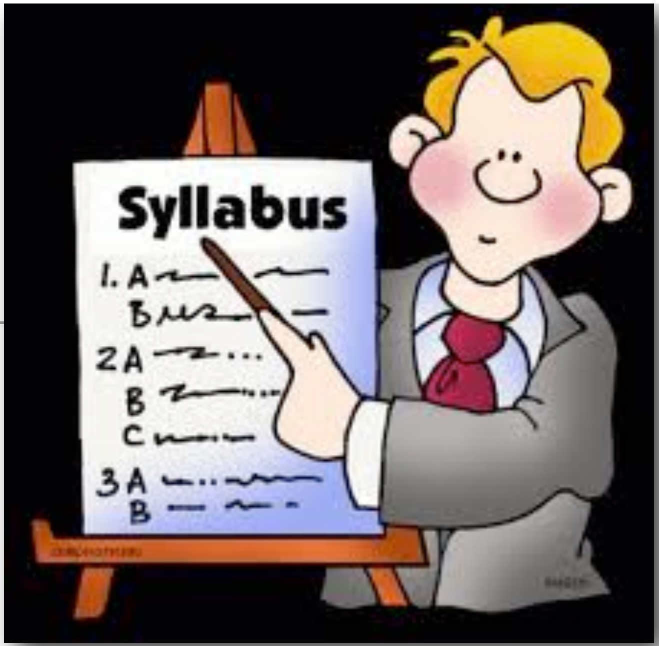
Machine Learning:
Modelos predictivos

Sebastián Moreno
Universidad Adolfo Ibañez

Machine learning
Modelos predictivos
Linear Discriminant Analysis

Contenidos



- Introducción
- Clustering
- **Modelos predictivos**
    - Introducción
    - K-Nearest Neighbors
    - Evaluación
    - Naive Bayes
    - **Linear Discriminant Analysis**
    - Árboles de decisión
    - Support Vector Machine
    - Ensamblados

Modelos predictivos, recordemos Gaussian Naive Bayes

- Gaussian Naive Bayes aprende una distribución de probabilidad condicional.
- Dado un punto **x**, el modelo retorna la “probabilidad” de que **x** pertenezca a una clase específica.
- Gaussian Naive Bayes tiene 4 aspectos claves: probabilidad condicional, el teorema de Bayes, la independencia condicional, y $P(X_i|C) \sim N(\mu, \sigma^2)$
- **Asumamos en forma ingenua (NAIVE)** independencia condicional de X_1, X_2, …,X_m dado C (muy probable que esto no sea cierto).

$$P(C|X_1, X_2, \ldots, X_m) \propto \prod_{i=1}^{m} P(X_i|C)P(C)$$

- Ventaja: Para cada variable tenemos que calcular solo dos parámetros (media y desviación estándar) => m variables => 2*m + K-1 parámetros, donde K es el número de clases.

Modelos predictivos, Linear discriminant analysis, introducción

- Linear discriminant analysis es similar a Gaussian Naive Bayes, pero no asume independencia entre variables.

- Por lo tanto, $P(C_i|X_1, X_2, \ldots, X_k) \propto P(\mathbf{X}|C_i)P(C_i)$

- Sin embargo, se asume que $P(\mathbf{X}|C_i) \sim N(\mu_i, \Sigma)$

- La media difiere para cada clase $\boldsymbol{\mu}_i$, pero la matriz de covarianza $\boldsymbol{\Sigma}$ es la misma independiente de la clase.

- Recordemos que la densidad de probabilidad de una distribución normal multivariada está dada por:

$$P(\mathbf{x}) = P(x_1, \cdots, x_m) = \frac{1}{\sqrt{(2\pi)^m|\Sigma|}} \exp\left(-\frac{1}{2}(\mathbf{x}-\mu)^T\Sigma^{-1}(\mathbf{x}-\mu)\right)$$

Modelos predictivos, Linear discriminant analysis, estimación de parámetros

- Si queremos estimar los parámetros del Linear discriminant analysis podemos usar el método de máxima verosimilitud.
- Este proceso es posible ya que conocemos la clase de cada punto. A diferencia de GMM donde no conocíamos el verdadero cluster (clase) al que pertenecía cada punto.

$$L(\mathcal{D}, M) = \prod_{k=1}^{K} \prod_{i=1}^{N_k} P(C_{ik}|\mathbf{x}_i) \propto \prod_{k=1}^{K} \prod_{i=1}^{N_k} P(\mathbf{x}_i|C_{ik}) P(C_{ik})$$

$$= \prod_{k=1}^{K} \prod_{j=1}^{N_k} \frac{1}{\sqrt{(2\pi)^m |\Sigma|}} \exp\left(-\frac{1}{2}(\mathbf{x}_{kj} - \mu_k)^T \Sigma^{-1} (\mathbf{x}_{kj} - \mu_k)\right) \pi_k$$

$$l = \sum_{k=1}^{K} \sum_{j=1}^{N_k} \left(-\frac{1}{2}\ln(|\Sigma|) - \frac{1}{2}(\mathbf{x}_{kj} - \mu_k)^T \Sigma^{-1} (\mathbf{x}_{kj} - \mu_k) + \ln(\pi_k)\right)$$

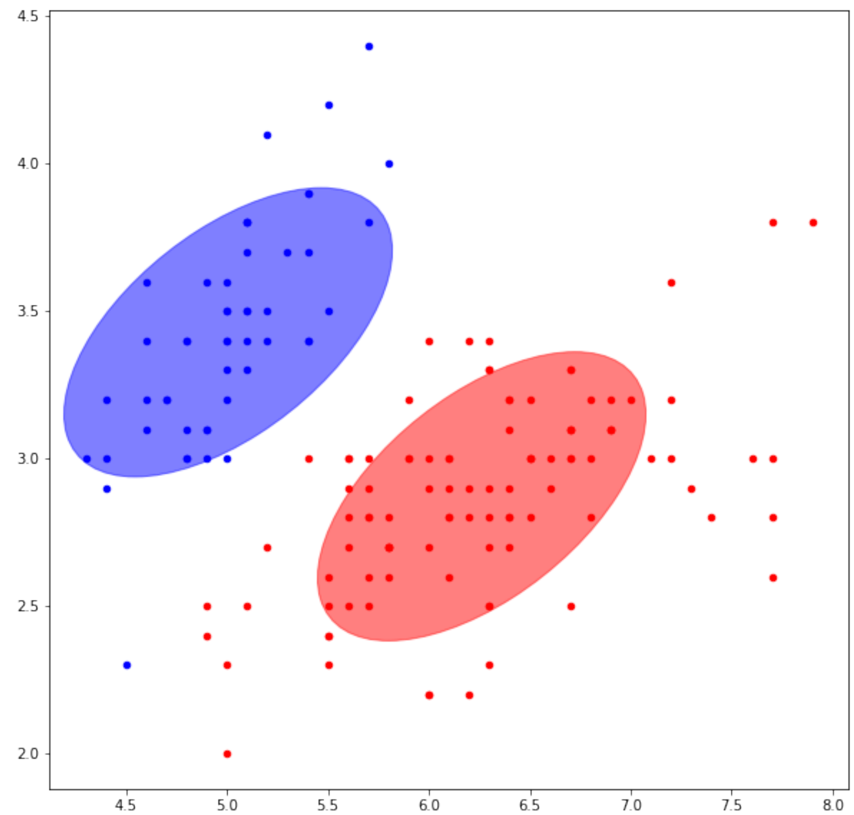
Modelos predictivos, Linear discriminant analysis, estimación de parámetros

- Al derivar con respecto a cada uno de los parámetros e igualar a 0 obtenemos:

$$\hat{\pi}_k = \frac{N_k}{N} \qquad \hat{\mu}_k = \frac{1}{N_k}\sum_{i=1}^{N_k} \mathbf{x}_{ik} \qquad \hat{\Sigma} = \frac{1}{N}\sum_{k=1}^{K}\sum_{i=1}^{N_k}(\mathbf{x}_{ik} - \hat{\mu}_k)(\mathbf{x}_{ik} - \hat{\mu}_k)^T$$

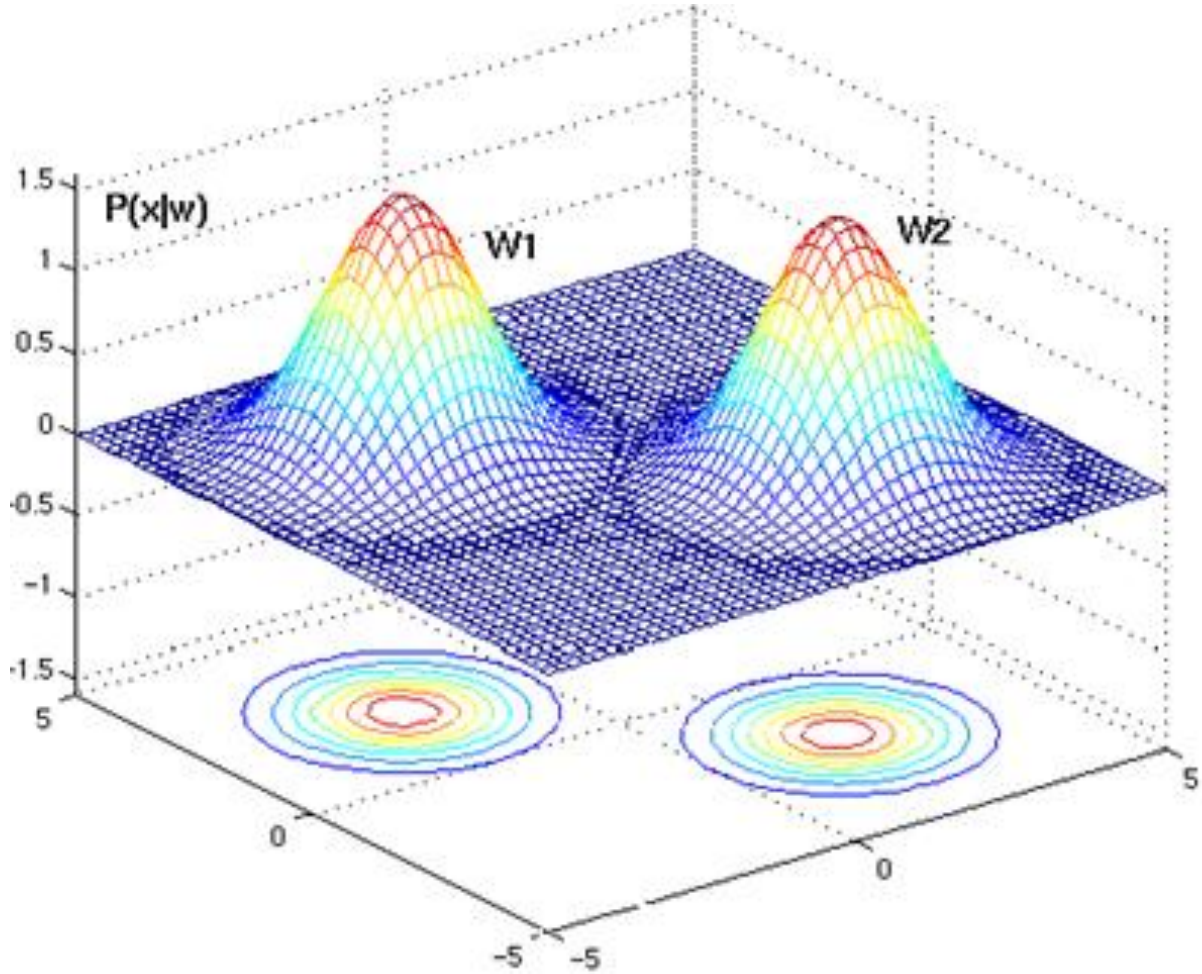
- Gráficamente (en 2 dimensiones), tenemos dos distribuciones gaussianas que están modelando los datos.

- Recuerde, se asumió que ambas distribuciones tienen la misma matriz de covarianza.



Modelos predictivos, Linear discriminant analysis, proceso de clasificación

- Tal como en Naive Bayes, si queremos clasificar un nuevo punto **x** calculamos $P(C_k|\mathbf{x}) \sim P(\mathbf{x}|C_k)P(C_k)$ para cada una de las clases y seleccionamos la que tenga mayor “probabilidad”.



Modelos predictivos, Linear discriminant analysis, proceso de clasificación

$$f(\mathbf{x}|\hat{\mu}_k,\hat{\Sigma}) = \frac{1}{\sqrt{(2\pi)^m|\hat{\Sigma}|}}\exp\left(-\frac{1}{2}(\mathbf{x}-\hat{\mu}_k)^T\hat{\Sigma}^{-1}(\mathbf{x}-\hat{\mu}_k)\right)$$

- Tal como en Naive Bayes, si queremos clasificar un nuevo punto **x** calculamos $P(C_k|\mathbf{x}) \sim P(\mathbf{x}|C_k)P(C_k)$ para cada una de las clases y seleccionamos la que tenga mayor “probabilidad”.

- Equivalentemente: $\arg\max_k(\hat{\pi}_k f(\mathbf{x}|\hat{\mu}_k,\hat{\Sigma}))$

$$= \arg\max_k(\ln(\hat{\pi}_k) + \ln(f(\mathbf{x}|\hat{\mu}_k,\hat{\Sigma})))$$

$$= \arg\max_k\left(\ln(\hat{\pi}_k) - \frac{1}{2}\ln(|\hat{\Sigma}|) - \frac{1}{2}(\mathbf{x}-\hat{\mu}_k)^T\hat{\Sigma}^{-1}(\mathbf{x}-\hat{\mu}_k)\right)$$

independiente de k y multiplicando por 2

$$= \arg\max_k\left(2\ln(\hat{\pi}_k) - (\mathbf{x}-\hat{\mu}_k)^T\hat{\Sigma}^{-1}(\mathbf{x}-\hat{\mu}_k)\right)$$

recordando Mahalanobis y multiplicando por -1

$$= \arg\min_k\left(d_M^2(\mathbf{x};\hat{\mu}_k,\hat{\Sigma}) - 2\ln(\hat{\pi}_k)\right)$$

En la práctica estamos buscando el punto más cercano a su centroide usando la distancia de Mahalanobis y el prior.

Modelos predictivos, Linear discriminant analysis, proceso de clasificación

- ¿Por qué se llama **linear discriminant** si no tiene nada de lineal?

- Primero reescribamos la distancia de Mahalanobis al cuadrado ($d^2_M(.)$)

$$(\mathbf{x} - \mu_k)^T \Sigma^{-1} (\mathbf{x} - \mu_k)$$

$$= (\mathbf{x}^T - \mu_k^T) \Sigma^{-1} (\mathbf{x} - \mu_k)$$

$$= \mathbf{x}^T \Sigma^{-1} \mathbf{x} - \mu_k^T \Sigma^{-1} \mathbf{x} - \mathbf{x}^T \Sigma^{-1} \mu_k + \mu_k^T \Sigma^{-1} \mu_k$$

$$= \mathbf{x}^T \Sigma^{-1} \mathbf{x} - 2\mu_k^T \Sigma^{-1} \mathbf{x} + \mu_k^T \Sigma^{-1} \mu_k$$

Modelos predictivos, Linear discriminant analysis, proceso de clasificación

- ¿Por qué se llama **linear discriminant** si no tiene nada de lineal?

- Segundo, comparemos la clasificación entre dos clases (0 ó 1) (¿será 1?)

$$d_M^2(\mathbf{x}; \hat{\mu}_0, \hat{\Sigma}) - 2\ln(\hat{\pi}_0) > d_M^2(\mathbf{x}; \hat{\mu}_1, \hat{\Sigma}) - 2\ln(\hat{\pi}_1)$$

$$\Rightarrow \mathbf{x}^T\hat{\Sigma}^{-1}\mathbf{x} - 2\hat{\mu}_0^T\hat{\Sigma}^{-1}\mathbf{x} + \hat{\mu}_0^T\hat{\Sigma}^{-1}\hat{\mu}_0 - 2\ln(\hat{\pi}_0) > \mathbf{x}^T\hat{\Sigma}^{-1}\mathbf{x} - 2\hat{\mu}_1^T\hat{\Sigma}^{-1}\mathbf{x} + \hat{\mu}_1^T\hat{\Sigma}^{-1}\hat{\mu}_1 - 2\ln(\hat{\pi}_1)$$

$$\Rightarrow \underbrace{2(\hat{\mu}_1^T - \hat{\mu}_0^T)\hat{\Sigma}^{-1}}_{\mathbf{w}}\mathbf{x} + \underbrace{\hat{\mu}_0^T\hat{\Sigma}^{-1}\hat{\mu}_0 - \hat{\mu}_1^T\hat{\Sigma}^{-1}\hat{\mu}_1 - 2\ln(\hat{\pi}_0) + 2\ln(\hat{\pi}_1)}_{\mathrm{b}} > 0$$

- Entonces, el punto **x** será de la clase 1 si **wx**+b>0 (menor distancia a la clase 1), lo que corresponde a un discriminante lineal.

Modelos predictivos, LDA, código

- El módulo sklearn.discriminant_analysis tiene la clase LinearDiscriminantAnalysis que aplica LDA sobre datos numéricos asumiendo distribuciones Gaussianas para cada clase.
  LinearDiscriminantAnalysis(solver='svd', priors=None, store_covariance=False, tol=0.0001, covariance_estimator=None)

- Parámetro
  solver: método para estimar la matriz de covarianza (‘svd’, ‘lsqr’, ‘eigen’).
  priors: las probabilidades a priori, en caso que hayan sido estimadas.
  store_covariance: False/True. solver="svd" no guarda la matriz de covarianza estimada.
  tol: tolerancia para el método svd (eigenvalues).
  covariance_estimator: matriz de covarianza estimada, en caso que haya sido estimada..

- Atributos
  coef_: vector de pesos (**w**)
  intercept_: intercepto del discriminador lineal (b)
  covariance_: matriz de covarianza estimada.
  means_: medias de las clases (μ_k)
  priors_: distribution a priori (π_k)
  classes_: etiquetas de los puntos

- métodos/funciones
  decision_function(X): aplica la función de decision a un set de datos X ($\ln p(y=k|x)$)
  fit(X, y), predict(X), predict_proba(X), predict_log_proba(X)

Modelos predictivos, LDA, código, ejemplo

- #Creando un objeto LDA con las condiciones iniciales

```
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
LDA = LinearDiscriminantAnalysis(store_covariance=True)
LDA = LDA.fit(X_train,y_train)

#Aplicando el modelo a otros datos
resultado=LDA.predict(X_test)
print("Resultado de la predicción:\n",resultado)
Resultado de la predicción:
[1 0 2 1 1 0 1 2 1 1 2 0 0 0 0 1 2 1 1 2 0 2 0 2 2 2 2 2 0 0 0 0 1 0 0 2 1 0 0 0 2 1 1 0 0 1 1 2 1 2]
print("Resultado original:\n",y_test)
Resultado original:
[1 0 2 1 1 0 1 2 1 1 2 0 0 0 0 1 2 1 1 2 0 2 0 2 2 2 2 2 0 0 0 0 1 0 0 2 1 0 0 0 2 1 1 0 0 1 2 2 1 2]

#Aplicando el modelo a otros datos
resultado=LDA.predict_proba(X_test)
print("Resultado de la predicción:\n",resultado)
Resultado de la predicción:
[[7.86323585e-23 9.99188847e-01 8.11153284e-04]
 [1.00000000e+00 3.82689236e-24 1.81248299e-43]
…
[1.46591816e-44 6.41602631e-07 9.99999358e-01]]
```

Modelos predictivos, LDA, código, ejemplo

- ```
  print("Clases:\n",iris.target_names)
  print("Vector de priors\n",LDA.priors_)
  print("\ncaracterísticas:\n",iris.feature_names)
  print("\nMedias de cada clase\n",LDA.means_)
  print("\nMatriz de covarianza\n”,LDA.covariance_)
  ```

- ```
  Clases:
  ['setosa' 'versicolor' 'virginica']

  Vector de priors
  [0.31 0.35 0.34]

  características:
  ['sepal length (cm)', 'sepal width (cm)', 'petal length (cm)', 'petal width (cm)']

  Medias de cada clase
   [[4.96451613 3.37741935 1.46451613 0.2483871 ]
   [5.85142857 2.72571429 4.22       1.30857143]
   [6.55294118 2.97058824 5.54411765 2.01176471]]

  Matriz de covarianza
   [[0.28103102 0.08891817 0.18335556 0.03396621]
   [0.08891817 0.10411639 0.06421279 0.03447921]
   [0.18335556 0.06421279 0.19750791 0.04479579]
   [0.03396621 0.03447921 0.04479579 0.04420142]]
  ```

Machine learning
Modelos predictivos
Quadratic Discriminant Analysis

Modelos predictivos, recordemos Linear Discriminant Analysis

- Linear Discriminant Analysis aprende una distribución de probabilidad condicional

$$P(C_i|X_1, X_2, \ldots, X_k) \propto P(\mathbf{X}|C_i)P(C_i)$$

- Dado un punto **x**, el modelo retorna la "probabilidad" de que **x** pertenezca a una clase específica, asumiendo

$$P(\mathbf{x}|C_i) \sim N(\mu_i, \Sigma)$$

- En el caso de Quadratic Discriminant Analysis, se asume que cada clase tiene su propia matriz de covarianza, obteniendo

$$P(\mathbf{x}|C_i) \sim N(\mu_i, \Sigma_i)$$

Modelos predictivos, Quadratic discriminant analysis, estimación de parámetros

- Si queremos estimar los parámetros del Quadratic discriminant analysis podemos usar el método de máxima verosimilitud.

- El proceso es el mismo que antes, excepto que cada clase tiene su propia matriz de covarianza.

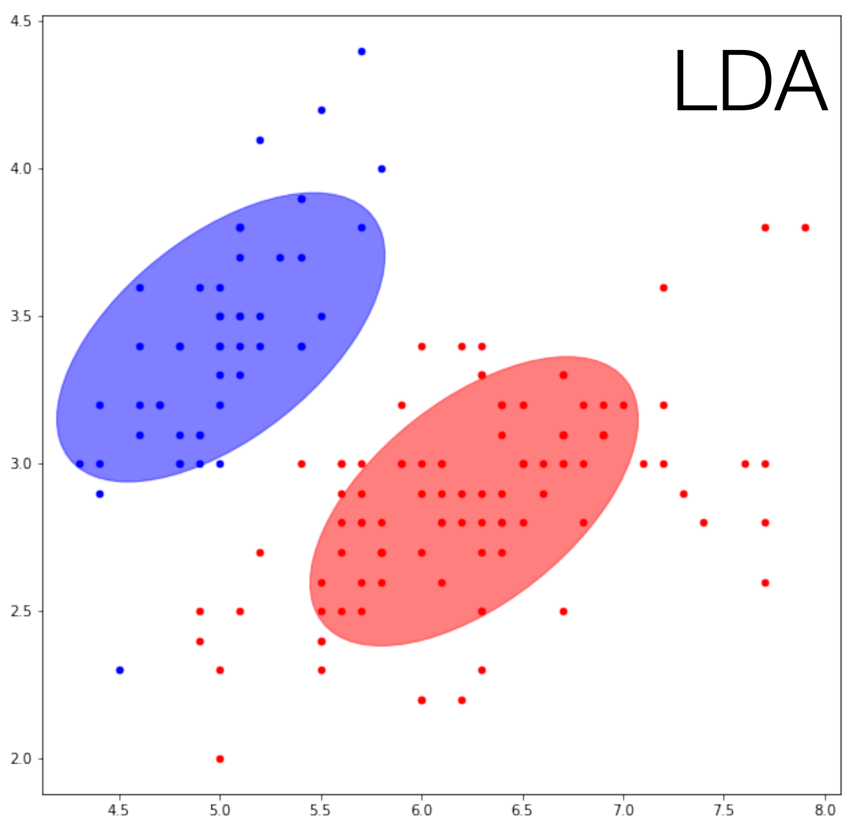
$$l(\mathcal{D}, \mathcal{M}) = \sum_{k=1}^{K} \sum_{j=1}^{N_k} \left(-\frac{1}{2} \ln(|\Sigma_k|) - \frac{1}{2} (\mathbf{x}_{kj} - \mu_k)^T \Sigma_k^{-1} (\mathbf{x}_{kj} - \mu_k) + \ln(\pi_k) \right)$$

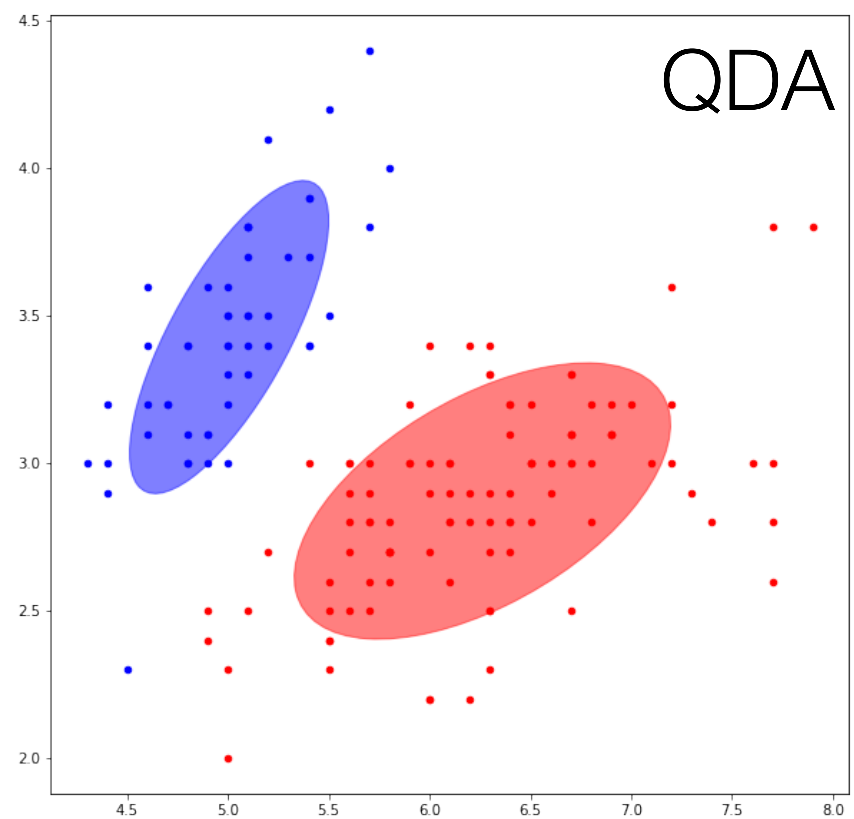
- Al derivar con respecto a cada uno de los parámetros e igualar a 0 obtenemos:

$$\hat{\pi}_k = \frac{N_k}{N} \qquad \hat{\mu}_k = \frac{1}{N_k} \sum_{i=1}^{N_k} \mathbf{x}_{ik} \qquad \hat{\Sigma}_k = \frac{1}{N_k} \sum_{i=1}^{N_k} (\mathbf{x}_{ik} - \hat{\mu}_k)(\mathbf{x}_{ik} - \hat{\mu}_k)^T$$

Modelos predictivos, Quadratic discriminant analysis, estimación de parámetros

- Gráficamente (en 2 dimensiones), tenemos dos o más distribuciones gaussianas que están modelando los datos.
- Recuerde, linear discriminant analysis asume que ambas distribuciones tienen la misma matriz de covarianza, quadratic discriminant analysis no.





Modelos predictivos, Quadratic discriminant analysis, proceso de clasificación

- El proceso de clasificación es el mismo que el linear discriminant analysis. Si queremos clasificar un nuevo punto **x** calculamos $P(C_k|\mathbf{x}) \sim P(\mathbf{x}|C_k)P(C_k)$ para cada una de las clases y seleccionamos la que tenga mayor probabilidad.

- Equivalentemente: $$\arg\max_k(\hat{\pi}_k f(\mathbf{x}|\hat{\mu}_k, \hat{\Sigma}_k))$$ $$= \arg\min_k \left(d^2_M(\mathbf{x}; \hat{\mu}_k, \hat{\Sigma}_k) - 2\ln(\hat{\pi}_k)\right)$$

- comparemos la clasificación entre dos clases (0 ó 1) (¿será 1?)

$$d^2_M(\mathbf{x}; \hat{\mu}_0, \hat{\Sigma}_k) - 2\ln(\hat{\pi}_0) > d^2_M(\mathbf{x}; \hat{\mu}_1, \hat{\Sigma}_k) - 2\ln(\hat{\pi}_1)$$

$$\Rightarrow \mathbf{x}^T\hat{\Sigma}_0^{-1}\mathbf{x} - 2\hat{\mu}_0^T\hat{\Sigma}_0^{-1}\mathbf{x} + \hat{\mu}_0^T\hat{\Sigma}_0^{-1}\hat{\mu}_0 - 2\ln(\hat{\pi}_0) > \mathbf{x}^T\hat{\Sigma}_1^{-1}\mathbf{x} - 2\hat{\mu}_1^T\hat{\Sigma}_1^{-1}\mathbf{x} + \hat{\mu}_1^T\hat{\Sigma}_1^{-1}\hat{\mu}_1 - 2\ln(\hat{\pi}_1)$$

$$\Rightarrow \mathbf{x}^T\underbrace{(\hat{\Sigma}_0 - \hat{\Sigma}_1)^{-1}}_{\mathbf{A}}\mathbf{x} + \underbrace{2(\hat{\Sigma}_1^{-1}\hat{\mu}_1 - \hat{\Sigma}_0^{-1}\hat{\mu}_0)}_{\mathrm{b}}\mathbf{x} + \underbrace{\hat{\mu}_0^T\hat{\Sigma}_0^{-1}\hat{\mu}_0 - \hat{\mu}_1^T\hat{\Sigma}_1^{-1}\hat{\mu}_1 - 2\ln(\hat{\pi}_0) + 2\ln(\hat{\pi}_1)}_{\mathrm{c}} > 0$$

- Entonces, el punto **x** será de la clase 1 si $\mathbf{x}^T\mathbf{A}\mathbf{x}+\mathrm{b}\mathbf{x}+\mathrm{c}>0$, lo que corresponde a un discriminante cuadrático.

Modelos predictivos, QDA, código

- El módulo sklearn.discriminant_analysis tiene la clase QuadraticDiscriminantAnalysis que aplica QDA sobre datos numéricos asumiendo distribuciones Gaussianas para cada clase.
  QuadraticDiscriminantAnalysis(*, priors=None, store_covariance=False)

- Parámetro
  priors: las probabilidades a priori, en caso que hayan sido estimadas.
  store_covariance: False/True. solver="svd" no guarda la matriz de covarianza estimada.

- Atributos
  covariance_: matriz de covarianza estimada.
  means_: medias de las clases (μ_k)
  priors_: distribution a priori (π_k)
  classes_: etiquetas de los puntos

- métodos/funciones
  decision_function(X): aplica la función de decision a un set de datos X ($\ln p(y=k|x)$)
  fit(X, y): "Entrenamiento" del modelo, se tiene que dar los datos y la clase Y
  predict(X): predice las etiquetas para los puntos dados
  predict_proba(X): predice las probabilidades de las clases para los puntos dados
  predict_log_proba(X): predice el logaritmo de las probabilidades de las clases para los puntos dados (mayor estabilidad)

Modelos predictivos, QDA, código, ejemplo

```
#Creando un objeto QDA con las condiciones iniciales
from sklearn.discriminant_analysis import QuadraticDiscriminantAnalysis
QDA = QuadraticDiscriminantAnalysis(store_covariance=True)
QDA = QDA.fit(X_train,y_train)

#Aplicando el modelo a otros datos
resultado=QDA.predict(X_test)
print("Resultado de la predicción:\n",resultado)
Resultado de la predicción:
[1 0 2 1 1 0 1 2 1 1 2 0 0 0 0 1 2 1 1 2 0 2 0 2 2 2 2 2 0 0 0 0 1 0 0 2 1 0 0 0 2 1 1 0 0 1 1 2 1 2]
print("Resultado original:\n",y_test)
Resultado original:
[1 0 2 1 1 0 1 2 1 1 2 0 0 0 0 1 2 1 1 2 0 2 0 2 2 2 2 2 0 0 0 0 1 0 0 2 1 0 0 0 2 1 1 0 0 1 2 2 1 2]

#Aplicando el modelo a otros datos
resultado=QDA.predict_proba(X_test)
print("Resultado de la predicción:\n",resultado)
Resultado de la predicción:
[[1.04680589e-081 9.89719761e-001 1.02802394e-002]
 [1.00000000e+000 2.17102276e-027 2.70439300e-064]
…
[1.58363065e-182 7.08541122e-008 9.99999929e-001]]
```

Modelos predictivos, QDA, código, ejemplo

- ```
  print("\ncaracterísticas:\n",iris.feature_names)
  print("\nMedias de cada clase\n",QDA.means_)
  print("\nMatriz de covarianza\n”,QDA.covariance_)
  ```

- ```
  características:
  ['sepal length (cm)', 'sepal width (cm)', 'petal length (cm)', 'petal width (cm)']

  Medias de cada clase
   [[4.96451613 3.37741935 1.46451613 0.2483871 ]
   [5.85142857 2.72571429 4.22       1.30857143]
   [6.55294118 2.97058824 5.54411765 2.01176471]]

  Matriz de covarianzas
   [array([[0.11569892, 0.09817204, 0.01669892, 0.00677419],
         [0.09817204, 0.14113978, 0.01950538, 0.00712903],
         [0.01669892, 0.01950538, 0.03436559, 0.00877419],
         [0.00677419, 0.00712903, 0.00877419, 0.01191398]]),
  array([[0.27963025, 0.08393277, 0.20070588, 0.05572269],
         [0.08393277, 0.08961345, 0.09123529, 0.04094958],
         [0.20070588, 0.09123529, 0.25164706, 0.08364706],
         [0.05572269, 0.04094958, 0.08364706, 0.04198319]]),
  array([[0.45832442, 0.09372549, 0.33365419, 0.03935829],
         [0.09372549, 0.09486631, 0.08285205, 0.05581105],
         [0.33365419, 0.08285205, 0.30799465, 0.04158645],
         [0.03935829, 0.05581105, 0.04158645, 0.0798574 ]])]
  ```

Machine learning
Modelos predictivos
Comparando modelos

Modelos predictivos, comparando modelos

- Gaussian Naive Bayes (NB), Linear discriminant analysis (LDA), y Quadratic discriminant analysis (QDA) son modelos con similitudes y diferencias.

- Los tres modelos se basan en $P(C_k|\mathbf{x})$ y usan distribuciones Gaussianas para modelar los datos.

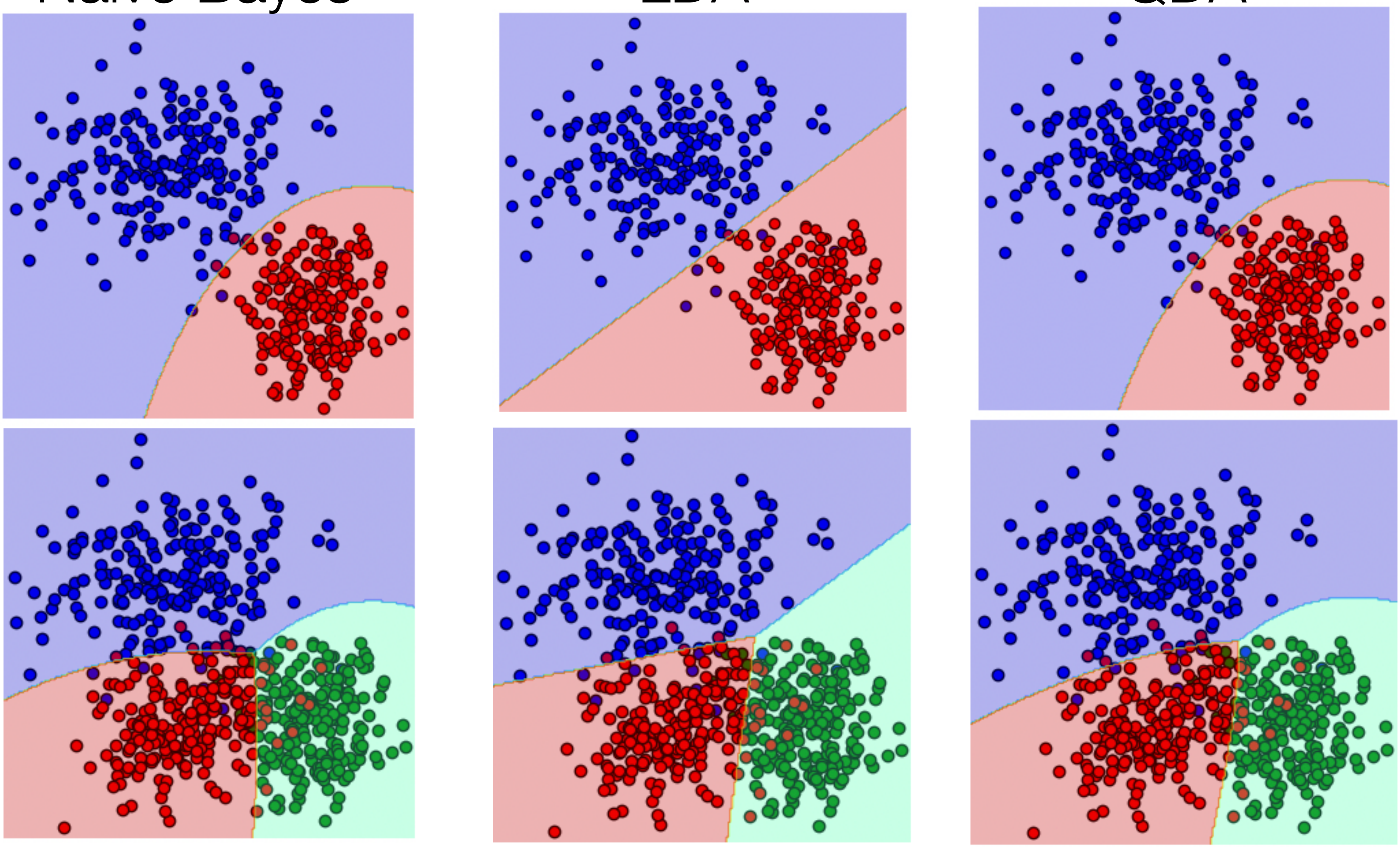
- Dado un problema de K clases y m dimensiones, el número de parámetros por modelo es (prior + media + covarianza):
  -NB: (K-1) + mK + mK => desviación estándar de cada clase
  -LDA: (K-1) + mK + m(m+1)/2 => matriz de Covarianza
  -QDA: (K-1) + mK + K*m(m+1)/2 => matriz de Covarianza para cada clase

- Un modelo con más parámetros se asocia a mayor flexibilidad (se adapta mejor a los datos) => predicción con menor sesgo, mayor varianza.

- Un modelo con más parámetros necesita más datos de entrenamiento, caso contrario podría sufrir de sobreentrenamiento.

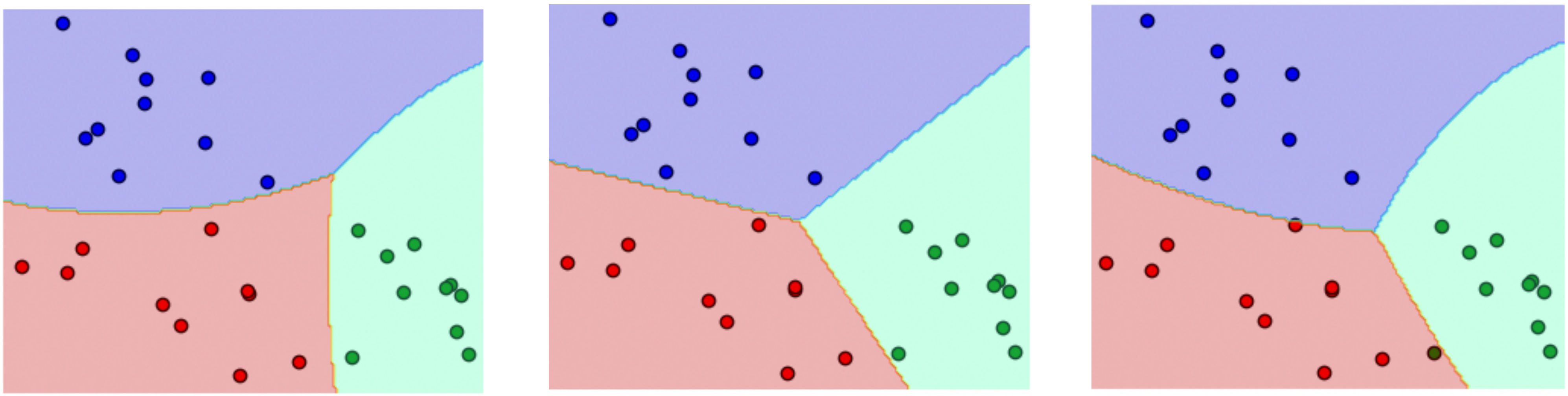
Modelos predictivos, comparando modelos, ejemplos

Naive Bayes | LDA | QDA



Modelos predictivos, comparando modelos, ejemplos

Naive Bayes | LDA | QDA

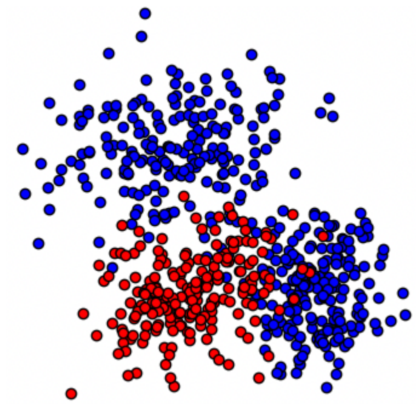


¿Por qué Naive Bayes es capaz de modelar el punto verde, pero no QDA?

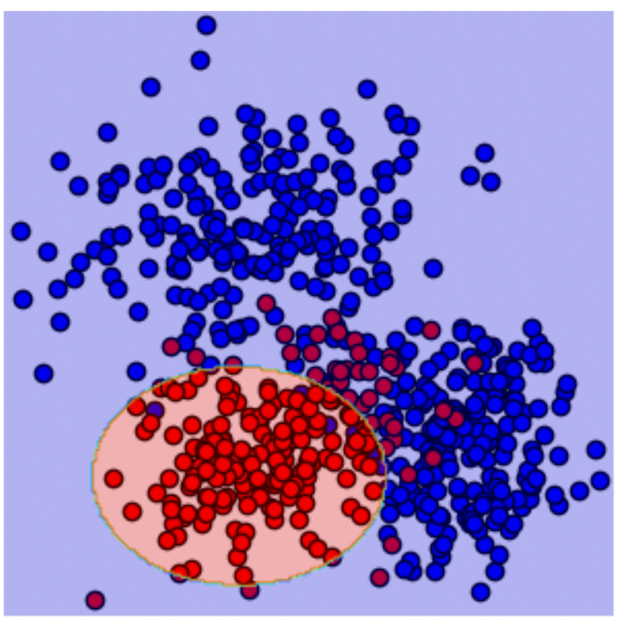
¿Cuál de los dos modelos debiera ser mejor?

Modelos predictivos, comparando modelos, ejemplos

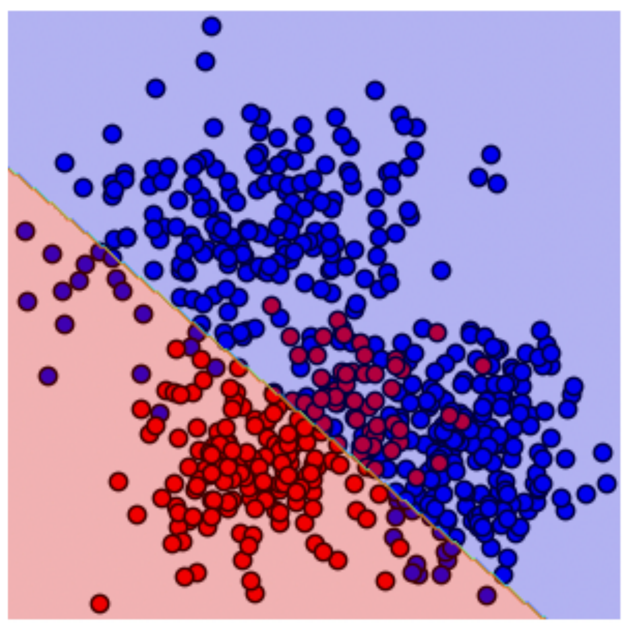
¿Existirá alguna diferencia significativa entre GNB y QDA en este set de datos? ¿por qué?



Naive Bayes



LDA



QDA