PROBLEMAS HIPERDIMENSIONALES

ESTRUCTURA DE LA CLASE

INTRODUCCIÓN

Presentación del tema de la clase

Introducción a las dimensiones

SVMs

Vectores de soporte

Búsqueda de hiperplano óptimo

Implementación para clasificación binaria

Clasificación multiclase

Kernels para problemas no lineales

Selección de kernels

TUNING

Hiperparámetros

Selección con GridSearch

Implementación de GridSearchCV

Métricas de evaluación de modelos

PCA

Reducción de dimensionalidad

Implementación de PCA

Creación de Pipelines

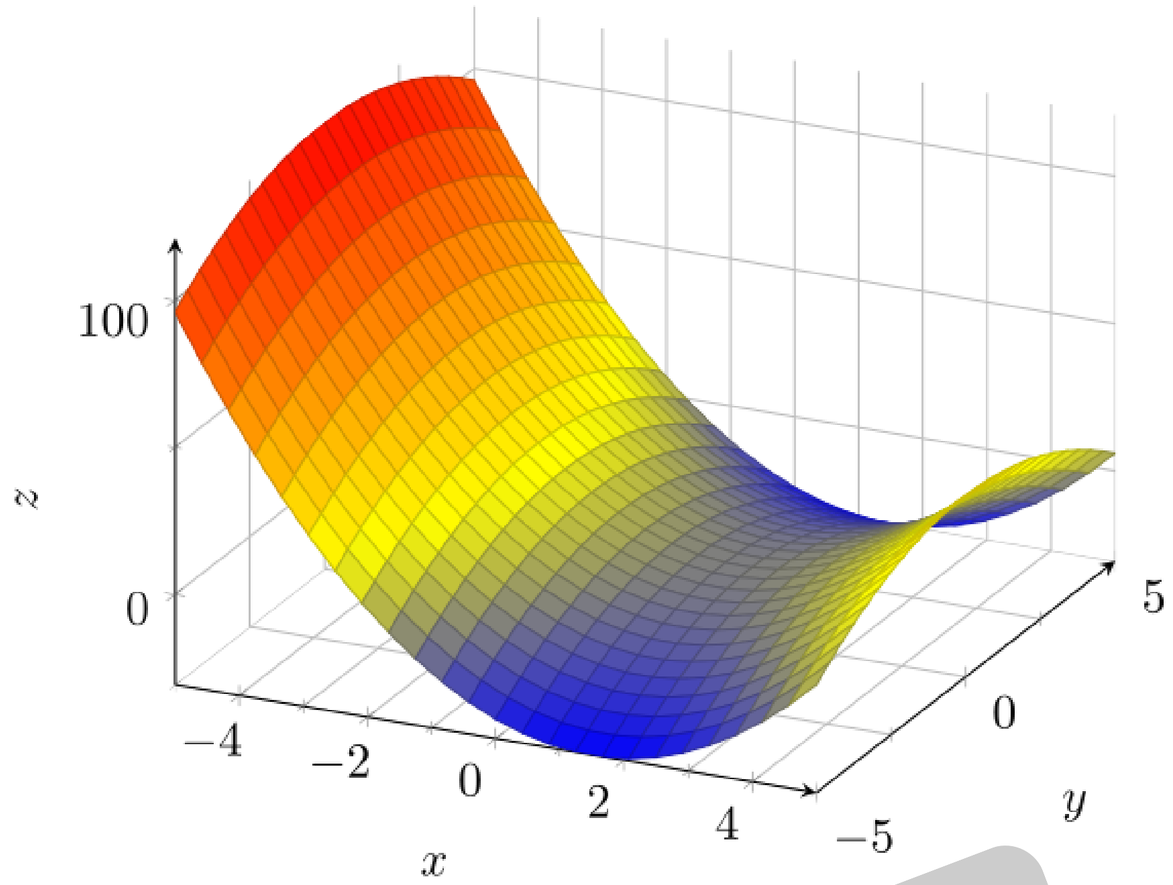
ACTIVIDAD PRÁCTICA

Identity

CONCLUSIONES

Recapitulación de los puntos clave de la clase

AJUSTE DE CURVAS

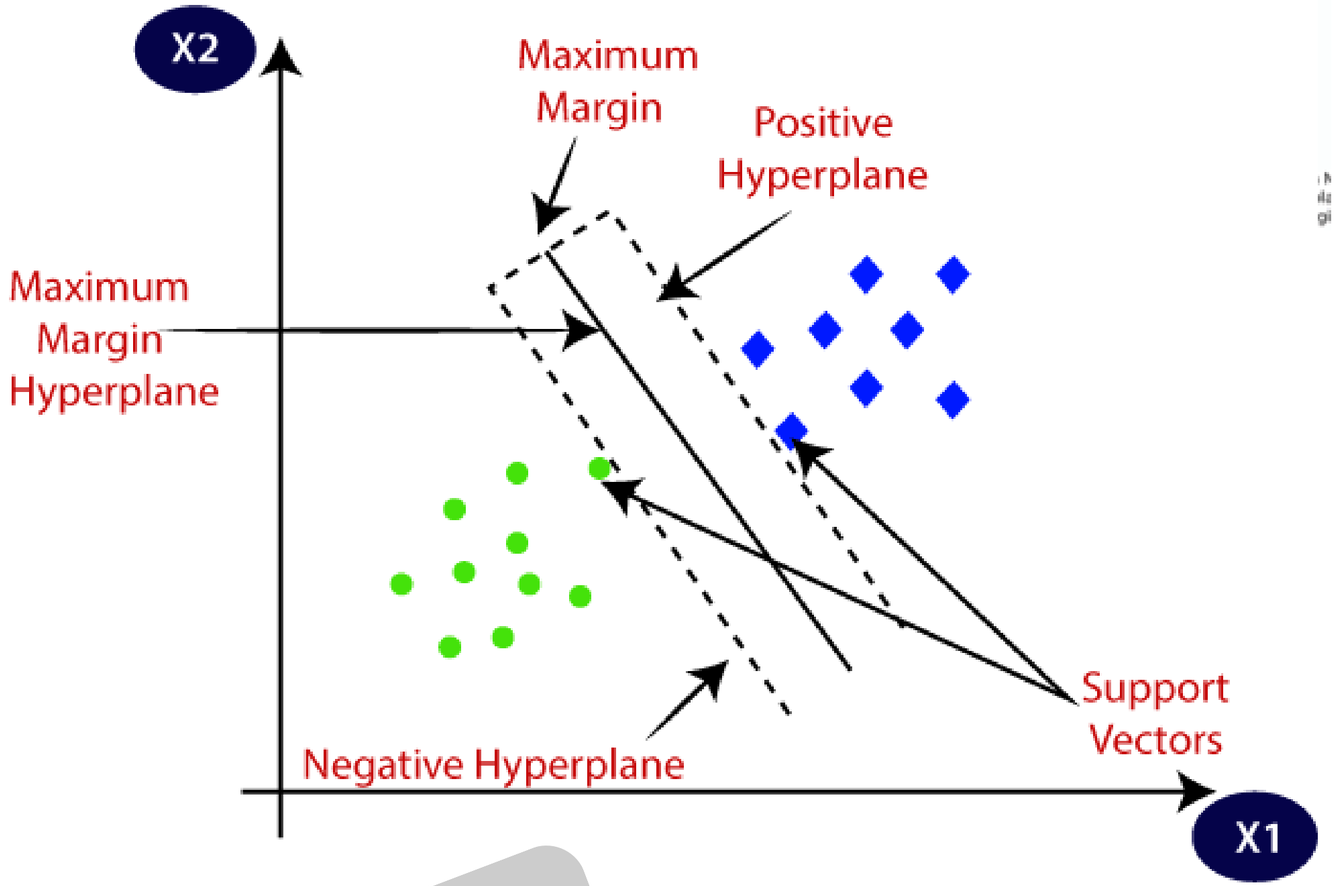


Los modelos de IA buscan ajustar una función a un conjunto de puntos

Cada feature corresponde a una dimensión en un espacio vectorial

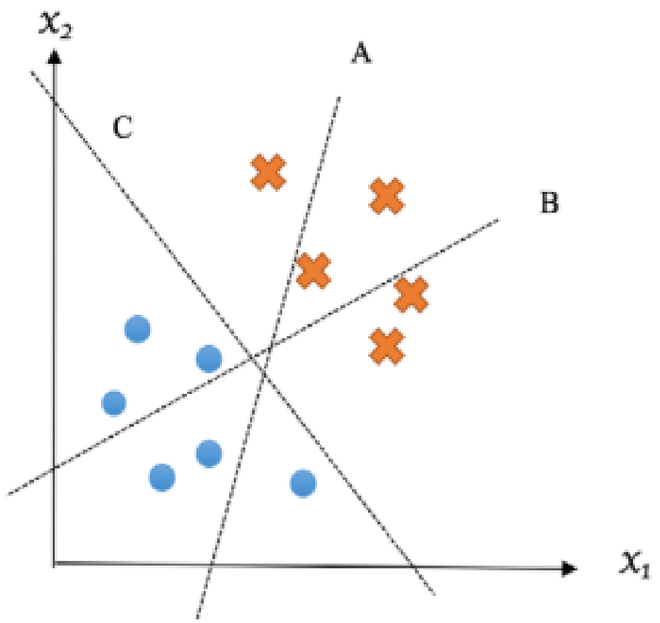
El hiperplano corresponde a frontera de decisión (C) o generalización del grupo (R)

VECTORES DE SOPORTE

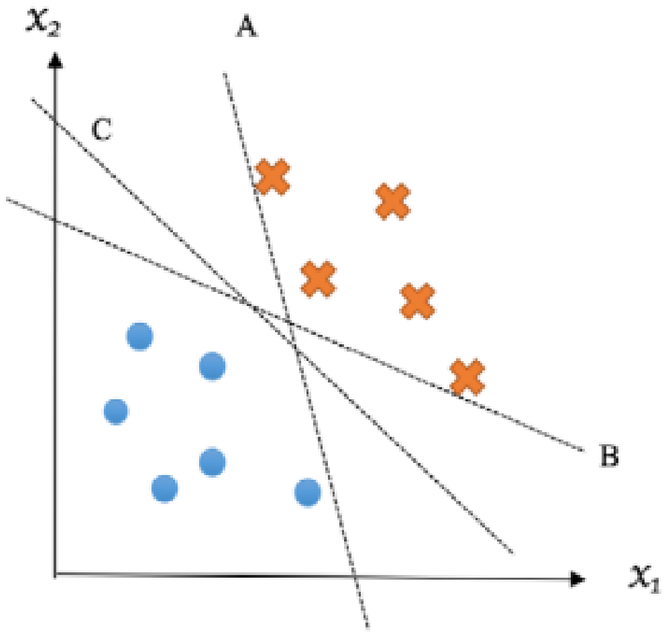


VECTORES DE SOPORTE

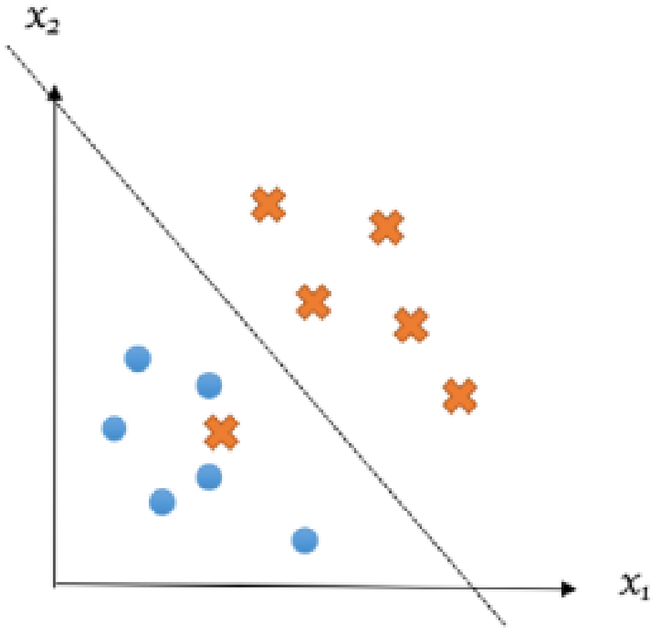
SEPARACIÓN



OPTIMIZACIÓN



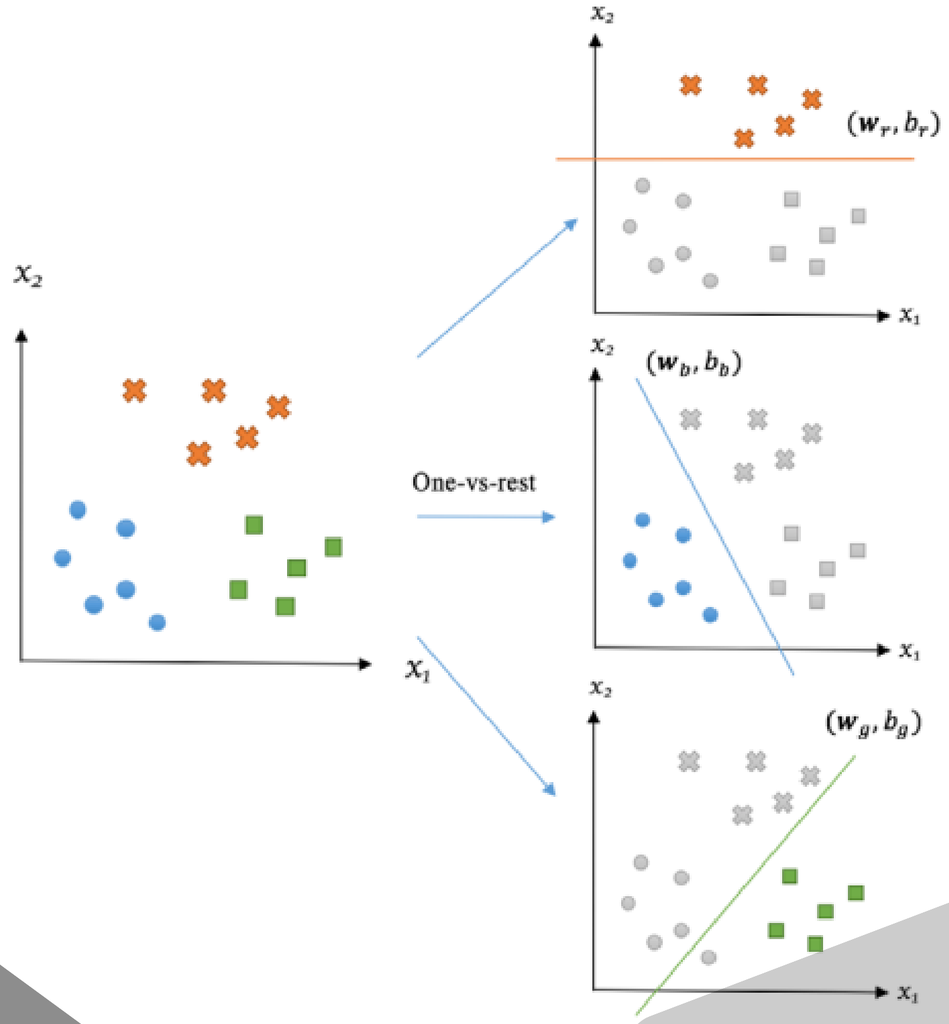
SEGREGACIÓN

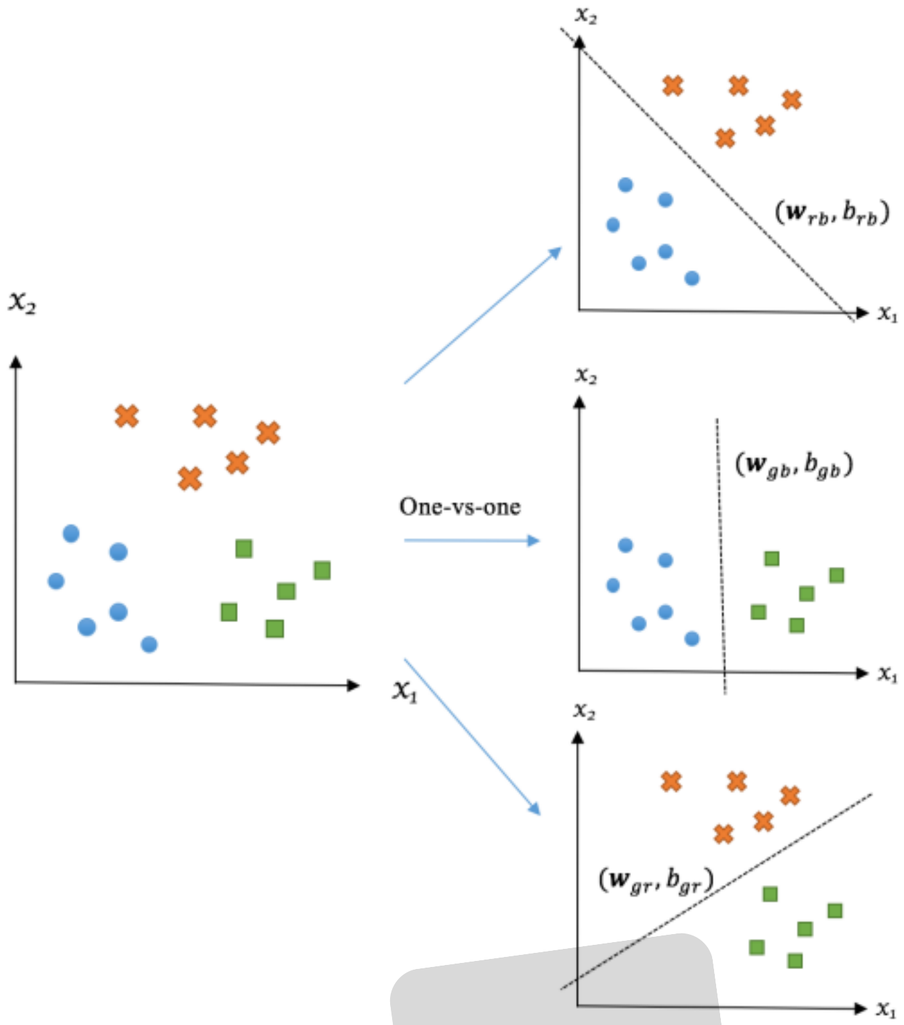


IMPLEMENTACIÓN DE SVC

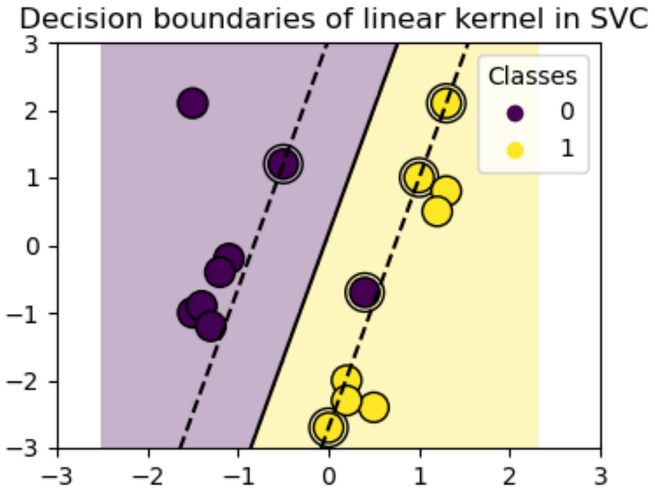
```
from sklearn.svm import SVC

clf = SVC(kernel='linear', C=segregacion, random_state=42)
```

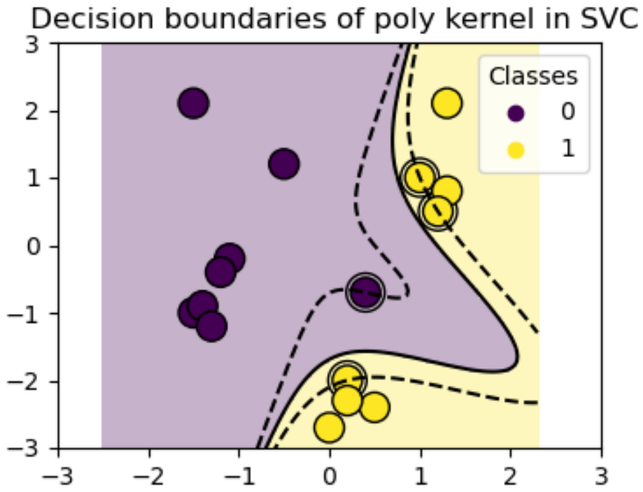
CLASIFICACIÓN MULTICLASE

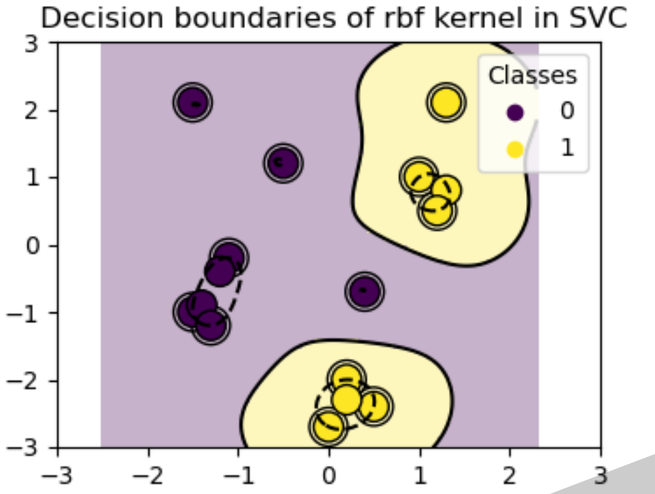


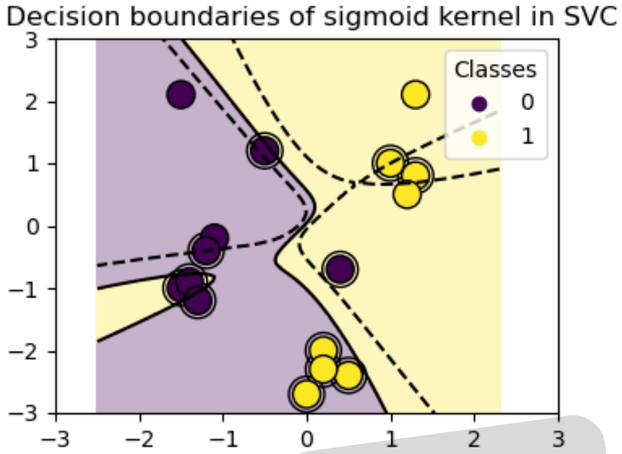


PROBLEMAS NO LINEALES

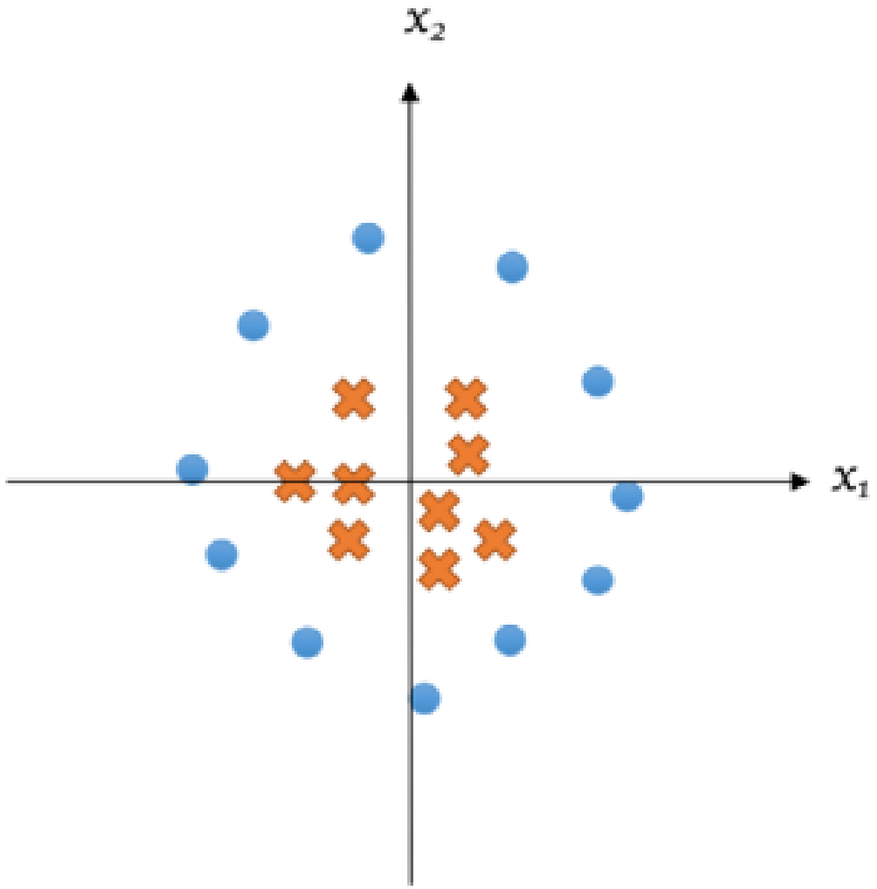








PROBLEMAS NO LINEALES



LINEAL

Linealmente separables

1ra opción con muchas features e instancias

1ra opción con más features que instancias

1ra opción con más instancias que features

RADIAL

No lineales

2da opción con muchas features e instancias

2da opción con más features que instancias

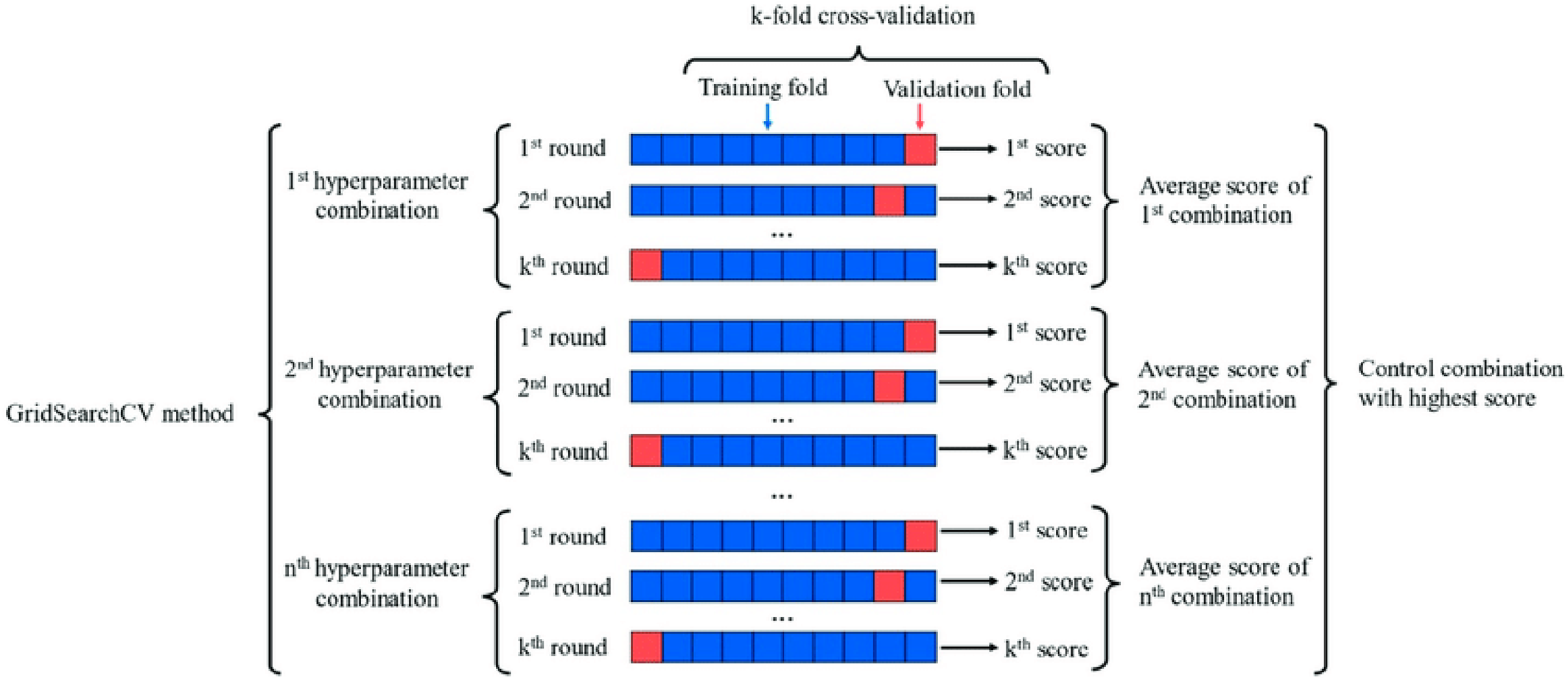
2da opción con más instancias que features

HIPERPARÁMETROS

Conjunto de configuraciones que el modelo usa para “aprender”

```
from sklearn.svm import SVC

clf = SVC(kernel='linear', C=segregacion, random_state=42)
```

SELECCIÓN CON GRIDSEARCH



IMPLEMENTACIÓN DE GRIDSEARCHCV

```
from sklearn.model_selection import GridSearchCV

# Seleccionamos los posibles hiperparámetros a probar para entrenar
nuestro modelo
parameters = {'C': [0.1, 1, 10],
              'gamma': [1e-07, 1e-08, 1e-06],
              'kernel' : ['rbf', 'linear'] }

# Creamos una búsqueda en permutación de hiperparámetros
grid_search = GridSearchCV(clf, parameters, n_jobs=-1, cv=5)

# entrenamos con los distintas permutaciones de hiperparámetros
grid_search.fit(X_train, Y_train)
```

MÉTRICAS DE EVALUACIÓN

	Predecido		
Esperado		F	V
	F	VN	FN
	V	FP	VP

PRECISIÓN

Un valor alto indica que hay pocos falsos positivos

SENSIBILIDAD

Un valor alto indica que hay pocos falsos negativos

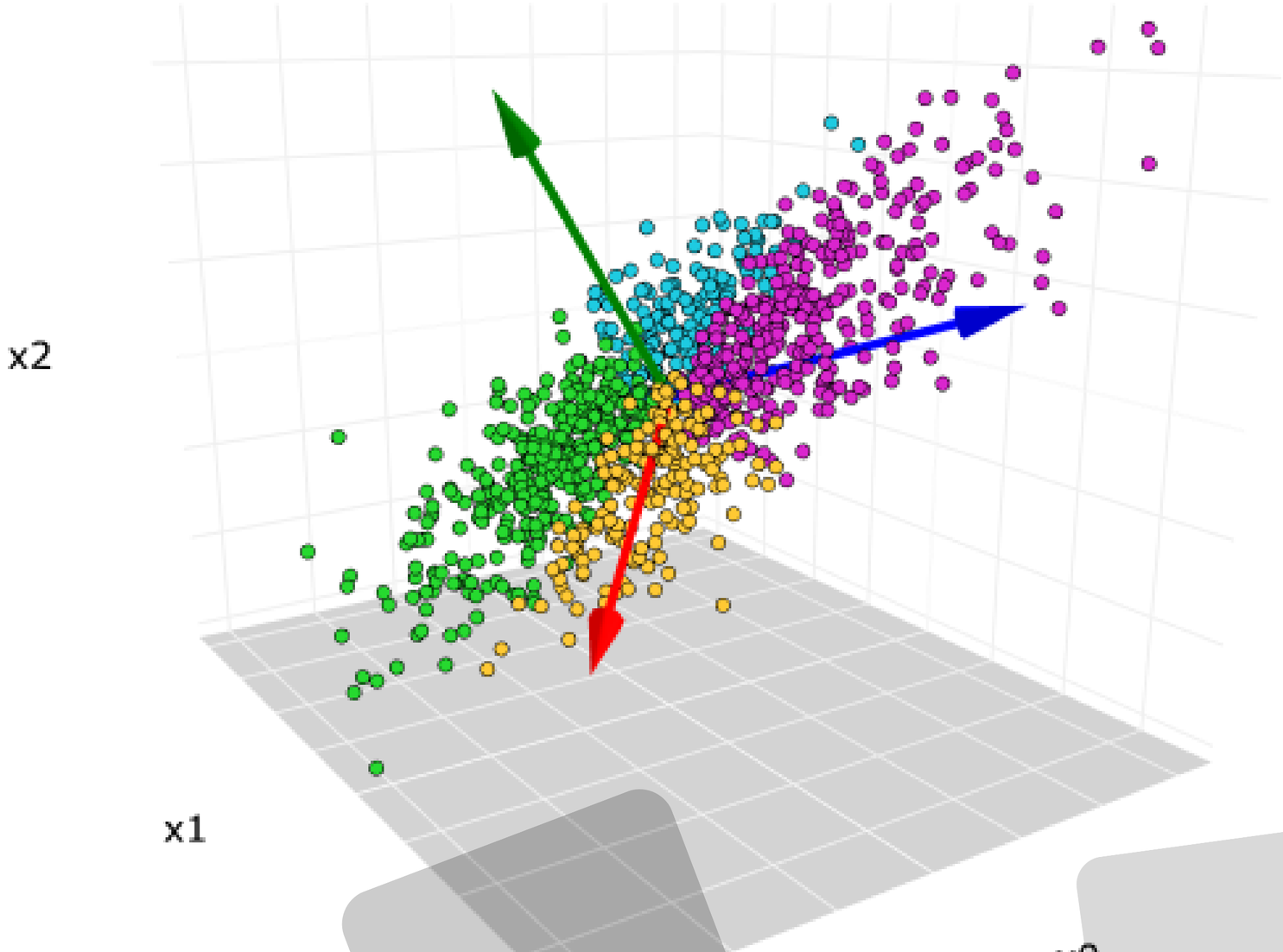
F1

Un valor alto indica equilibrio entre precision y recall.

SOPORTE

Muestras por clase

REDUCCIÓN DE DIMENSIONALIDAD



IMPLEMENTACIÓN DE PCA

```
from sklearn.decomposition import PCA

pca = PCA(n_components=numero_de_componentes,
whiten=eliminar_no_corrlacionados, random_state=42)
```

IMPLEMENTACIÓN DE PIPELINES

```
from sklearn.pipeline import Pipeline

model = Pipeline([('nombre_modelo_1', modelo1),
                  ('nombre_modelo_2', modelo2)])
```

CONCLUSIONES

SVM

Los vectores de soporte son una herramienta muy poderosa para conjuntos de datos con muchas dimensiones

TUNING

Con GridSearchCV Podemos encontrar los mejores hiperparámetros para nuestros modelos

PCA

La reducción de dimensionalidad de nuestros problemas permite optimizar el entrenamiento sin perder calidad en predicciones.