Generative Classifiers

LDA, QDA, Naïve Bayes

EN5422/EV4238 | Fall 2023

w07_density.pdf

(Week 7 – 1/1)

Contents

1 Classification and Pattern Recognition

- The outcome variable is categorical and denoted $G \in \mathcal{G}$
  - Default Credit Card Example: $\mathcal{G} = \{\text{"Yes"}, \text{"No"}\}$
  - Medical Diagnosis Example: $\mathcal{G} = \{\text{"stroke"}, \text{"heart attack"}, \text{"drug overdose"}\}$
  - Climate Extreme Example: $\mathcal{G} = \{\text{extremely dry"}, \text{"dry"}, \text{"wet"}, \text{"extremely wet"}\}$
- The training data is $D = \{(X_1, G_1), (X_2, G_2), \dots, (X_n, G_n)\}$
- The optimal decision/classification is often based on the posterior probability Pr $(G = g|\mathbf{X} = \text{x})$

1.1 Binary Classification

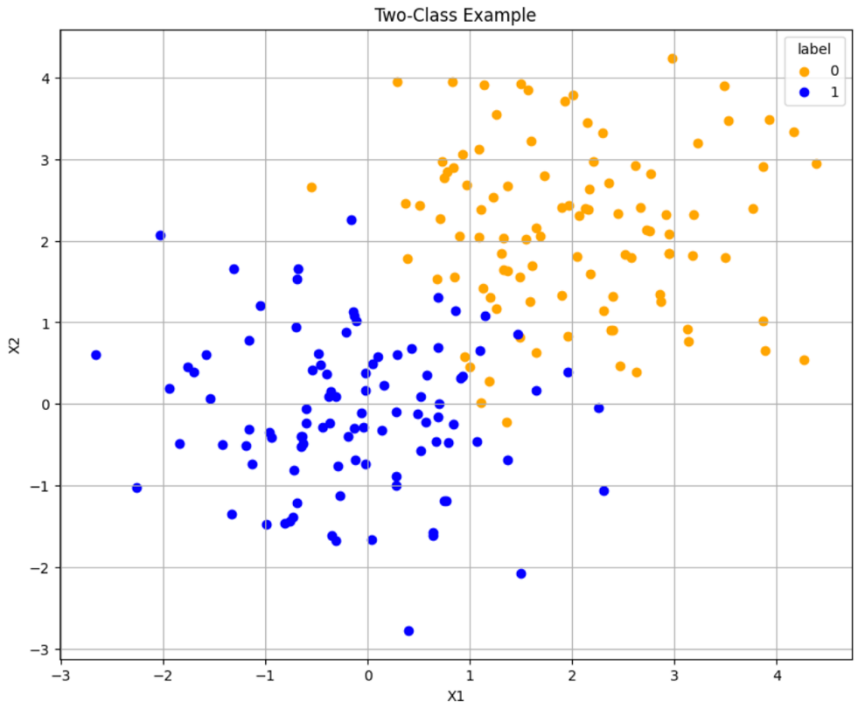
- Classification is simplified when there are only 2 classes.
  - Many multi-class problems can be addressed by solving a set of binary classification problems (e.g., one-vs-rest)
- It is often convenient to transform the outcome variable to a binary {0,1} variable:

$$Y_i = \begin{cases} 1 & G_i = \mathcal{G}_1 \text{ (outcome of interest)} \\ 0 & G_i = \mathcal{G}_1 \end{cases}$$

- Or, like with SVM, as a {-1, +1} variable:

$$Y_i = \begin{cases} +1 & G_i = \mathcal{G}_1 \text{ (outcome of interest)} \\ -1 & G_i = \mathcal{G}_1 \end{cases}$$

1.2 Two-Class Example



Your Turn #1
I simulated these data. How do you think I did it?

1.3 Conditional/Discriminative Models

- The classification models we have covered in this course so far (Linear regression, Logistic Regression and KNN) attempt to conditionally estimate a score related to the $\Pr(Y = 1|X = x)$ **conditional on** $X = x$. These models are considered *discriminative* models.

- Their goal is to directly estimate $\Pr(Y = 1 \mid X = x)$ **conditional on** $X = x$**.**

$$p(x) = \Pr(Y = 1|X = x)$$

a. Linear Regression (for binary outcomes)

$$\hat{p}(x) = \hat{\beta}^{\mathrm{T}} x$$

b. Logistic Regression

$$\log\left(\frac{\hat{p}(x)}{1-\hat{p}(x)}\right) = \hat{\beta}^{\intercal}x$$

and thus,

$$\hat{p}(x) = \frac{e^{\hat{\beta}^{\intercal}x}}{1+e^{\hat{\beta}^{\intercal}x}} = \left(1+e^{\hat{\beta}^{\intercal}x}\right)^{-1}$$

c. kNN (for binary outcomes)

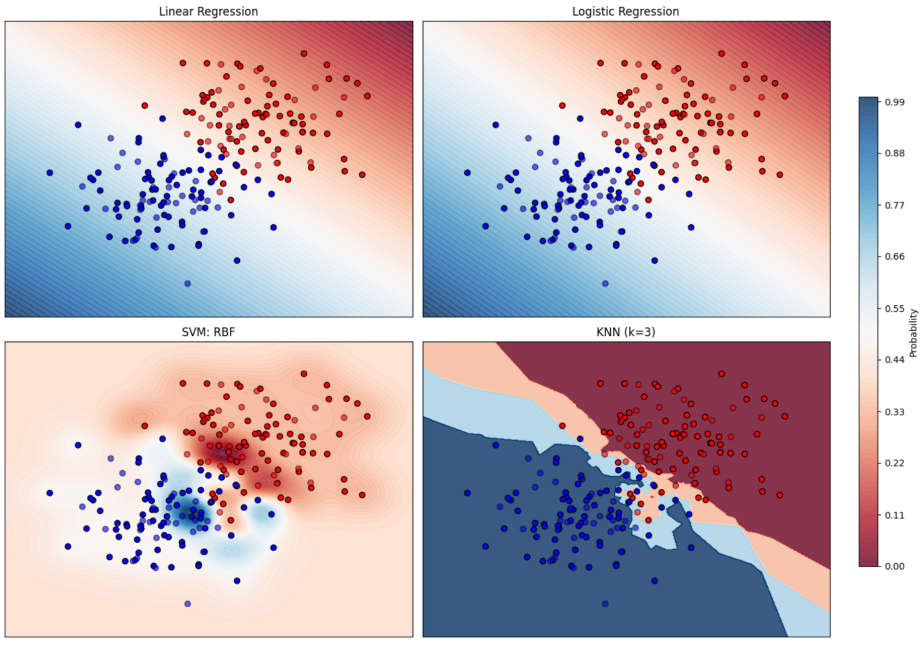
$$\hat{p}(x;k) = \frac{1}{k}\sum_{i:x_i \in N_k(x)} y_i = Avg(y_i | x_i \in N_k(x))$$

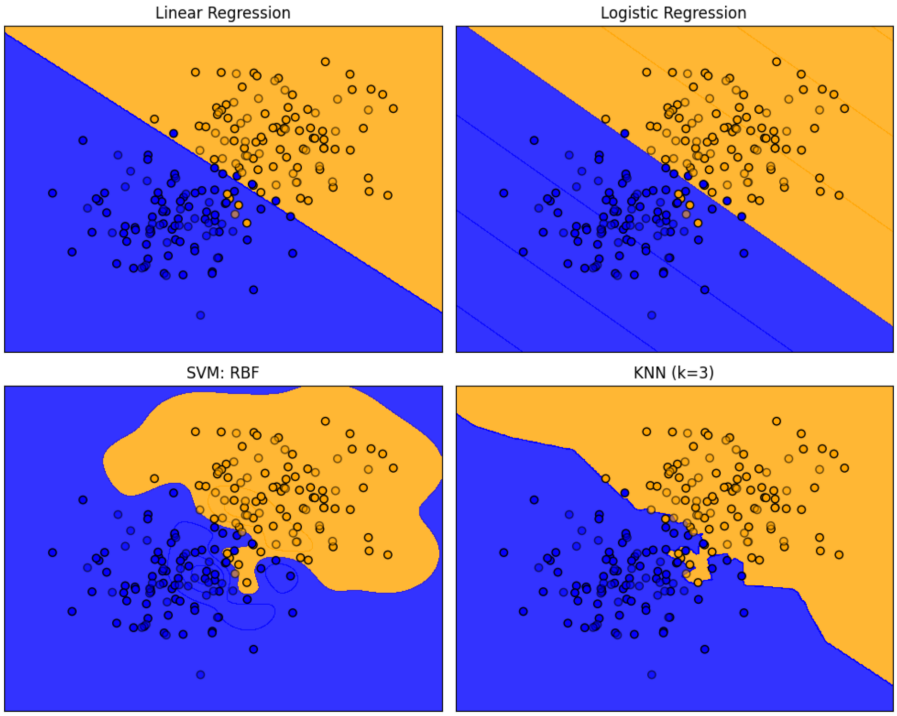
- $N_k(x)$ are the set of k closest training points to x

d. Support Vector Machines (SVM)

$$\hat{g}(x) = \hat{\beta}_0 + \sum_{i=1}^{n} \hat{\alpha}\, y_i K(x, x_i)$$

- Decide $\hat{Y} = 1$ if $\hat{g}(x) > 0$
- Or calibrated probability: $\log\left(\frac{\hat{p}(x)}{1-\hat{p}(x)}\right) = \hat{\alpha}_0 + \hat{\alpha}_1 \hat{g}(x)$
  - i.e., using logistic regression with $\hat{g}(x)$ as the predictor





2 Generative Classification Models

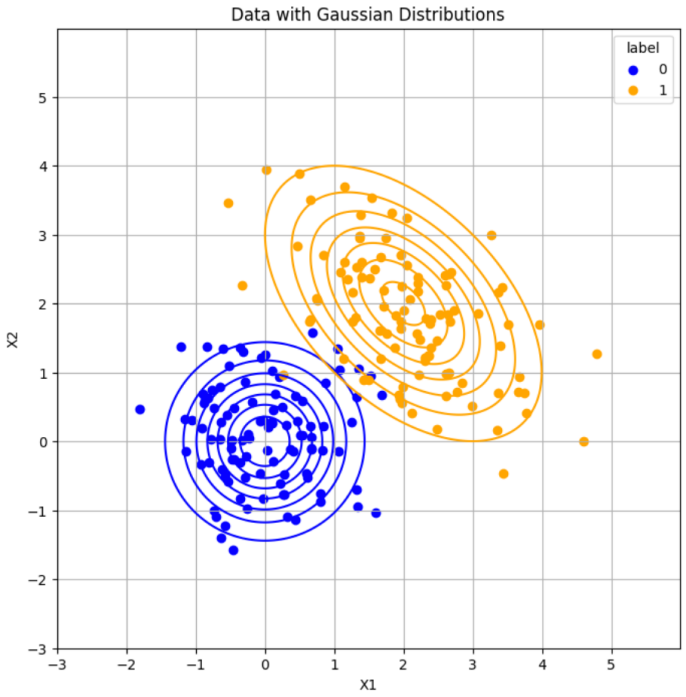
Consider how the data $D = \{(X_1, G_1), (X_2, G_2), \dots, (X_n, G_n)\}$ could be generated.

1. First, the class label is selected according to the prior probability $\pi = [\pi_1, \dots, \pi_K]$.
   a. That is, $\Pr(G_i = k) = \pi_k$
2. Given the class is k, the X value is generated $X \mid G = k \sim f_k$
   a. Let $f_k(x)$ be the (pdf/pmf/mixed) of the predictors from class k.

Example

- Two classes, $k \in \{0, 1\}$
  - $\pi_1 = 0.6, \pi_1 = 0.4$
  - I expect 60% of the observations to be from class 1.

- If $G_i = 1$, then $X \sim N\left(\mu_1 = \begin{bmatrix}2\\2\end{bmatrix}, \Sigma_1 = \begin{bmatrix}1 & -0.5\\-0.5 & 1\end{bmatrix}\right)$
- If $G_i = 0$, then $X \sim N\left(\mu_0 = \begin{bmatrix}0\\0\end{bmatrix}, \Sigma_1 = \begin{bmatrix}0.5 & 0\\0 & 0.5\end{bmatrix}\right)$



Your Turn #2

Use Bayes Theorem to re-write the expression for $\Pr(Y = 1 \mid X = x)$

$$\Pr(Y = 1 \mid X = x) = \frac{\Pr(X = x|Y = 1)\Pr(Y = 1)}{\Pr(X = x)}$$
$$= \frac{\Pr(X = x|Y = 1)\Pr(Y = 1)}{\Pr(X = x|Y = 1)\Pr(Y = 1) + \Pr(X = x|Y = 0)\Pr(Y = 0)}$$

- The prior probability $\Pr(Y = 1)$ is 0.5, since we generated an equal number of samples for each class.
- The likelihood $\Pr(X = x|Y = 1)$ using the Gaussian probability density function for class 1 at [2, 2] is approximately 0.1838.
- Therefore, the joint probability $\Pr(X = x|Y = 1)\Pr(Y = 1) = 0.0919$.

2.1 From Discriminative to Generative, and Back Again

- The models we have discussed in this course so far are considered *discriminative* and focused on estimating the **conditional** probability $\Pr(Y = k \mid X = x)$.
  - Or in case of SVM, a score representing the distance to the separating boundary.

- But there is another class of models termed *generative* which try to directly estimate the **joint** probability $\Pr(Y = k, X = x) = \Pr(X = x|Y = k)\Pr(Y = k)$
  - This flips the script; instead of using supervised models to estimate $\Pr(Y = k \mid X = x)$, we use unsupervised density estimation to estimate $\Pr(X = x|Y = k)$.

2.1.1 The Bayes Breakdown (Binary Classification)

Bayes Theorem

$$p_k(x) = \Pr(Y = k \mid X = x) = \frac{\Pr(X = x|Y = k)\Pr(Y = k)}{\Pr(X = x)} = \frac{f_k(x)\pi_k}{\sum_j f_j(x)\pi_j}$$

- $f_k(x)$ is the *class conditional density* (pdf/pmf)
- $0 \leq \pi_k \leq 1$ are the *prior class probabilities*
- $\sum \pi_k = 1$
- X is distributed as a finite mixture model: $f(x) = \sum_j f_j(x)\pi_j$

2.1.1.1 Special case when $K = 2$ (binary classification)

$$p(x) = \Pr(Y = 1 \mid X = x) = \frac{\Pr(X = x|Y = 1)\Pr(Y = 1)}{\Pr(X = x)} = \frac{f_1(x)\pi}{f_1(x)\pi + f_0(x)(1 - \pi)}$$

Recall our notation for the log-odds:

- $\gamma(x) = \log\frac{p(x)}{1-p(x)}$

The log-odds reduces to a combination of prior odds and density ratios

$$\gamma(x) = \log\left(\frac{p(x)}{1 - p(x)}\right) = \log\left(\frac{f_1(x)\pi}{f_0(x)(1 - \pi)}\right) = \log\left(\frac{\pi}{1 - \pi}\right) + \log\left(\frac{f_1(x)}{f_0(x)}\right)$$

$$\underbrace{\log\left(\frac{\pi}{1 - \pi}\right)}_{\text{log prior odds}} + \underbrace{\log\left(\frac{f_1(x)}{f_0(x)}\right)}_{\text{log density ratio}}$$

2.1.2 The Bayes Breakdown (Binary Classification)

- We know that the optimal decision can be based on the density ratios

Choose $\hat{G}(x) = 1$ if:

$$\hat{\gamma}(x) > \log\left(\frac{C_{FP}}{C_{FN}}\right)$$

$$\log\left(\frac{1-\hat{\pi}}{\hat{\pi}}\right) + \log\left(\frac{\widehat{f_1(x)}}{f_0(x)}\right) > \log\left(\frac{C_{FP}}{C_{FN}}\right)$$

$$\log\left(\frac{\widehat{f_1(x)}}{f_0(x)}\right) > \log\left(\frac{\hat{\pi}}{1-\hat{\pi}}\right) + \log\left(\frac{C_{FP}}{C_{FN}}\right)$$

- $\hat{\pi}$ typically represents the probability of the positive class (i.e., $\Pr(Y = 1)$).
- $f_1(x)$ is the probability density function of X given that Y=1.
- $f_0(x)$ is the probability density function of X given that Y=0.
- C_{FP} is the cost of a false positive.
- C_{FN} is the cost of a false negative.

2.1.3 Estimation

- $\hat{\pi} = \frac{n_k}{n}$ is a natural estimate for the class priors if we think the testing data will have the same proportions as the training data.
- The other term to estimate is the log density ratio: $\log\left(\frac{\widehat{f_1(x)}}{f_0(x)}\right)$
- Generative Models estimate this term by

$$\log\left(\frac{\widehat{f_1(x)}}{f_0(x)}\right) = \log\left(\frac{\hat{f}_1(x)}{\hat{f}_0(x)}\right)$$

- That is, generative models estimate the class conditional densities $\{f_k(\cdot)\}$.
- The different generative models take different approaches to estimate these component densities.

Generative Models
Generative Classification Models use *density estimation* to make predictions!

2.1.3.1 Linear/Quadratic Discriminant Analysis (LDA/QDA)

- Both LDA and QDA model the class conditional densities $f_k(x)$ with a *Gaussian* density
  - Thus, they model the observations as coming from a *Gaussian mixture model*
  - Each class has its own mean vector u_k
  - The difference between LDA and QDA is what they use for their covariance matrix
- **LDA**

$$f_k(x) = (2\pi)^{-\frac{p}{2}}|\Sigma|^{-\frac{1}{2}} \exp\left\{-\frac{1}{2}(\mathbf{x}-\mu_k)^{\intercal}\Sigma^{-1}(\mathbf{x}-\mu_k)\right\}$$

  - $\Sigma_k = \Sigma \quad \forall k$ *(uses the same variance-covariance for all classes)*

- **QDA**

$$f_k(x) = (2\pi)^{-\frac{p}{2}}|\Sigma_k|^{-\frac{1}{2}} \exp\left\{-\frac{1}{2}(\mathbf{x}-\mu_k)^{\intercal}{\Sigma_k}^{-1}(\mathbf{x}-\mu_k)\right\}$$

  - Σ_k is *different* for each classes

2.1.3.2 Kernel Discriminant Analysis (KDA)

- Model the class conditional densities $f_k(x)$ with a multivariate *kernel density estimate (KDE)*

$$\hat{f}_k(x) = \frac{1}{n_k}\sum_{i:g_i=k} K(x - x_i; H)$$

where H is the $p \times p$ bandwidth matrix.

2.1.3.3 Naïve Bayes

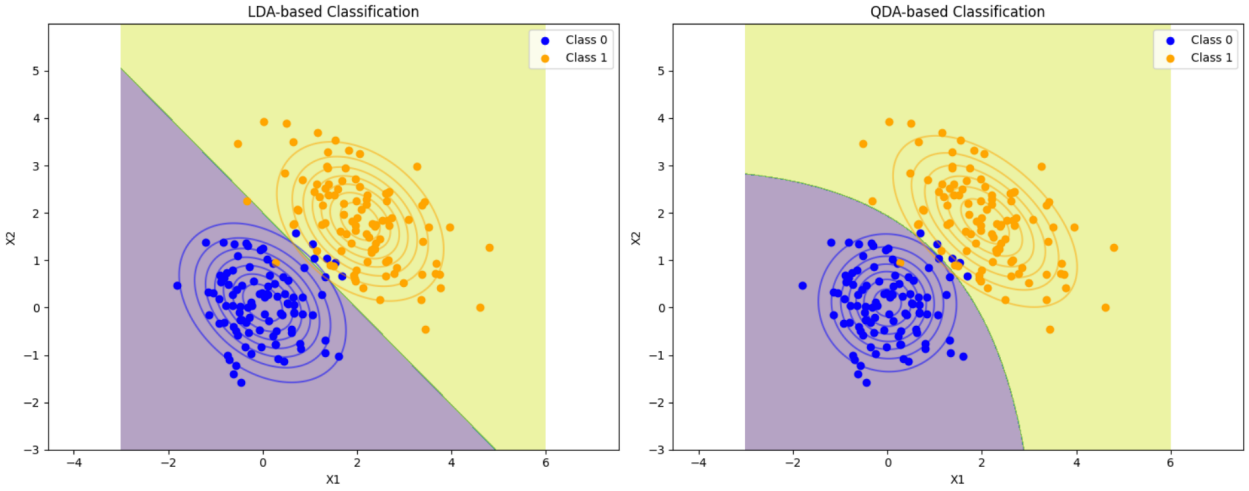
- Naïve Bayes ignores potential associations between predictors and estimates the density of each predictor variable independently.

$$\hat{f}_k(x) = \prod_{j=1}^{p} \hat{f}_{jk}(x_j)$$

  - This greatly simplifies the estimation
  - You will often find $\hat{f}_{jk}(\mu) = \mathcal{N}(\mu;\, \hat{\mu}_{jk}, \hat{\sigma}_{jk})$
  - But KDE is a great approach $\hat{f}_{jk}(\mu) = \frac{1}{n_k}\sum_{i:G_i=k} K_h(\mu - x_{ij})$
  - And including mix continuous and discrete variable is very easily
  -

Your Turn #3
How would you estimate the probability for a categorical predictor?

3 Linear/Quadratic Discriminant Analysis (LDA/QDA)



- Linear Discriminant Analysis (LDA) finds *linear* boundaries between classes
- Quadratic Discriminant Analysis (QDA) finds *quadratic* boundaries between classes
- Setup: $K = |\mathcal{G}|$ classes in the training data, $D = \{(\mathbf{X}_i, G_i)\}_{i=1}^{n}$
  - where $\mathbf{X}_i \in \mathbf{R}^p, G_i \in \mathcal{G}$
- The posterior probability of class g, given $X = x$,

$$\Pr(G = g \mid \mathbf{X} = \mathbf{x}) = \frac{f(x|G = g)\Pr(G = g)}{f(x)} = \frac{f_g(x)\pi_g}{\sum_{k=1}^{K} f_k(x)\pi_k}$$

  - $f_k(x)$ is the *class conditional density*
  - $0 \leq \pi_k \leq 1$ are the *prior class probabilities;* $\sum_{k=1}^{K} \pi_k = 1$

3.1 Estimation

- Both LDA and QDA model the class conditional density $f_k(x)$ with *Gaussians*
  - Thus, they model the observations as coming from a K component *Gaussian mixture model*
  - Each class ahs its own mean vector μ_k
  - The difference between LDA and QDA is what they use for their covariance matrix

$$f_k(x) = \mathcal{N}(x; \mu_k, \Sigma_k)$$

- LDA: $\hat{\Sigma}_1 = \hat{\Sigma}_2 = \cdots = \hat{\Sigma}_K = \hat{\Sigma}$ Common covariance
- QDA: $\hat{\Sigma}_1 \neq \hat{\Sigma}_2 \neq \cdots \neq \hat{\Sigma}_K = \hat{\Sigma}$ Different covariances

- **LDA**

$$f_k(x) = (2\pi)^{-\frac{p}{2}}|\Sigma|^{-\frac{1}{2}} \exp\left\{-\frac{1}{2}(\mathbf{x}-\mu_k)^{\intercal}\Sigma^{-1}(\mathbf{x}-\mu_k)\right\}$$

  - $\Sigma_k = \Sigma \quad \forall k$ *(uses the same variance-covariance for all classes)*

- **QDA**

$$f_k(x) = (2\pi)^{-\frac{p}{2}}|\Sigma_k|^{-\frac{1}{2}} \exp\left\{-\frac{1}{2}(\mathbf{x}-\mu_k)^{\intercal}\Sigma_k{}^{-1}(\mathbf{x}-\mu_k)\right\}$$

  - Σ_k is *different* for each classes

Your Turn #4

The LDA model uses a common covariance matrix while QDA allow each class to have a different covariance (which permits quadratic boundaries). But this flexibility comes at a cost.

1. **How may parameters have to be estimated in an LDA model with K classes and p dimensions?**

   Means: For each of the K classes, we estimate a mean vector in p dimensions. So, there are K×p mean parameters.

   Covariance Matrix: LDA assumes a shared covariance matrix across all classes. A covariance matrix for p dimensions is a $p \times p$ matrix. However, since it is symmetric (i.e., the covariance between dimension i and j is the same as between j and i), we only need to estimate the elements in the upper triangle and the diagonal. The number of elements in the upper triangle and the diagonal of a $p \times p$ matrix is given by $\frac{p(p+1)}{2}$.

   Therefore, the total number of parameters to be estimated in an LDA model is:

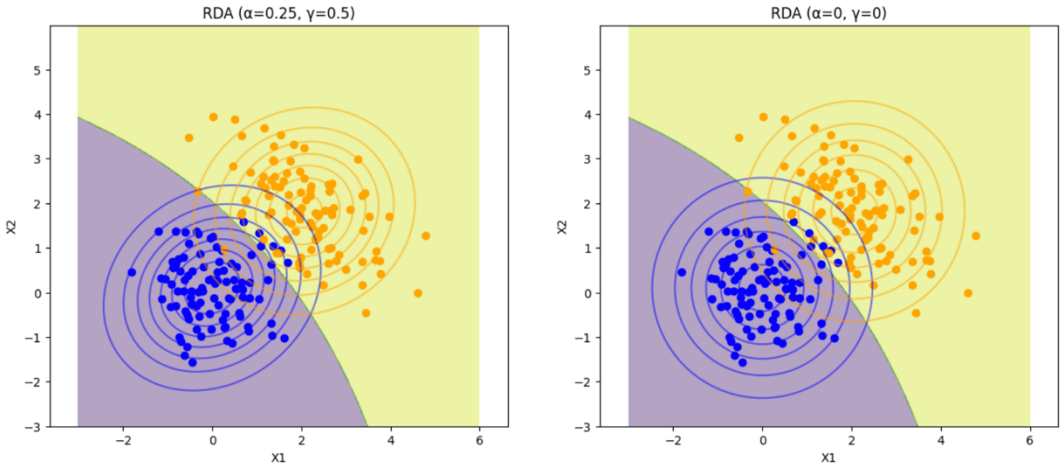
$$(\mathrm{K} \times p) + \frac{p(p+1)}{2}$$

2. **How many parameters have to be estimated in an QDA model with *K* classes and *p* dimensions?**

$$(\mathrm{K} \times p) + \frac{p(p+1)}{2}\mathrm{K}$$

- There are a few methods to maintain some flexibility, yet protect the model from high variance
- One is to use a *regularized covariance matrix* (see ESL 4.3.1). Called Regularized Discriminant Analysis (RDA)

$$\hat{\Sigma}_k(\alpha,\gamma) = \alpha\hat{\Sigma}_k + (1-\alpha)\{\gamma\hat{\Sigma} + (1-\gamma)\hat{\sigma}^2 I_p\}$$



- A special case of above using diagonal covariance matrices only $\left(\hat{\Sigma}_k(\alpha=0,\gamma=0)\right)$. This covariance matrix has all off-diagonal terms set to 0.

$$\hat{\Sigma}_k = diag\left(\hat{\sigma}_1^2, \hat{\sigma}_2^2, \hat{\sigma}_3^2, \ldots, \hat{\sigma}_p^2\right) = \begin{pmatrix} \hat{\sigma}_1^2 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \hat{\sigma}_p^2 \end{pmatrix}$$

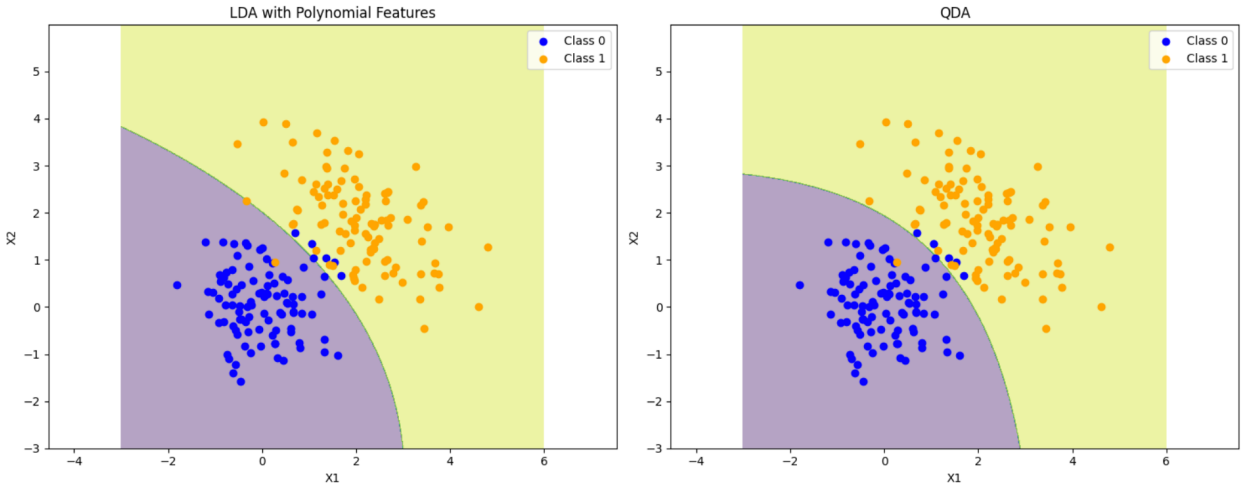
  - This treats predictors/features as uncorrelated/independent.
  - It is a special case of Naïve Bayes!

- A more restrictive (less complex) model specifies that variance in all dimensions are equal

$$\hat{\Sigma}_k = \hat{\sigma}^2 I_p = \begin{pmatrix} \hat{\sigma}^2 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \hat{\sigma}^2 \end{pmatrix}$$

  - This treats predictors/features as uncorrelated/independent.
  - It is a special case of Naïve Bayes!
  - Models all variances as equal.

- In some settings (large K, small p), edf could be reduced by fitting an LDA model in an *enlarged feature space*
  - E.g., for $p = 2$ dimensions, use $X_1, X_2, X_1X_2, X_1^2, X_2^2$ instead of QDA in X_1, X_2.
  - Think basis expansion like what we did with polynomial regression or B-splines
- We can transform the feature space using polynomial features of degree 2 and then applying LDA to this transformed space. This approach can offer a compromise between LDA and QDA, especially in scenarios where the number of classes K is large and the number of features p is small.



Mahalanobis Distance
Notice that a multivariate normal density is a function of the squared *Mahalanobis distance* from x to the mean. $$f(\mathbf{x}) = (2\pi)^{-\frac{p}{2}}\lvert\Sigma\rvert^{-\frac{1}{2}}\exp\left\{-\frac{1}{2}(\mathbf{x}-\mu)^{\intercal}\Sigma^{-1}(\mathbf{x}-\mu)\right\} = (2\pi)^{-\frac{p}{2}}\lvert\Sigma\rvert^{-\frac{1}{2}}\exp\left\{-\frac{1}{2}D_m^2(x)\right\}$$ $D_m(x) = \sqrt{(\mathbf{x}-\mu)^{\intercal}\Sigma^{-1}(\mathbf{x}-\mu)}$ is the Mahalanobis distance.

3.2 Connections: LDA, QDA, and Logistic Regression

ISL 4.5 and ESL 4.4.5 show more details about the parametric form LDA and QDA take.
Recall the notation for generative models:

$$\hat{\gamma}(x) = \log\left(\frac{\hat{p}(x)}{1-\hat{p}(x)}\right) = \log\left(\frac{\hat{\pi}}{1-\hat{\pi}}\right) + \log\left(\frac{\hat{f}_1(x)}{\hat{f}_0(x)}\right)$$

Logistic Regression

$$\hat{\gamma}(x) = \hat{\beta}_0 + \sum_{j=1}^{p} \hat{\beta}_j x_j \quad \text{Main Effects}$$

$$\hat{\gamma}(x) = \hat{\beta}_0 + \sum_{j=1}^{p} \hat{\beta}_j x_j + \sum_{j=1}^{p} \sum_{k=1}^{p} \hat{\beta}_{jk} x_j x_k \quad \text{Quadratic Terms}$$

LDA

$$\hat{\gamma}(x) = \hat{\alpha}_0 + \sum_{j=1}^{p} \hat{\alpha}_j x_j$$

$$\hat{\alpha}_0 = \log \frac{\hat{\pi}}{1-\hat{\pi}} - \frac{1}{2}(\hat{\mu}_1 - \hat{\mu}_0)^{\intercal} \hat{\Sigma} (\hat{\mu}_1 - \hat{\mu}_0)$$

$$\hat{\alpha}_j = \text{the } j^{th} \text{ element of } \hat{\Sigma}^{-1}(\hat{\mu}_1 - \hat{\mu}_0)$$

QDA

$$\hat{\gamma}(x) = \hat{\alpha}_0 + \sum_{j=1}^{p} \hat{\alpha}_j x_j + \sum_{j=1}^{p} \sum_{k=1}^{p} \hat{\alpha}_{jk} x_j x_k$$

$$\hat{\alpha}_0 = \log \frac{\hat{\pi}}{1-\hat{\pi}} - \frac{1}{2} \log \frac{\left|\hat{\Sigma}_1\right|}{\left|\hat{\Sigma}_0\right|} - \frac{1}{2}\left(\hat{\mu}_1^{\intercal} \hat{\Sigma}_1^{-1} - \hat{\mu}_0^{\intercal} \hat{\Sigma}_0^{-1}\right)$$

$$\hat{\alpha}_j = \text{the } j^{th} \text{ element of } \hat{\Sigma}_1^{-1} \hat{\mu}_1 - \hat{\Sigma}_0^{-1} \hat{\mu}_0$$

$$\hat{\alpha}_{jk} = \text{the } (j,k)^{th} \text{ element of } (\hat{\Sigma}_0^{-1} - \hat{\Sigma}_1^{-1})/2$$

3.2.1 Estimation

LDA and QDA estimates model parameters by maximizing the *joint* likelihood:

$$\hat{\alpha} = \arg\max_{\alpha} \Pr(X, Y) = \arg\max_{\alpha} \Pr(X|\, Y) \Pr(Y) = \arg\max_{\alpha} \Pr(Y|\, X) \Pr(X)$$

Logistic Regression estimates model parameters by maximizing the *conditional* likelihood

$$\hat{\beta} = \arg\max_{\beta} \Pr(Y|X)$$

Generative Models
Unlike discriminative models, which differentiate between different kinds of data, generative models aim to capture the underlying data distribution. This means they try to model how the data is formed, capturing the probability distribution of the observed data.

4 Kernel Discriminant Analysis (KDA)

- Model the class conditional densities $f_k(x)$ with a multivariate *kernel density estimate (KDE)*

$$f_k(x) = \frac{1}{n_k} \sum_{i:g_i=k} K(x - x_i; H_k)$$

where H_k is the $p \times p$ bandwidth matrix.

There are three primary approaches to multivariate (p dimensional) KDE:

1. Multivariate kernels
   - e.g., $K(u) = N(0, H)$:

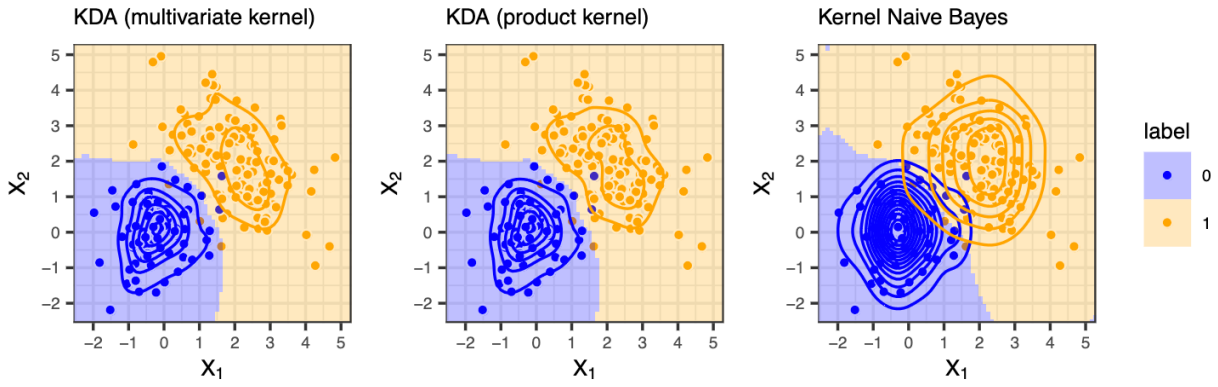
$$\hat{f}_k(x) = \frac{1}{(2\pi)^{\frac{d}{2}} |H|^{\frac{1}{2}} n} \sum_{i=1}^{n} \exp\left(-\frac{1}{2}(x - x_i)^\intercal H_k^{-1}(x - x_i)\right)$$

2. Product Kernels
   - $H_k = diag(h_{k1}, h_{k2}, \dots, h_{kp})$

$$\hat{f}_k(x) = \frac{1}{n} \sum_{i=1}^{n} \left(\prod_{j=1}^{p} K(x_j - x_{ij}; h_{kj}) \right)$$

3. Independence
   - This is a special case of Naïve Bayes (Kernel Naïve Bayes)

$$\hat{f}_k(x) = \prod_{j=1}^{p} \hat{f}_{kj}(x) = \prod_{j=1}^{p} \left(\frac{1}{n} \sum_{i=1}^{n} K(x_j - x_{ij}; h_{kj}) \right)$$



5 Naïve Bayes

$$\Pr(G = g \mid X = x) = \frac{\pi_g \prod_{j=1}^{p} \hat{f}_{gj}(x_j)}{\sum_k \pi_k \prod_{j=1}^{p} \hat{f}_{kj}(x_j)}$$

Naïve Bayes is a generative model that ignores potential associations between predictors and estimates the density of each predictor variable independently.
In the context of machine learning, a "generative model" refers to a type of model that is concerned with understanding and simulating how data is generated.

$$\hat{f}_k(x) = \prod_{j=1}^{p} \hat{f}_{kj}(x_j)$$

- This greatly simplifies the estimation
- The densities do *not* have to be Gaussian (e.g., KDE is a good option)
- Categorical densities (i.e., pmfs) can be thrown in the mix without a problem.
- Because of the independence, this is easy to implement in parallel (and thus can be fast)

The estimated posterior probability under Naïve Bayes becomes

$$\widehat{\Pr}(G = g \mid X = x) = \hat{p}_g(x) = \frac{\hat{\pi}_g \prod_{j=1}^{p} \hat{f}_{gj}(x_j)}{\sum_k \hat{\pi}_k \prod_{j=1}^{p} \hat{f}_{kj}(x_j)}$$

For Binary outcomes the decision function is:

$$\begin{aligned}
\hat{\gamma}(x) &= \log\left(\frac{\hat{p}(x)}{1-\hat{p}(x)}\right) \\
&= \log\left(\frac{\hat{\pi}}{1-\hat{\pi}}\right) + \log\left(\frac{\hat{f}_1(x)}{\hat{f}_0(x)}\right) \\
&= \log\left(\frac{\hat{\pi}}{1-\hat{\pi}}\right) + \log\left(\frac{\prod_{j=1}^{p} \hat{f}_{1j}(x_j)}{\prod_{j=1}^{p} \hat{f}_{0j}(x_j)}\right) \\
&= \log\left(\frac{\hat{\pi}}{1-\hat{\pi}}\right) + \log\left(\prod_{j=1}^{p} \frac{\hat{f}_{1j}(x_j)}{\hat{f}_{0j}(x_j)}\right) \\
&= \log\left(\frac{\hat{\pi}}{1-\hat{\pi}}\right) + \sum_{j=1}^{p} \log\left(\frac{\hat{f}_{1j}(x_j)}{\hat{f}_{0j}(x_j)}\right)
\end{aligned}$$

5.1 Gaussian Naïve Bayes

- Recall in LDA/QDA, the class conditional densities were estimated as Gaussians:

$$\hat{f}_k(x) = \mathcal{N}\left(\mathbf{x};\ \hat{\mu}_k, \hat{\Sigma}_k\right)$$

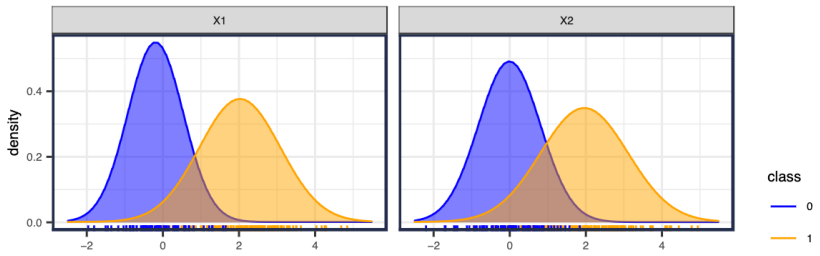
  - But when the dimensionality of **x** gets larger or there is high correlation, estimation of $\hat{\Sigma}_k$ can be poor
- If we force $\hat{\Sigma}_k$ to be *diagonal* then the densities are product of univariate Gaussians (called Gaussian Naïve Bayes)

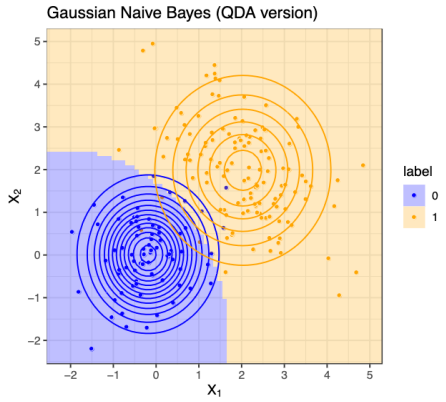
$$\hat{f}_k(x) = \prod_{j=1}^{p} \mathcal{N}\left(x_j;\ \mu_{kj}, \sigma_{kj}\right)$$

  - Even if the data are not independent, this may give better estimates by reducing the variance (at the expense of a bit of bias)
  - This is a special case of QDA, where we restrict the off-diagonal terms in the variance-covariance to be 0.

```
# Sample Data
X = np.array([[1, 2], [2, 3], [3, 2], [4, 3], [5, 5]])
y = np.array([0, 0, 1, 1, 1])
new_data = np.array([3, 3])

# Calculate prior probabilities π_g for each class
unique_classes, counts = np.unique(y, return_counts=True)
prior_probabilities = counts / len(y)

# Calculate mean and variance for each feature in each class
means = {}
variances = {}

for cls in unique_classes:
    features = X[y == cls]
    means[cls] = np.mean(features, axis=0)
    variances[cls] = np.var(features, axis=0)

# Gaussian Naive Bayes likelihood function
def gaussian_likelihood(x, mean, var):
    return (np.exp(-0.5 * ((x - mean) ** 2) / var) / np.sqrt(2 * np.pi * var))

# Calculate the likelihoods and posterior probabilities
posterior_probabilities = []

for cls in unique_classes:
    likelihood = 1
    for j in range(len(new_data)):
        likelihood *= gaussian_likelihood(new_data[j], means[cls][j], variances[cls][j])

    posterior_prob = prior_probabilities[cls] * likelihood
    posterior_probabilities.append(posterior_prob)

# Normalize the posterior probabilities
posterior_probabilities = np.array(posterior_probabilities)
posterior_probabilities /= np.sum(posterior_probabilities)
posterior_probabilities
```

class	predictor	mu	sd	density
0	X1	-0.18	0.73	N(mu = -0.18, sd = 0.73)
0	X2	0.01	0.81	N(mu = 0.01, sd = 0.81)
1	X1	2.04	1.06	N(mu = 2.04, sd = 1.06)
1	X2	1.97	1.15	N(mu = 1.97, sd = 1.15)

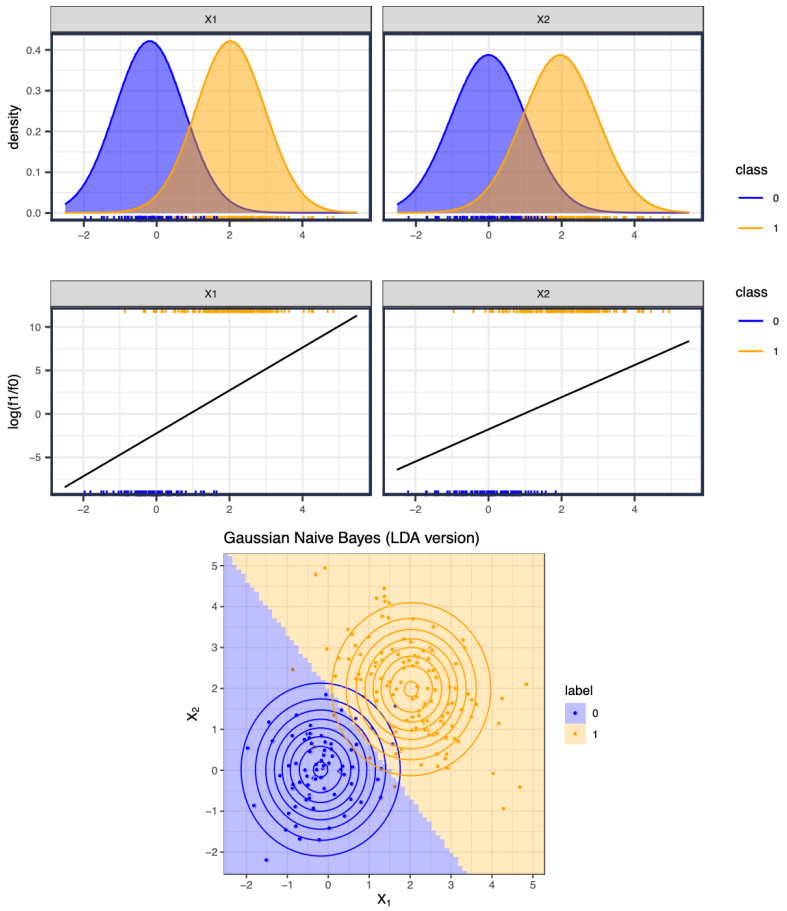




- A simpler model (less complexity/edf) forces a common standard deviation for all class (special case of LDA)

$$\hat{f}_k(x) = \prod_{j=1}^{p} \mathcal{N}(x_j;\ \mu_{kj}, \sigma_j)$$

class	predictor	mu	sd
0	X1	-0.18	0.95
0	X2	0.01	1.03
1	X1	2.04	0.95
1	X2	1.97	1.03



5.2 Kernel Naïve Bayes

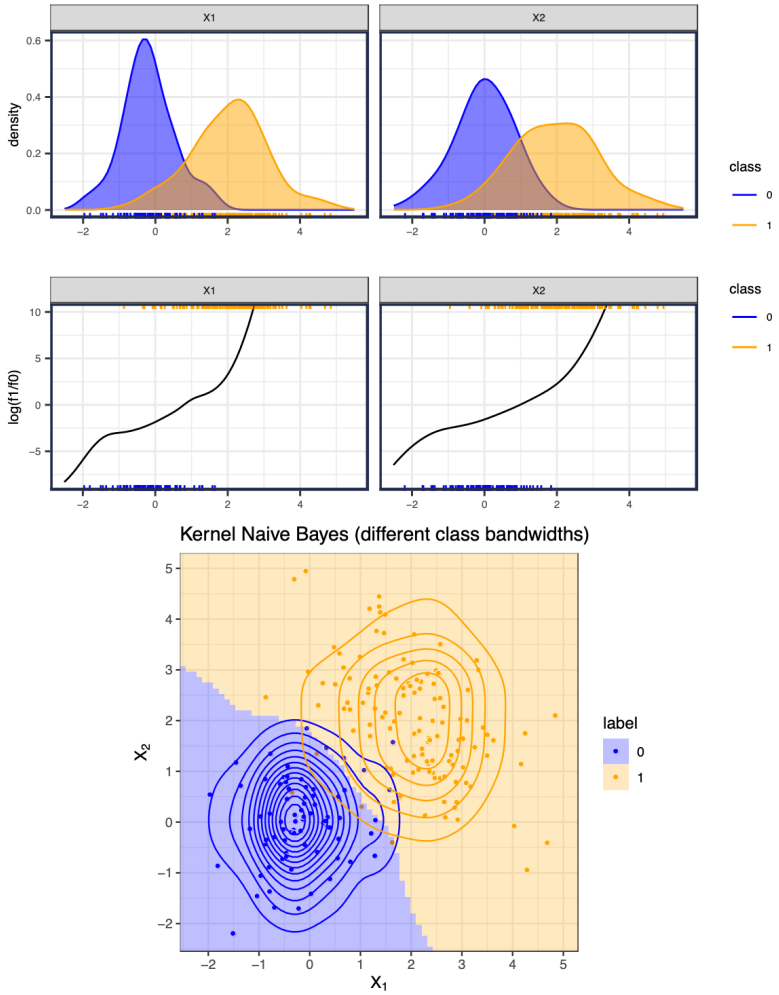
In *kernel density* Naïve Bayes, use Kernel Density Estimation (KDE) to estimate each component density:

$$\hat{f}_{kj}(x_j) = \frac{1}{n_k} \sum_{i:g_i=k} K(x - x_{ij}; h_{kj})$$

With bandwidth parameter h_{kj}.

The density ratio becomes,

$$\frac{\hat{f}_{1j}(x_j)}{\hat{f}_{0j}(x_j)} = \frac{\frac{1}{n_1}\sum_{i:g_i=1} K(x_j - x_{ij}; h_{1j})}{\frac{1}{n_0}\sum_{i:g_i=0} K(x_j - x_{ij}; h_{0j})}$$



- for less complex models, use same bandwidth parameter for each class.

Note: this gives a different solution than using KDE with *product kernel*! (which is not a Naïve Bayes model)

$$\hat{f}_k(\mathbf{x}) = \frac{1}{n_k}\sum_{i:g_i=k}\prod_{j=1}^{p} K(x - x_{ij}; h_{kj})$$

6 Connections: Generalized Additive Models (GAM)

It turns out that there is a close connection between Logistic Regression, Naïve Bayes, and LDA. To help see this, notice that all three methods can be written:

$$\gamma(x) = \log\left(\frac{\pi}{1-\pi}\right) + \log\left(\frac{f_1(x)}{f0(x)}\right)$$
$$= \alpha_0 + \sum_{j=1}^{p} \alpha_j S_j$$

- **Logistic Regression**
  $\hat{\alpha}_0 = \hat{\beta}_0$
  $\hat{\alpha}_j = \hat{\beta}_j$
  $\hat{S}_j = x_j$

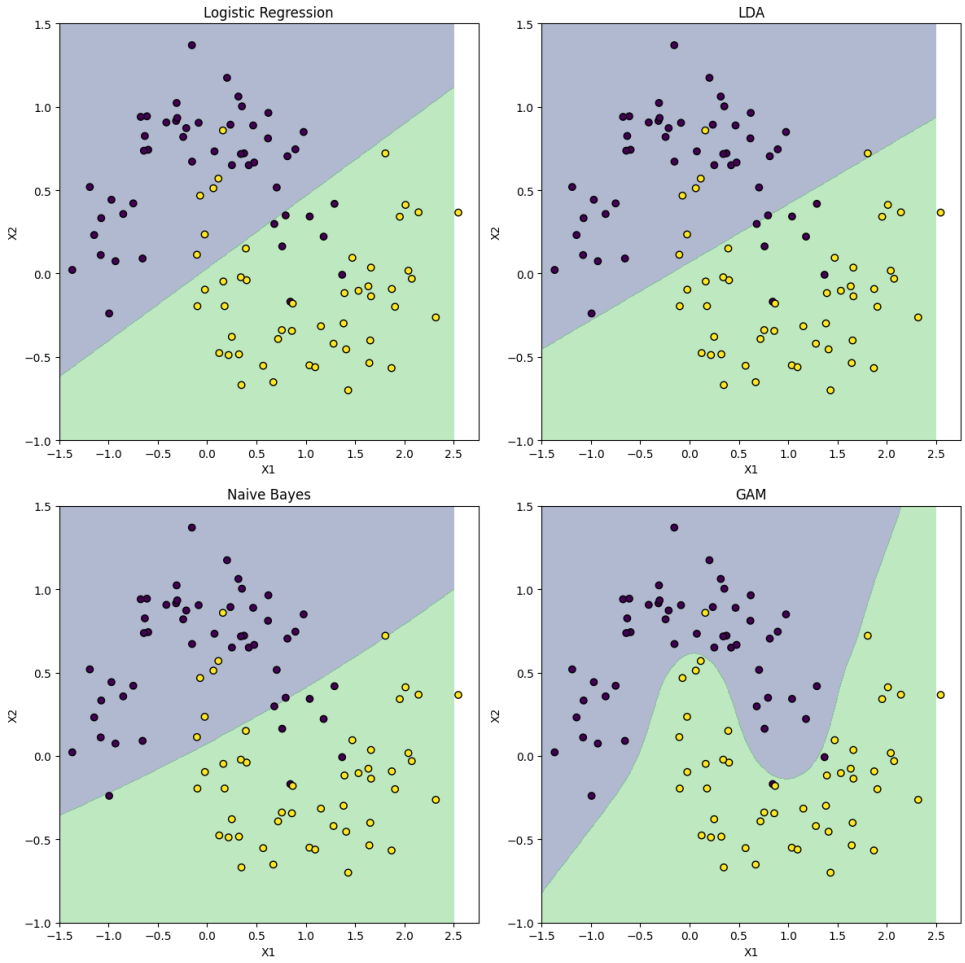
- **Naïve Bayes**
  $\hat{\alpha}_0 = \log\left(\frac{\hat{\pi}}{1-\hat{\pi}}\right)$
  $\hat{\alpha}_j = 1$
  $\hat{S}_j = \log\left(\frac{\hat{f}_{1j}(x_j)}{\hat{f}_{0j}(x_j)}\right)$

- **LDA**
  $\hat{\alpha}_0 = \log\left(\frac{\hat{\pi}}{1-\hat{\pi}}\right) - \frac{1}{2}(\hat{\mu}_1 + \hat{\mu}_0)^{\intercal}\hat{\Sigma}^{-1}(\hat{\mu}_1 - \hat{\mu}_0)$
  $\hat{\alpha}_j = \hat{\Sigma}^{-1}(\hat{\mu}_1 - \hat{\mu}_0)$
  $\hat{S}_j = x_j$

- **Generalized Additive Models (GAM)**
  - GAM models are made to directly estimate models of this form.

$$\hat{\gamma}(x) = \hat{\alpha} + \sum_{j=1}^{p} \hat{g}_j(x_j)$$

  - $\hat{g}_j(x_j)$ is non-linear (usually based on penalized splines)
  - In Python, the pyGAM library is worth becoming familiar with to implement GAM.
  - See ESL 9.1 for more details


Logistic Regression
LDA
Naive Bayes
GAM
X1
X2