: Full Generative Model

Joint distribution for observation i:

$$p(\boldsymbol{x}^{(i)}, y^{(i)} = c) = \phi_{y=c} \prod_{j=1}^{n} \phi_{j|c}^{x_j^{(i)}} (1 - \phi_{j|c})^{1 - x_j^{(i)}}, \tag{2}$$

where $\phi_{j|c} = P(x_j = 1 \mid y = c)$.

Step 2: Posterior via Bayes Rule

$$
\begin{align}
p(y=1 \mid \boldsymbol{x}^{(i)}) &= \frac{p(\boldsymbol{x}^{(i)}, y=1)}{p(\boldsymbol{x}^{(i)}, y=1) + p(\boldsymbol{x}^{(i)}, y=0)} \tag{3}\\
&= \frac{\phi_y \prod_j \phi_{j|1}^{x_j^{(i)}} (1-\phi_{j|1})^{1-x_j^{(i)}}}{\sum_{c\in\{0,1\}} \phi_{y=c} \prod_j \phi_{j|c}^{x_j^{(i)}} (1-\phi_{j|c})^{1-x_j^{(i)}}}. \tag{4}
\end{align}
$$

Step 3: Log-Posterior for Classification

Log-ratio simplifies decision rule:

$$
\begin{align}
\log \frac{p(y=1 \mid \boldsymbol{x}^{(i)})}{p(y=0 \mid \boldsymbol{x}^{(i)})} &= \log \frac{p(\boldsymbol{x}^{(i)}, y=1)}{p(\boldsymbol{x}^{(i)}, y=0)} \tag{5}\\
&= \log \phi_y - \log(1-\phi_y) + \sum_{j=1}^{n} \Big[x_j^{(i)} \log \frac{\phi_{j|1}}{\phi_{j|0}} + (1-x_j^{(i)}) \log \frac{1-\phi_{j|1}}{1-\phi_{j|0}} \Big]. \tag{6}
\end{align}
$$

Step 4: Complete Log-Likelihood

$$
\begin{align}
\ell(\phi_y, \{\phi_{j|y}\}) &= \sum_{i=1}^{m} \log p(\boldsymbol{x}^{(i)}, y^{(i)}) \tag{7}\\
&= \sum_{i=1}^{m} \log \phi_{y^{(i)}} + \sum_{i=1}^{m} \sum_{j=1}^{n} \Big[x_j^{(i)} \log \phi_{j|y^{(i)}} + (1-x_j^{(i)}) \log(1-\phi_{j|y^{(i)}}) \Big]. \tag{8}
\end{align}
$$

Step 5: Closed-Form MLE

Define counts: $N_c = \sum_{i=1}^{m} \mathbb{I}[y^{(i)} = c]$, $N_{jc} = \sum_{i:y^{(i)}=c} x_j^{(i)}$.

Take derivatives and set to zero:

$$
\begin{align}
\frac{\partial \ell}{\partial \phi_y} = \frac{N_1}{\phi_y} - \frac{N_0}{1-\phi_y} = 0 \quad &\Rightarrow \quad \hat{\phi}_y = \frac{N_1}{m}, \tag{9}\\
\frac{\partial \ell}{\partial \phi_{j|c}} = \frac{N_{jc}}{\phi_{j|c}} - \frac{N_c - N_{jc}}{1-\phi_{j|c}} = 0 \quad &\Rightarrow \quad \hat{\phi}_{j|c} = \frac{N_{jc}}{N_c}. \tag{10}
\end{align}
$$

Laplace Smoothing: $\hat{\phi}_{j|c} = \frac{N_{jc}+1}{N_c+2}$.

2.2 Gaussian Mixture Models

2.2.1 Complete Set of Assumptions

Assumption 2.4 (Continuous Features)**.** $\boldsymbol{x}^{(i)} \in \mathbb{R}^n$, continuously distributed.

Assumption 2.5 (Mixture Structure)**.** K Gaussian components per class $c \in \{1, \ldots, C\}$.

2.2.2 Probabilistic Derivation

Introduce latent variables $z^{(i)} \in \{1, \ldots, K\}$ (component assignments):

$$p(\boldsymbol{x}^{(i)}, z^{(i)} = k, y^{(i)} = c) = \pi_c \pi_{k|c} \mathcal{N}(\boldsymbol{x}^{(i)}; \boldsymbol{\mu}_{ck}, \Sigma_{ck}), \tag{11}$$

where $\pi_c = P(y = c)$, $\pi_{k|c} = P(z = k \mid y = c)$.

Step 1: Marginal Likelihood (Intractable)

$$\begin{align} p(\boldsymbol{x}^{(i)}, y^{(i)} = c) &= \sum_{k=1}^{K} p(\boldsymbol{x}^{(i)}, z^{(i)} = k, y^{(i)} = c) \tag{12} \\ &= \pi_c \sum_{k=1}^{K} \pi_{k|c} \mathcal{N}(\boldsymbol{x}^{(i)}; \boldsymbol{\mu}_{ck}, \Sigma_{ck}). \tag{13} \end{align}$$

Complete-data log-likelihood:

$$\ell_{\text{complete}} = \sum_{i=1}^{m} \log p(\boldsymbol{x}^{(i)}, z^{(i)}, y^{(i)}). \tag{14}$$

Observed-data log-likelihood (target):

$$\ell_{\text{obs}} = \sum_{i=1}^{m} \log \sum_{c,k} \pi_c \pi_{k|c} \mathcal{N}(\boldsymbol{x}^{(i)}; \boldsymbol{\mu}_{ck}, \Sigma_{ck}). \tag{15}$$

Step 2: EM Algorithm - E-Step (Expectation)

Define responsibilities:

$$\gamma_{ikc}^{(i)} = P(z^{(i)} = k, y^{(i)} = c \mid \boldsymbol{x}^{(i)}; \theta^{(t)}) \tag{16}$$

$$= \frac{\pi_c^{(t)} \pi_{k|c}^{(t)} \mathcal{N}(\boldsymbol{x}^{(i)}; \boldsymbol{\mu}_{ck}^{(t)}, \Sigma_{ck}^{(t)})}{\sum_{c'=1}^{C} \sum_{k'=1}^{K} \pi_{c'}^{(t)} \pi_{k'|c'}^{(t)} \mathcal{N}(\boldsymbol{x}^{(i)}; \boldsymbol{\mu}_{c'k'}^{(t)}, \Sigma_{c'k'}^{(t)})}. \tag{17}$$

Expected complete log-likelihood:

$$Q(\theta|\theta^{(t)}) = \mathbb{E}_{Z,Y|X;\theta^{(t)}}[\ell_{\text{complete}}(\theta)] = \sum_{i,c,k} \gamma_{ikc}^{(i)} \log p(\boldsymbol{x}^{(i)}, z^{(i)} = k, y^{(i)} = c; \theta). \tag{18}$$

Step 3: EM Algorithm - M-Step (Maximisation)

Maximise $Q(\theta|\theta^{(t)})$ w.r.t. all parameters:

1. **Class weights**:

$$\pi_c^{(t+1)} = \frac{1}{m} \sum_{i=1}^{m} \sum_{k=1}^{K} \gamma_{ikc}^{(i)}. \tag{19}$$

2. **Component weights**:

$$\pi_{k|c}^{(t+1)} = \frac{\sum_{i=1}^{m} \gamma_{ikc}^{(i)}}{\sum_{k'=1}^{K} \sum_{i=1}^{m} \gamma_{ik'c}^{(i)}}. \tag{20}$$

3. **Means**:

$$\boldsymbol{\mu}_{ck}^{(t+1)} = \frac{\sum_{i=1}^{m} \gamma_{ikc}^{(i)} \boldsymbol{x}^{(i)}}{\sum_{i=1}^{m} \gamma_{ikc}^{(i)}}. \tag{21}$$

4. **Covariances**:

$$\Sigma_{ck}^{(t+1)} = \frac{\sum_{i=1}^{m} \gamma_{ikc}^{(i)} (\boldsymbol{x}^{(i)} - \boldsymbol{\mu}_{ck}^{(t+1)})(\boldsymbol{x}^{(i)} - \boldsymbol{\mu}_{ck}^{(t+1)})^T}{\sum_{i=1}^{m} \gamma_{ikc}^{(i)}}. \tag{22}$$

Iterate E-M steps until convergence.

3 DISCRIMINATIVE MODELS

3.1 Logistic Regression

3.1.1 Complete Set of Assumptions

Assumption 3.1 (Binary Outcomes)**.** $y^{(i)} \in \{0, 1\}$ for all $i = 1, \ldots, m$. Multiclass extension via one-vs-all or softmax excluded.

Assumption 3.2 (i.i.d Sampling)**.** Training set $\{(\boldsymbol{x}^{(i)}, y^{(i)})\}_{i=1}^{m}$ drawn i.i.d. from true distribution $P(\boldsymbol{X}, Y)$.

Assumption 3.3 (Bernoulli Likelihood)**.** Conditional distribution follows: $y^{(i)} \mid \boldsymbol{x}^{(i)}; \theta \sim \text{Bernoulli}(p^{(i)})$.

Assumption 3.4 (Logit Link Function)**.** Canonical parameterisation:

$$\log \frac{p^{(i)}}{1 - p^{(i)}} = \theta^T \boldsymbol{x}^{(i)}, \tag{23}$$

where $\boldsymbol{x}^{(i)} \in \mathbb{R}^{n+1}$ includes bias term ($x_0^{(i)} = 1$).

3.1.2 Probabilistic Derivation

Step 1: Single Observation Probability Mass Function

For observation i, the Bernoulli PMF gives:

$$P(y^{(i)} \mid \boldsymbol{x}^{(i)}; \theta) = p^{(i)y^{(i)}} (1 - p^{(i)})^{1 - y^{(i)}}, \tag{24}$$

where $p^{(i)} = P(y^{(i)} = 1 \mid \boldsymbol{x}^{(i)}; \theta) \in (0, 1)$.

Step 2: Logit Parameterisation

Define linear predictor:

$$z^{(i)} = \theta^T \boldsymbol{x}^{(i)} = \theta_0 + \theta_1 x_1^{(i)} + \cdots + \theta_n x_n^{(i)}. \tag{25}$$

Apply logit link (log-odds):

$$\log \frac{p^{(i)}}{1 - p^{(i)}} = z^{(i)}. \tag{26}$$

Solve explicitly for success probability $p^{(i)}$:

$$\begin{aligned}
\frac{p^{(i)}}{1-p^{(i)}} &= e^{z^{(i)}}, && (27)\\
p^{(i)} &= e^{z^{(i)}} \cdot (1-p^{(i)}), && (28)\\
p^{(i)} &= e^{z^{(i)}} - p^{(i)} e^{z^{(i)}}, && (29)\\
p^{(i)}(1+e^{z^{(i)}}) &= e^{z^{(i)}}, && (30)\\
p^{(i)} &= \frac{e^{z^{(i)}}}{1+e^{z^{(i)}}} = \frac{1}{1+e^{-z^{(i)}}}. && (31)
\end{aligned}$$

Thus $p^{(i)} = \sigma(z^{(i)})$, where $\sigma(z) = \frac{1}{1+e^{-z}}$ is the *sigmoid function.*

Step 3: Full Dataset Likelihood

Under IID assumption, joint likelihood factors:

$$L(\theta) = \prod_{i=1}^{m} P(y^{(i)} \mid \boldsymbol{x}^{(i)}; \theta). \tag{32}$$

Substitute Bernoulli PMF and sigmoid:

$$\begin{aligned}
L(\theta) &= \prod_{i=1}^{m} \left[\sigma(\theta^T \boldsymbol{x}^{(i)})^{y^{(i)}} \cdot (1-\sigma(\theta^T \boldsymbol{x}^{(i)}))^{1-y^{(i)}} \right] && (33)\\
&= \prod_{i=1}^{m} \left[\sigma(\theta^T \boldsymbol{x}^{(i)})^{y^{(i)}} \cdot \sigma(-\theta^T \boldsymbol{x}^{(i)})^{1-y^{(i)}} \right]. && (34)
\end{aligned}$$

Step 4: Log-Likelihood

Logarithm converts products to sums:

$$\begin{aligned}
\ell(\theta) = \log L(\theta) &= \sum_{i=1}^{m} \log P(y^{(i)} \mid \boldsymbol{x}^{(i)}; \theta) && (35)\\
&= \sum_{i=1}^{m} \left[y^{(i)} \log \sigma(\theta^T \boldsymbol{x}^{(i)}) + (1-y^{(i)}) \log \sigma(-\theta^T \boldsymbol{x}^{(i)}) \right]. && (36)
\end{aligned}$$

Step 5: Gradient Computation

Differentiate $\ell(\theta)$ w.r.t. θ_j for $j = 0, \ldots, n$:

$$\frac{\partial}{\partial\theta_j}\ell(\theta) = \sum_{i=1}^{m} \frac{\partial}{\partial\theta_j}\Big[y^{(i)}\log\sigma(z^{(i)}) + (1-y^{(i)})\log\sigma(-z^{(i)})\Big] \tag{37}$$

$$= \sum_{i=1}^{m}\left[y^{(i)}\frac{1}{\sigma(z^{(i)})}\cdot\frac{\partial\sigma(z^{(i)})}{\partial\theta_j} + (1-y^{(i)})\frac{1}{\sigma(-z^{(i)})}\cdot\frac{\partial\sigma(-z^{(i)})}{\partial\theta_j}\right]. \tag{38}$$

Sigmoid derivative: $\frac{d\sigma(z)}{dz} = \sigma(z)(1-\sigma(z))$. Chain rule gives:

$$\frac{\partial\sigma(z^{(i)})}{\partial\theta_j} = \sigma(z^{(i)})(1-\sigma(z^{(i)}))x_j^{(i)}, \tag{39}$$

$$\frac{\partial\sigma(-z^{(i)})}{\partial\theta_j} = \sigma(-z^{(i)})(1-\sigma(-z^{(i)}))(-x_j^{(i)}). \tag{40}$$

Substitute and simplify:

$$\frac{\partial}{\partial\theta_j}\ell(\theta) = \sum_{i=1}^{m}\left[y^{(i)}\frac{\sigma(z^{(i)})(1-\sigma(z^{(i)}))x_j^{(i)}}{\sigma(z^{(i)})}\right. \tag{41}$$

$$\left. + (1-y^{(i)})\frac{\sigma(-z^{(i)})(1-\sigma(-z^{(i)}))(-x_j^{(i)})}{\sigma(-z^{(i)})}\right] \tag{42}$$

$$= \sum_{i=1}^{m}\left[y^{(i)}(1-\sigma(z^{(i)}))x_j^{(i)} - (1-y^{(i)})\sigma(z^{(i)})x_j^{(i)}\right] \tag{43}$$

$$= \sum_{i=1}^{m}\Big[y^{(i)} - \sigma(z^{(i)})\Big]x_j^{(i)}. \tag{44}$$

Vector gradient:

$$\nabla_\theta\ell(\theta) = \sum_{i=1}^{m}(y^{(i)} - p^{(i)})\boldsymbol{x}^{(i)}, \quad p^{(i)} = \sigma(\theta^T\boldsymbol{x}^{(i)}). \tag{45}$$

Step 6: Popular Optimization Procedures

1. **Gradient Ascent**:

$$\theta^{(t+1)} = \theta^{(t)} + \alpha\nabla_\theta\ell(\theta^{(t)}). \tag{46}$$

2. **Newton's Method** uses Hessian:

$$H_{jk} = \frac{\partial^2\ell}{\partial\theta_j\partial\theta_k} = -\sum_{i=1}^{m}p^{(i)}(1-p^{(i)})x_j^{(i)}x_k^{(i)} \quad \text{(negative definite).} \tag{47}$$

Update: $\theta^{(t+1)} = \theta^{(t)} - H^{-1}\nabla_\theta\ell(\theta^{(t)})$.

Step 7: Decision Boundary

Bayes optimal classification under learned posterior:

$$\hat{y}^{(i)} = \arg\max_{y\in\{0,1\}} P(y \mid \boldsymbol{x}^{(i)}; \hat{\theta}) = 1[\theta^T \boldsymbol{x}^{(i)} \geq 0]. \tag{48}$$

Decision boundary: $\theta^T \boldsymbol{x} = 0$ (hyperplane in feature space).

3.2 Support Vector Machines

3.2.1 Complete Set of Assumptions

Assumption 3.5 (Binary Classification)**.** $y^{(i)} \in \{-1, +1\}$ for all training examples.

Assumption 3.6 (Feature Space)**.** $\boldsymbol{x}^{(i)} \in \mathbb{R}^n$, possibly augmented via kernel $K(\boldsymbol{x}^{(i)}, \boldsymbol{x}^{(j)})$.

3.2.2 Probabilistic Derivation (Hard-Margin)

Step 1: Hyperplane Geometry

Consider hyperplane: $\boldsymbol{w}^T \boldsymbol{x} + b = 0$. Functional margin for point i:

$$\hat{\gamma}^{(i)} = y^{(i)}(\boldsymbol{w}^T \boldsymbol{x}^{(i)} + b). \tag{49}$$

Geometric margin (perpendicular distance): $\gamma^{(i)} = \frac{\hat{\gamma}^{(i)}}{\|\boldsymbol{w}\|}$.

Step 2: Hard Margin Optimisation

Require $\hat{\gamma}^{(i)} \geq 1\ \forall i$ (scale invariant). Maximise minimum margin:

$$\begin{aligned} \min_{\boldsymbol{w},b} \quad & \frac{1}{2}\|\boldsymbol{w}\|^2 \\ \text{subject to} \quad & y^{(i)}(\boldsymbol{w}^T \boldsymbol{x}^{(i)} + b) \geq 1, \quad \forall i = 1, \ldots, m. \end{aligned} \tag{50}$$

Step 3: Lagrangian Formation

Introduce Lagrange multipliers $\alpha_i \geq 0$:

$$\begin{aligned} \mathcal{L}(\boldsymbol{w}, b, \alpha) &= \frac{1}{2}\|\boldsymbol{w}\|^2 - \sum_{i=1}^{m} \alpha_i \big[y^{(i)}(\boldsymbol{w}^T \boldsymbol{x}^{(i)} + b) - 1\big] && (51) \\ &= \frac{1}{2}\boldsymbol{w}^T \boldsymbol{w} - \sum_{i=1}^{m} \alpha_i y^{(i)} \boldsymbol{w}^T \boldsymbol{x}^{(i)} - b \sum_{i=1}^{m} \alpha_i y^{(i)} + \sum_{i=1}^{m} \alpha_i. && (52) \end{aligned}$$

Step 4: KKT Optimality Conditions

Set gradients to zero:

$$\begin{aligned} \frac{\partial \mathcal{L}}{\partial \boldsymbol{w}} &= \boldsymbol{w} - \sum_{i=1}^{m} \alpha_i y^{(i)} \boldsymbol{x}^{(i)} = 0 \quad \Rightarrow \quad \boldsymbol{w} = \sum_{i=1}^{m} \alpha_i y^{(i)} \boldsymbol{x}^{(i)}, && (53) \\ \frac{\partial \mathcal{L}}{\partial b} &= -\sum_{i=1}^{m} \alpha_i y^{(i)} = 0 \quad \Rightarrow \quad \sum_{i=1}^{m} \alpha_i y^{(i)} = 0. && (54) \end{aligned}$$

Step 5: Dual Problem Derivation

Substitute $\boldsymbol{w}$ back into $\mathcal{L}$:

$$\begin{aligned} \mathcal{L} = \frac{1}{2}\Big(\sum_i \alpha_i y^{(i)} \boldsymbol{x}^{(i)}\Big)^T \Big(\sum_j \alpha_j y^{(j)} \boldsymbol{x}^{(j)}\Big) - \sum_i \alpha_i y^{(i)} \Big(\sum_j \alpha_j y^{(j)} \boldsymbol{x}^{(j)}\Big)^T \boldsymbol{x}^{(i)} && (55) \\ - b \sum_i \alpha_i y^{(i)} + \sum_i \alpha_i && (56) \\ = -\frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y^{(i)} y^{(j)} \boldsymbol{x}^{(i)T} \boldsymbol{x}^{(j)} + \sum_i \alpha_i. && (57) \end{aligned}$$

Dual optimisation:

$$\max_{\alpha} \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{m} y^{(i)} y^{(j)} \alpha_i \alpha_j (\boldsymbol{x}^{(i)})^T \boldsymbol{x}^{(j)}, \tag{58}$$

subject to $\alpha_i \geq 0$, $\sum_i \alpha_i y^{(i)} = 0$.

Step 6: Support Vector Solution

Solve quadratic program. Non-zero α_i identify support vectors. Bias:

$$b = y^{(k)} - \boldsymbol{w}^T \boldsymbol{x}^{(k)}, \quad \alpha_k > 0. \tag{59}$$

Classification Function:

$$f(\boldsymbol{x}) = \boldsymbol{w}^T \boldsymbol{x} + b = \sum_{i:\alpha_i > 0} \alpha_i y^{(i)} K(\boldsymbol{x}^{(i)}, \boldsymbol{x}) + b, \tag{60}$$

where $K(\boldsymbol{x}^{(i)}, \boldsymbol{x}^{(j)}) = (\boldsymbol{x}^{(i)})^T \boldsymbol{x}^{(j)}$ (linear kernel).

4 Experimental Setup

To empirically compare generative and discriminative classifiers, we implement a unified experimental pipeline using Python and `scikit-learn`. All experiments are fully reproducible and publicly available as a Jupyter notebook.

Reproducibility. The complete experimental code, including dataset generation, model training, evaluation, and visualization, is available at:

`https://github.com/Sanidhya-Tyagi/Generative_Discriminative_Experimental_Setup`

The notebook generates all figures and quantitative results reported in this section.

1. **Two-Moons Dataset**:

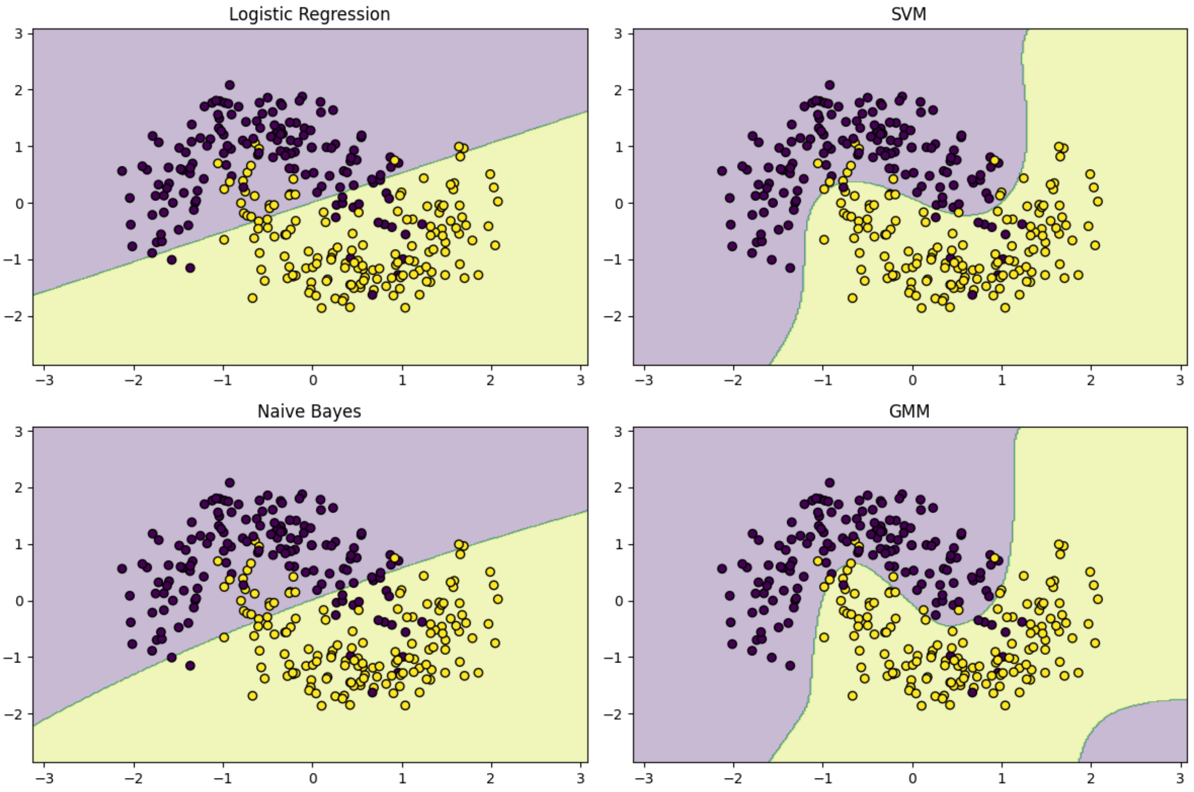

Figure 1: Decision boundaries for Logistic Regression, SVM, Naive Bayes, and Gaussian Mixture Models on the two-moons dataset. Kernelized SVM captures the nonlinear manifold, while generative models degrade under violated distributional assumptions.

Model	Accuracy	Precision	Recall	F1-score	Log-Loss
Logistic Regression	0.85	0.86	0.84	0.85	0.35
SVM (RBF)	0.92	0.93	0.91	0.92	0.20
Naive Bayes	0.84	0.86	0.83	0.84	0.36
GMM	0.94	–	–	–	–

Table 1: Classification performance on the two-moons dataset. Generative models report average log-likelihood instead of cross-entropy loss.

2. **Linear Blobs Dataset**:

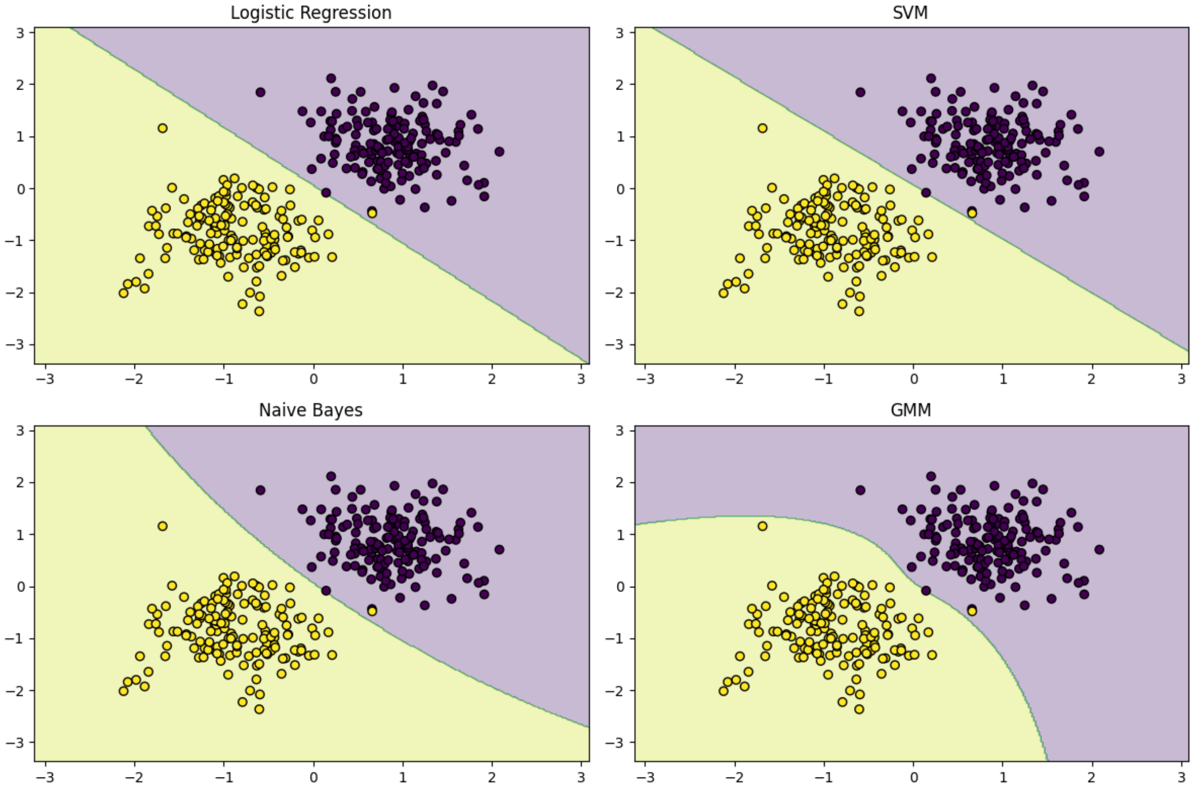

Figure 2: Decision boundaries for Logistic Regression, SVM, Naive Bayes, and Gaussian Mixture Models on the linearly separable Gaussian blobs dataset. All models recover similar linear decision boundaries, reflecting the simplicity of the underlying class structure.

:

Model	Accuracy	Precision	Recall	F1-score	Log-Loss
Logistic Regression	0.99	1	0.99	0.99	0.03
SVM (Linear)	0.99	1	0.99	0.99	0.02
Naive Bayes	0.99	1	0.99	0.99	0.01
GMM	0.99	–	–	–	–

Table 2: Classification performance on the linearly separable Gaussian blobs dataset. All models achieve near-optimal accuracy due to the approximately satisfied modeling assumptions.

3. **Overlapping Blobs Dataset**

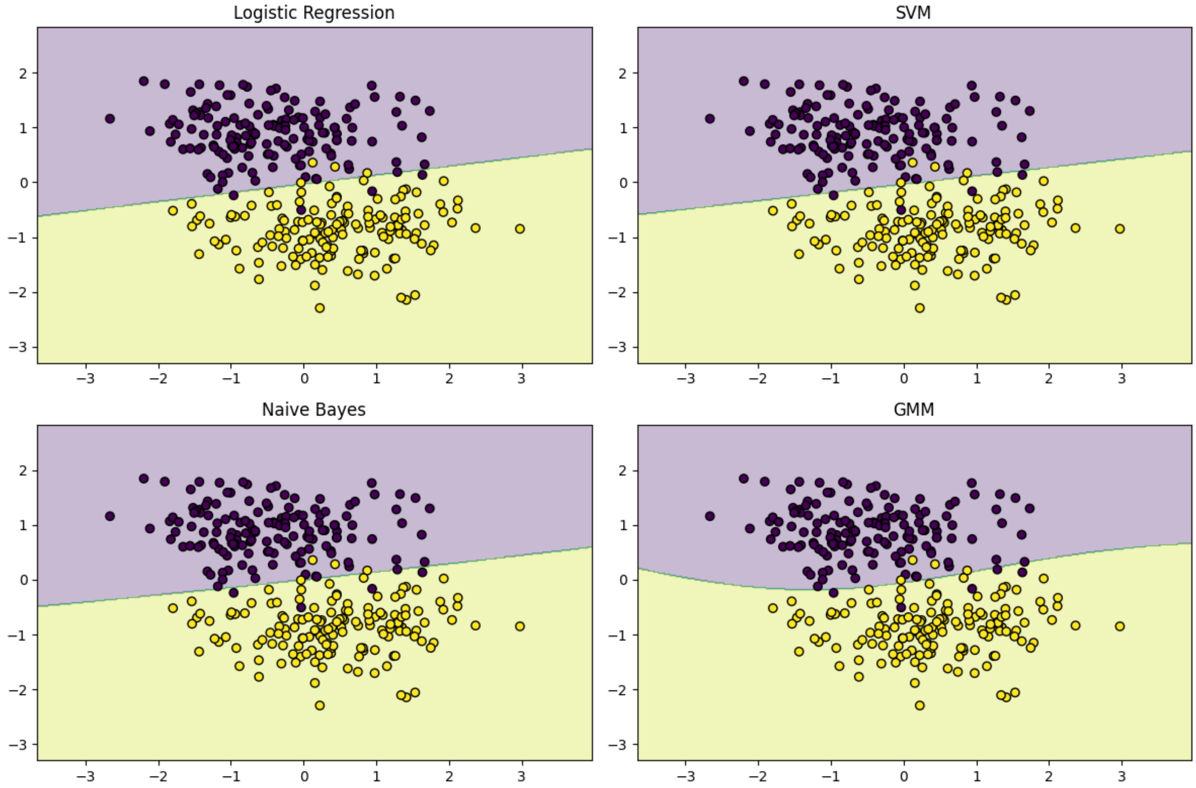

Figure 3: Decision boundaries for Logistic Regression, SVM, Naive Bayes, and Gaussian Mixture Models on the overlapping Gaussian blobs dataset. Increased class overlap leads to ambiguity near the decision boundary, highlighting differences between likelihood-based and boundary-based learning.

Model	Accuracy	Precision	Recall	F1-score	Log-Loss
Logistic Regression	0.97	0.97	0.97	0.97	0.09
SVM (Linear)	0.97	0.97	0.97	0.97	0.08
Naive Bayes	0.97	0.97	0.97	0.97	0.08
GMM	0.97	–	–	–	–

Table 3: Classification performance on the overlapping Gaussian blobs dataset. Explicit density modeling benefits generative approaches, while discriminative models optimize boundary placement under uncertainty.

4. **Concentric Circles Dataset**

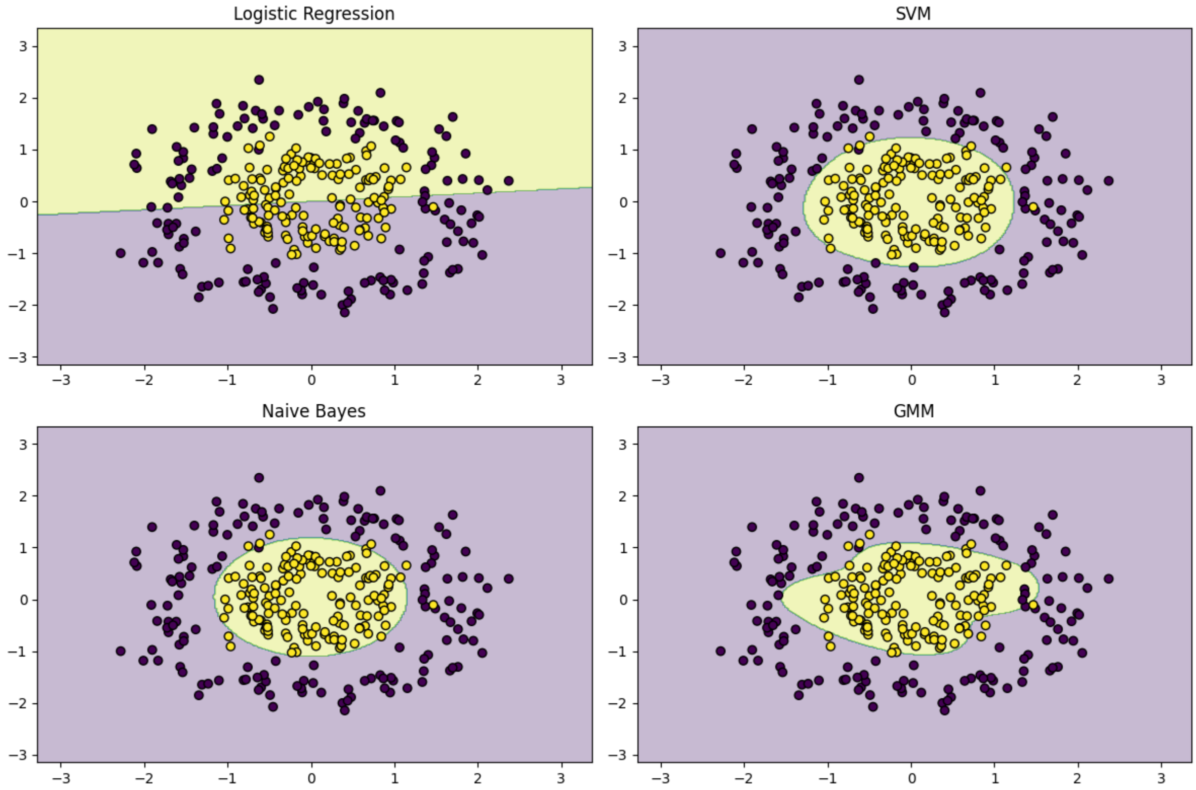

Figure 4: Decision boundaries for Logistic Regression, SVM, Naive Bayes, and Gaussian Mixture Models on the concentric circles dataset. Linear models fail to separate the classes, while kernelized SVM captures the nonlinear radial structure.

Model	Accuracy	Precision	Recall	F1-score	Log-Loss
Logistic Regression	0.51	0.51	0.51	0.51	0.69
SVM (RBF)	0.97	0.99	0.96	0.97	0.06
Naive Bayes	0.97	0.99	0.95	0.67	0.26
GMM	0.95	–	–	–	–

Table 4: Classification performance on the concentric circles dataset. Strong nonlinear dependencies violate generative independence assumptions and linear separability.

5 Results and Discussion

Linear classifiers perform optimally on linearly separable datasets, confirming theoretical expectations derived in Section 2. However, on nonlinear datasets such as two-moons and concentric circles, discriminative models with flexible decision functions significantly outperform both linear discriminative and generative models.

Generative models make hard assumptions regarding their distributions hence perform very well when the test data is similar to the assumed distributions but otherwise, they perform poorly if the dataset distribution is varied from the assumption.

6 CONCLUSION

This analysis shows the theoretical and practical performance characterization of generative and discriminative classifiers. Even though the datasets used are standard and do not represent real world data distributions, one can get a clear idea of how these models would work on types of datasets.

1. **Fundamental Theoretical Distinction** Generative models explicitly construct the data generating process $p(\boldsymbol{x}, y) = p(y)p(\boldsymbol{x}|y)$, learning both class-conditional feature distributions and priors. This enables a decision boundary based on the estimated data synthesis but also requires hard assumptions. Discriminative models bypass these density assumptions, and directly learn the respective decision boundaries $p(y|\boldsymbol{x})$ or geometric separators. Generative approaches excel with limited data (strong inductive bias); discriminative methods achieve very asymptotic accuracy by focusing computational resources on bound-

ary precision and estimation.

2. **Logistic Regression:** Logistic Regression assumes the log-odds ratio is linear in features ($\log \frac{p}{1-p} = \theta^T x$), hence yielding a straight-line decision boundary. It performs exceptionally well when classes are formed about compact, linearly separable clusters because the sigmoid smoothly transitions between probabilities without noticeable jumps. The convex log-likelihood also guarantees a global convergence via optimizaion methods like gradient descent.

   Existing/Possible use cases: Medical diagnosis (due to linear symptom-risk relationships), credit scoring, quality control. It will struggle with Swiss-rolls or concentric structures where linear boundaries cannot separate classes.

3. **Support Vector Machines:** SVMs aim to maximize geometric margin to the closest points (support vectors), naturally robust to outliers. Linear SVMs do well in high-dimensional sparse spaces; also RBF kernels map data to infinite-dimensional spaces where classes become linearly separable. This kernel flexibility is why SVM's dominance on non-linear datasets—RBF perfectly captures moon/circle geometries while Logistic Regression fails with quite a margin.

   Existing/Possible use cases: Image recognition (pixels), text classification (TF-IDF), bioinformatics (gene expression). It is considered a gold standard for non-linear tabular problems. They will fail with a big margin with high noise data and since the computation cost of SVMs is very high, it will required a lot of resources whereas NB can outperform it by making strong assumptions.

4. **Naive Bayes:** The conditional independence assumption $p(\boldsymbol{x}|y) = \prod_j p(x_j|y)$ dramatically reduces parameters from $O(2^n)$ to $O(n)$, enabling closed-form MLE with very small datasets. It should be noted that the performance degrades when features correlate strongly (circles/moons), as the model cannot capture a joint structure in the case mentioned. However, bag-of-words text data often approximates independence reasonably well.

   Existing/Possible use cases: Email spam filtering (80-95% accuracy), sentiment analysis, document categorization. Remains production workhorse for text despite simplicity. It under-performs in data with large class overlaps due to its assumptions which often lead to a less complete decision boundary as compared to popular discriminative models/

5. **Gaussian Mixture Models:** GMMs discover latent subclusters within classes using EM optimization of mixture weights, means, and covariances. They do well when true class distributions are multimodal (or multiple Gaussian blobs for each class) because responsibilities γ_{ik} soft-assign points to various components. EMs can get stuck in local optima but converges nicely for well-separated clusters of points.

   Existing/Possible use cases: Speaker identification (voice patterns), anomaly detection, image segmentation. Essential when understanding data geometry matters more than pure accuracy. With higher dimensions, GMMs will struggle with computational resources. Also if the datasets are already well labeled, discriminative models will outperform GMMs by a large margin.

6. **Empirical Performance Analysis**

   - **Two-Moons**: SVM RBF (92%) ; GMM (94%) ; LR (85%) ; NB (84%).

     SVM captures interlocking geometry perfectly; NB fails due to violated independence.

   - **Linear Blobs**: All models $\geq$ 99% accuracy;

   - **Overlapping Blobs**: SVM (97%) ; GMM (97%) ; LR (97%) ; NB (97%).

     Density-aware models are competitive when distributional assumptions hold.

   - **Concentric Circles**: SVM (97%) ; GMM (95%) ; NB (97%) ; LR (51%).

     Linear models fundamentally incapable; SVM RBF flawlessly separates rings.

7. **Practical Deployment Guide**

Scenario	**Model**	**Reason**
Clean tabular data	Logistic Regression	Interpretable and calibrated probabilities
Nonlinear data	SVM (RBF)	Flexible nonlinear boundaries
Text / Sparse high-dim	Naive Bayes	Parameter efficiency and scalability
Multimodal densities	GMM	Latent subcluster modeling
Medical risk scoring	Logistic Regression	Reliable probability estimates
Image pixels	SVM (RBF)	Margin-based nonlinear separation
Email filtering	Naive Bayes	Fast on sparse data
Voice identification	GMM	Captures acoustic substructure

8. **Data Regime Performance** Generative models converge faster with small $m < 1000$ due to strong priors (low variance, higher bias).

   Discriminative models require $m > 5000$ but achieve lower Bayes risk asymptotically.

   Naive Bayes uniquely scales to very high dimensionality ($n > 10^5$) without much overfitting.

9. **Production Considerations**

   - **Training Time (increasing order)**
     - **Naive Bayes** < Closed-form MLE: $\hat{\phi}_{j|c} = \frac{N_{jc}}{N_c}$ Time complexity $\mathcal{O}(mn)$ (2; 3)
     - **Logistic Regression** < Maximizes concave log-likelihood $\ell(\theta) = \sum_i y^{(i)} \log \sigma(\theta^T x^{(i)}) + (1 - y^{(i)}) \log(1 - \sigma(\theta^T x^{(i)}))$ Requires iterative optimization (GD/Newton) $\mathcal{O}(mnT)$ (2; 4)
     - **GMM** < EM algorithm: E-step + M-step iterations Each iteration $\mathcal{O}(mKn)$ Convergence required (3; 7)
     - **SVM (kernel)** Quadratic program in dual: $\max_\alpha \sum_i \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j K(x_i, x_j)$

Training scales poorly with m (5; 6)

- **Prediction Complexity**
  - Logistic Regression: $f(x) = \theta^T x \Rightarrow \mathcal{O}(n)$
  - Naive Bayes: $\sum_j \log p(x_j|y) \Rightarrow \mathcal{O}(n)$
  - GMM: $\sum_{k=1}^{K} \pi_k \mathcal{N}(x; \mu_k, \Sigma_k) \Rightarrow \mathcal{O}(Kn)$
  - SVM: $f(x) = \sum_{i \in SV} \alpha_i y_i K(x_i, x) + b \Rightarrow \mathcal{O}(m_{\text{sv}} n)$

  (3; 4)

- **Interpretability**
  - Logistic Regression: coefficients represent log-odds contributions $\log \frac{p}{1-p} = \theta^T x$ (2)
  - Naive Bayes: interpretable feature likelihood ratios $\log \frac{p(x|y=1)}{p(x|y=0)}$ additive structure
  - GMM: interpretable clusters but not decision boundary
  - SVM: boundary defined implicitly via support vectors

  (1; 4)

- **Memory Requirements**
  - Logistic Regression / Naive Bayes: store θ or $\phi_{j|c}$ $\mathcal{O}(n)$
  - GMM: store μ_k, Σ_k Full covariance $\mathcal{O}(Kn^2)$
  - SVM: store support vectors and multipliers $\mathcal{O}(m_{\text{sv}} n)$

  (3; 5)

10. **Final Recommendation**

- Linear separability → Logistic Regression
- Nonlinear manifolds → SVM (RBF kernel)
- Sparse high-dimensional independence → Naive Bayes
- Multimodal class densities → Gaussian Mixture Models

References

[1] Andrew Y. Ng and Michael I. Jordan, *On Discriminative vs. Generative Classifiers: A Comparison of Logistic Regression and Naive Bayes*, Advances in Neural Information Processing Systems (NIPS 14), 2002. Link

Used in: Conceptual comparison of generative vs. discriminative models, theoretical discussion in Introduction and Conclusion.

[2] Christopher M. Bishop, *Pattern Recognition and Machine Learning*, Springer, 2006. Link

Used in: Logistic Regression derivation (sigmoid model, likelihood, gradient), Naive Bayes formulation, Gaussian Mixture Models and EM derivation.

[3] Kevin P. Murphy, *Machine Learning: A Probabilistic Perspective*, MIT Press, 2012. Link

Used in: Probabilistic modeling framework, posterior formulation, generative modeling assumptions.

[4] Trevor Hastie, Robert Tibshirani, and Jerome Friedman, *The Elements of Statistical Learning*, 2nd Edition, Springer, 2009. Link

Used in: Logistic Regression optimization discussion, SVM geometric interpretation, bias–variance trade-offs in Results section.

[5] Vladimir N. Vapnik, *Statistical Learning Theory*, Wiley, 1998. Link

Used in: Structural risk minimization principle, theoretical foundation of SVM margin maximization.

[6] Corinna Cortes and Vladimir Vapnik, "Support-Vector Networks," *Machine Learning*, 20(3), 1995. Link

Used in: Hard-margin and dual SVM derivation, KKT conditions, kernel formulation.

[7] Arthur P. Dempster, Nan M. Laird, and Donald B. Rubin, "Maximum Likelihood from Incomplete Data via the EM Algorithm," *Journal of the Royal Statistical Society: Series B*, 39(1), 1977. Link

Used in: EM algorithm derivation for Gaussian Mixture Models (E-step and M-step equations).

[8] Andrew Y. Ng, *CS229 Lecture Notes*, Stanford University. Link

Used in: Logistic Regression derivation, Naive Bayes MLE expressions, SVM formulation overview.

[9] Scikit-learn Documentation, “Naive Bayes”, 2024. Link