1. PCA (Analisis de Componentes Principales).

Modelo: Reducción de dimensionalidad lineal que busca nuevas características (componentes principales) con la mayor varianza

Optimización: Encuentra la matriz de transformación W que maximiza la varianza proyectada, resolviendo para los vectores propios de la matriz de covarianza Σ maximizar traza $(W^T \Sigma W)$ sujeto a $W^T W = I$.

2 UMAP (Uniform Manifold Approximation and Projection)

Modelo: Reducción de dimension no lineal que preserva la estructura topológica de los datos

Optimización: Minimiza la divergencia de entropía cruzada. entre las distribuciones de probabilidad de similitud en espacio original (p_{ij}) y el espacio reducido (q_{ij}).

$$L = \Sigma_{ij}\left[p_{ij}\log\left(\frac{p_{ij}}{q_{ij}}\right) + (1-p_{ij})\log\left(\frac{1-p_{ij}}{1-q_{ij}}\right)\right]$$

esto se logra mediante descenso de gradiente estocástico. El parámetro n-neighbors es crucial para balancear la preservación local y global.

3. Naive Bayes (GaussianNB)

Modelo: Clasificador probabilístico que asume independencia condicional de las características y modela las características continuas con una distribución Gaussiana

Optimización: no tiene optimización iterativa el aprendizaje es la estimación de probabilidades (máxima verosimilitud). probabilidad a priori de clase $P(C_k) = \frac{N_k}{N}$

Probabilidad condicional (Gaussiana):

$$P(x_j \mid C_k) = \frac{1}{\sqrt{2\pi\sigma^2_{jk}}} \exp\left(-\frac{(x_j - \mu_{jk})^2}{2\sigma^2_{jk}}\right).$$

4. SGDClassifier (Stochastic Gradient Descent classifier)

Modelo: Optimizador que entrena modelos lineales (como Regresión logística, SVM) usando descenso de gradiente estocástico.

Optimización: minimizar una función de pérdida. convexa. L (ej. pérdida logarítmica o hinge) para los pesos w, con actualizaciones estocásticas $w \leftarrow w - \eta \nabla L (w, x_i, y_i)$ $W \leftarrow W - \eta \nabla L (w, x,i, y.i)$ donde η es la taza de aprendizaje. La función de pérdida general es: $\min w \, 1 N \Sigma i = 1 N L (y_i, h(x_i; W)) + \alpha R(w)$ $\min w \; N \, 1 \, \Sigma i = 1 N L (y_i; h(x,i, w)) + \alpha R(w)$.

5. Logistic Regresión (Regresión Logística)

Modelo: clasificador lineal que predice la probabilidad de clase usando la función sigmoide / softmax.

Optimización: Encuentra los pesos (w, b) que maximiza la la verosimilitud, equivalente a minimizar la entropía cruzada negativa

$J(w, b) = -1 N \Sigma i = 1 N [y_i \log(\hat{y}^i) + y_i \log(1 - \hat{y}^i)]$

$J(w, b) = -N 1 \Sigma i = 1 N [y_i \log(\hat{y}^i) + (1 - y_i) \log(1 - \hat{y}^i)$

donde $\hat{y}_i = \sigma(w^T x_i + b)$. es un problema de optimización convexa.

6. Linear Discriminant Analysis (LDA).

Modelo: Clasificador y reductor de dimensionalidad lineal que busca la máxima separación entre clases

Optimización: Maximiza la razón de la dispersión entre clases S_B y la dispersión dentro de clases S_w encontrando vectores propios generalizados de $S_w^{-1} S_B$: $J(w) = \frac{|w^T S_B w|}{|w^T S_w w|}$

7. KNeighbors Classifier (KNN)

Modelo: Algoritmo no paramétrico y perezoso que clasifica una nueva muestra basándose en la clase mayoritaria de sus k vecinos más cercanos.

Optimización: No tiene un problema de optimización explícito. El "entrenamiento" es almacenar los datos. La elección de k es clave.

8: SVC (Support Vector Classifier)

Modelo: Clasificador que busca un hiperplano óptimo que maximice el margen entre clases, usando el "truco de Kernel" para no linealidades

Optimización: minimiza una función de pérdida que balancea la maximización del margen y la penalización de errores con variables de holgura ξ_i $\min_{w,b,\xi} \frac{1}{2}\|w\|^2 + C \Sigma_{i=1}^{n} \xi_i$ sujeto a $y_i(w^T x_i + b) \geq 1 - \xi_i$ y $\xi_i \geq 0$. es una optimización cuadrática convexa.

9. Random forest classifier (Bosque aleatorio).

- **Modelo:** un ensamble de múltiples árboles de decisión entrenados de forma aleatoria (bagging y subconjuntos de características).
- **Optimización:** no hay una optimización global cada árbol se construye greedymente, busca divisiones que maximizan la ganancia de información o minimizan la impureza (ej. Gini o Entropía) en cada nodo: maximizar Ganancia de información = Impureza (Padre) − $\Sigma_{c \in \text{Hijos}} \frac{N_c}{N}$ Impureza (Hijo$_c$).

10 GaussianProcessClassifier (GPC).

Modelo: Clasificador probabilístico y no paramétrico basado en procesos gaussianos, que ofrece predicciones con incertidumbre.

Optimización: aproxima la integridad de la probabilidad posterior y maximiza la margen log-verosimilitud con respecto a los hiperparámetros de Kernel. θ: $\log P(y \mid X, \theta) = \log \int P(y \mid f) P(f \mid X, \theta) df$

11. Clasificadores basados en Deep learning

Modelos: Redes neuronales artificiales (CNNs, RNNs, Transformers) con múltiples capas para aprender representaciones complejas

Optimización: Minimiza una función de pérdida (ej. la entropía cruzada categórica) para los pesos y sesgos de la red: $\min_{W,b} L(Y, \hat{Y}(X; W, b))$ esto se logra usando descenso de gradiente estocástico (SGD) y sus variantes (Adam, RMSprop) con retropropagación. Es un problema de optimización no convexa.