Día 2 - 16/11/21 (Martes)

Aprendizaje

Supervisado

Lic. Ronaldo Armando Canizales Turcios

UNIVERSIDAD DE GRANADA

Universidad Centroamericana José Simeón Cañas

Agenda Día 2

Bloque A

- Aprendizaje supervisado: modelos y conceptos.

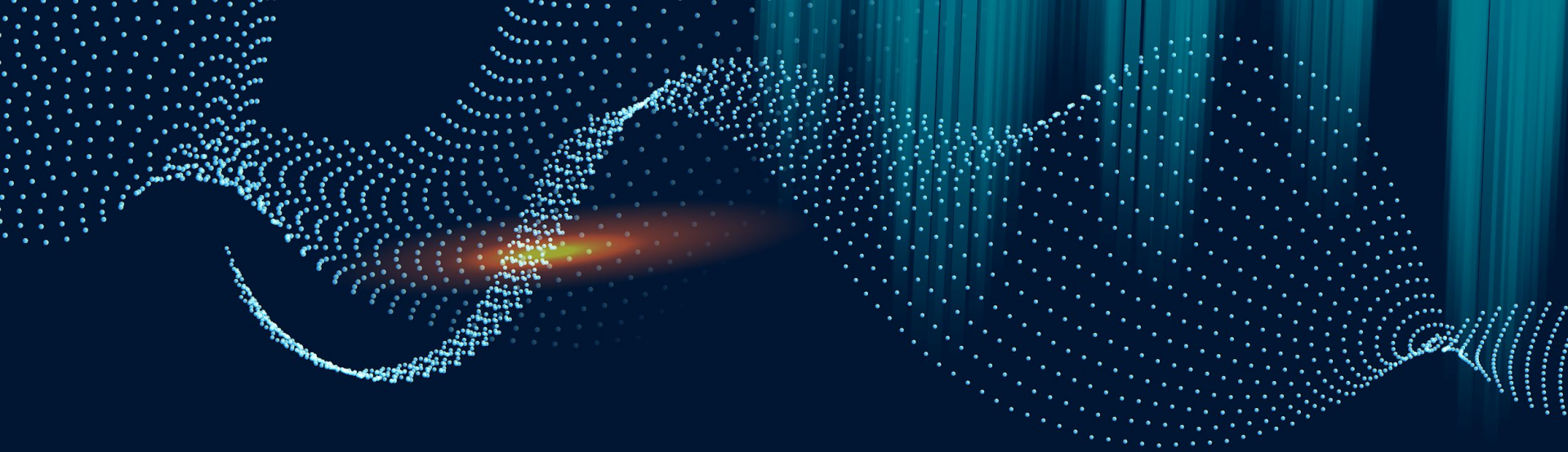
- Estimación de propinas mediante varios modelos de ML: regresión lineal, k-vecinos más cercanos, bosques aleatorios y máquinas de soporte vectorial.

Bloque B

- Conceptos estadísticos útiles: matrices de confusión, curvas ROC y Lift.

- Aprendizaje supervisado: predicción de abandono de clientes.

- ¡Manos a la obra!. **[Práctica]**



01 Aprendizaje supervisado

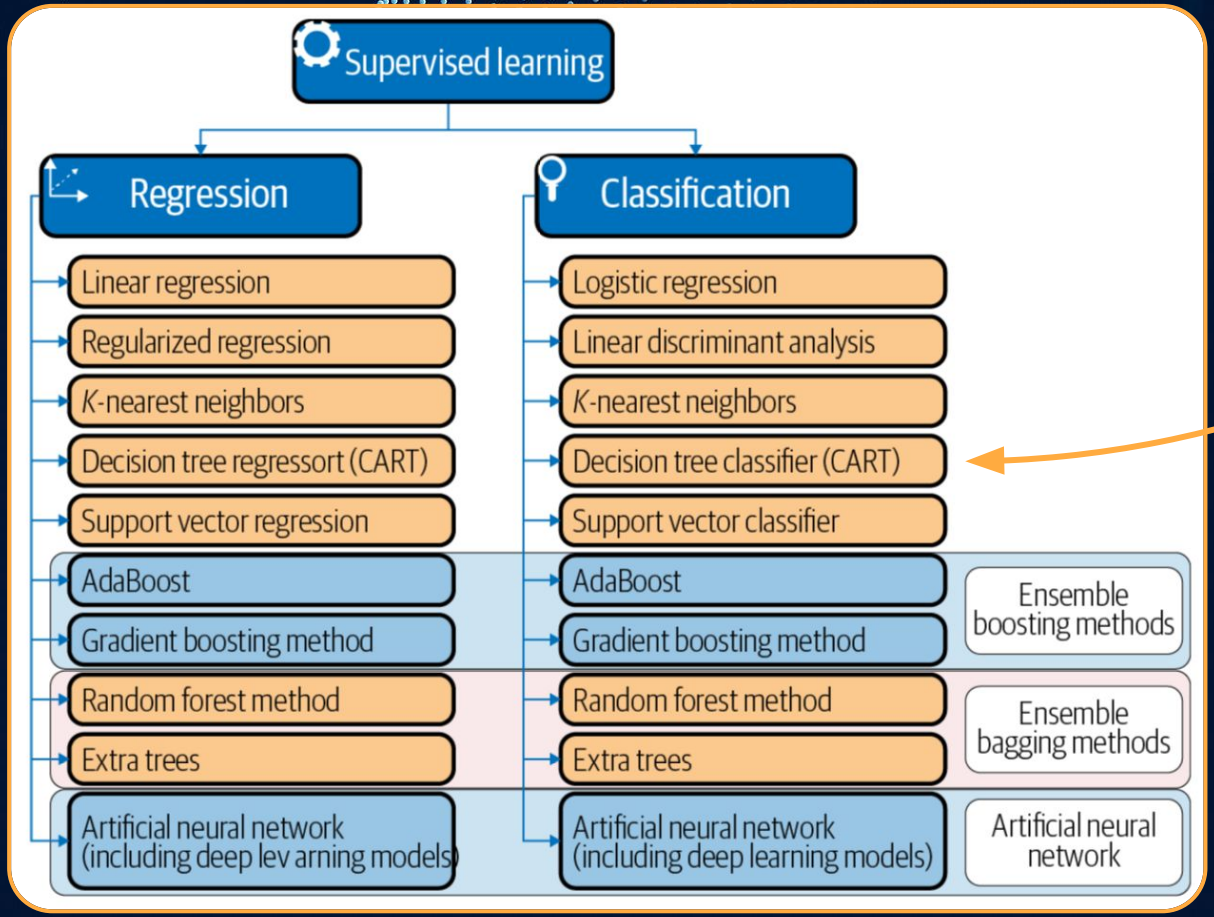
Modelos y conceptos

Definición

- El aprendizaje supervisado es un tipo de ML.
- En primer instancia, se provee un conjunto de datos llamados “de entrenamiento” a los algoritmos en cuestión.
- Partiendo de un conjunto masivo de datos, el algoritmo aprenderá una regla útil para predecir las etiquetas de nuevos datos.
- En otras palabras, los algoritmos supervisados son provistos de datos históricos y luego se les pregunta cuál es la regla que tiene el mejor poder predictivo.

Tipos

- De regresión: predice salidas reales (infinito número de soluciones) en base en sus variables de entrada (tipo variado).
- De clasificación: identifican a cuál categoría (conjunto finito y usualmente pequeño) es más probable que pertenezca cierta entrada (tipos de variables variados).



Algunos pueden usarse para ambos fines sin mayor diferencia o trabajo añadido.

Modelos

Regresión lineal (mínimos cuadrados de toda la vida)

Hiperparámetros

$$y = \beta_0 + \beta_1 x_1 + \dots + \beta_i x_i$$

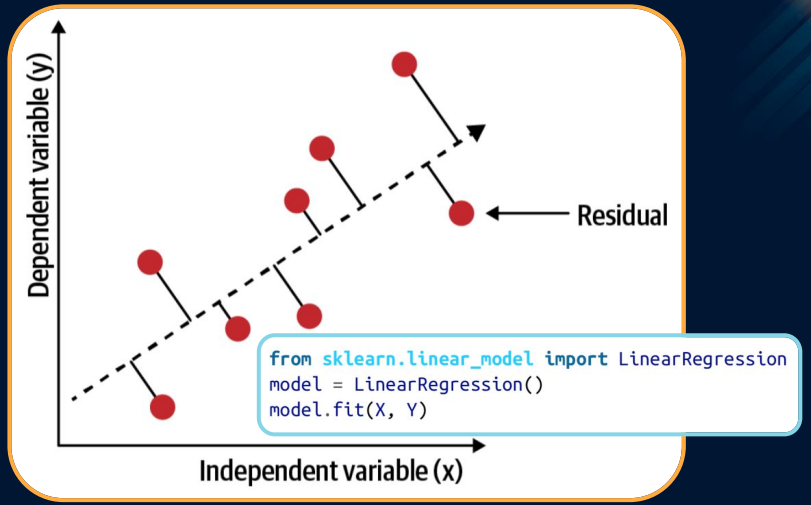
De los modelos más conocidos en estadística y en aprendizaje de máquina.

Se asume una relación lineal entre las variables de entrada y la variable de salida.

El objetivo es predecir el valor “y” de un valor “x” no visto anteriormente, con el menor error posible.

Ventajas: sencillo de interpretar.

Desventajas: necesita que haya una relación lineal entre las entradas y la salida. Puede ser afectada por el sobreajuste.



Regresión regularizada

Regresión Lasso (Regularización L1): añade la suma del valor absoluto de los coeficientes en la función de costo de la regresión lineal. Esto puede eliminar (disminuir hasta ser casi cero) algunas características, de forma que reduce la complejidad del modelo.

Regresión Ridge (Regularización L2): añade la suma del cuadrado de los coeficientes en la función de costo de la regresión lineal. Esto evita que los valores sean muy grandes, disminuyendo la varianza.

Red elástica: combina las dos anteriores mediante un parámetro, α (Ridge = 0, Lasso = 1).

$$RSS = \sum_{i=1}^{n} \left(y_i - \beta_0 - \sum_{j=1}^{n} \beta_j x_{ij}\right)^2$$

$$CostFunction = RSS + \lambda * \Sigma_{j=1}^{p} |\beta_j|$$

```
from sklearn.linear_model import Lasso
model = Lasso()
model.fit(X, Y)
```

$$CostFunction = RSS + \lambda * \Sigma_{j=1}^{p} \beta_j^2$$

```
from sklearn.linear_model import Ridge
model = Ridge()
model.fit(X, Y)
```

$$CostFunction = RSS +$$

$$\lambda * ((1 - \alpha) / 2 * \Sigma_{j=1}^{p} \beta_j^2 + \alpha * \Sigma_{j=1}^{p} |\beta_j|)$$

```
from sklearn.linear_model import ElasticNet
model = ElasticNet()
model.fit(X, Y)
```

Regresión logística

$$y = \frac{\exp\left(\beta_0 + \beta_1 x_1 + ... + \beta_i x_1\right)}{1 + \exp\left(\beta_0 + \beta_1 x_1 + ... + \beta_i x_1\right)}$$

```
from sklearn.linear_model import LogisticRegression
model = LogisticRegression()
model.fit(X, Y)
```

Hiperparámetros:

- Regularización
- Fuerza de la Reg.

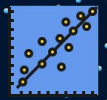
De los modelos más utilizados para clasificación. Se asegura que la salida pertenecerá al rango entre 0 y 1 (lo esperado para una probabilidad).

Si un modelo se entrenase solo con datos cuya salida sea 0 o 1, no será extraño obtener valores negativos o mayores a la unidad (valor que se desea evitar).

Ventajas: sencillo de interpretar.

Desventajas: puede ser afectada por el sobreajuste. No es la mejor opción para aprender relaciones muy complejas.

Máquinas de soporte vectorial



```python
from sklearn.svm import SVR
model = SVR()
model.fit(X, Y)
```

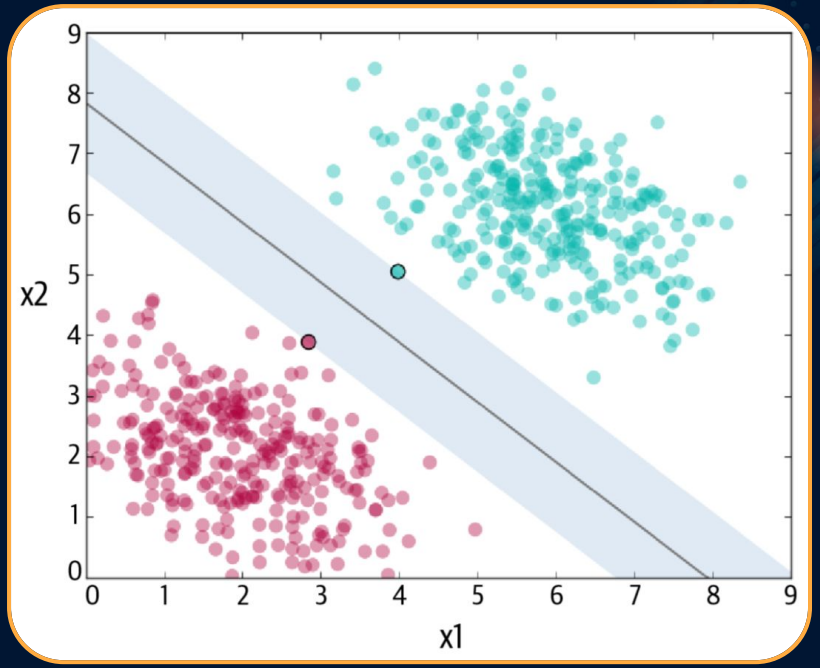
```python
from sklearn.svm import SVC
model = SVC()
model.fit(X, Y)
```

Su propósito es encontrar un hiperplano que maximice la distancia entre las instancias de entrenamiento.

En la práctica, los datos casi nunca pueden ser separados perfectamente por un hiperplano (como en la figura), así que se hace necesario flexibilizar esa regla: permitir que algunos puntos violen la frontera.

Ventajas: es robusto ante el sobreajuste, especialmente en altas dimensiones. Es muy útil para casos no-lineales

Desventajas: no es muy intuitivo. Requiere gran cantidad de memoria cuando se utiliza en conjuntos de datos grandes.

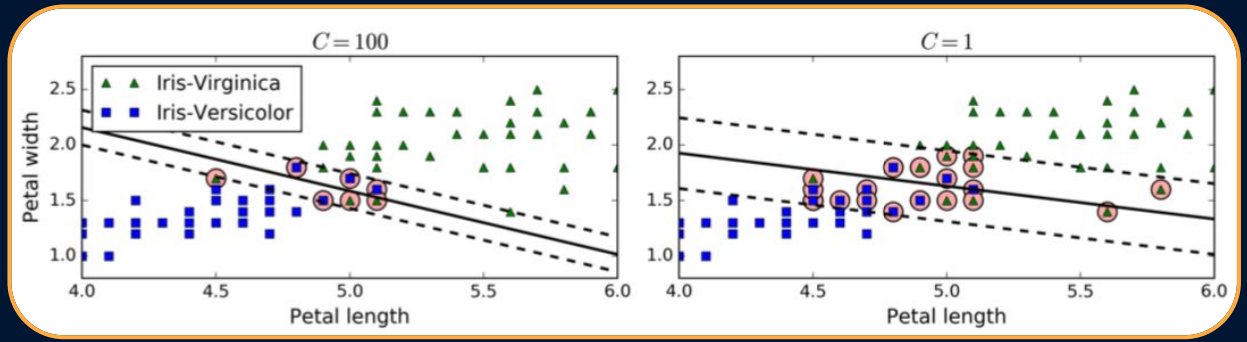


Hiperparámetros de las máquinas de soporte vectorial

Penalty (C en sklearn)

- Se le especifica al algoritmo qué tan estrictos debemos ser para evitar que hayan violaciones a la regla.
- Valores grandes crearán un hiperplano con un margen más pequeño.
- Valores pequeños conllevarán a márgenes más grandes, y de esa forma se reducirá el sobreajuste.

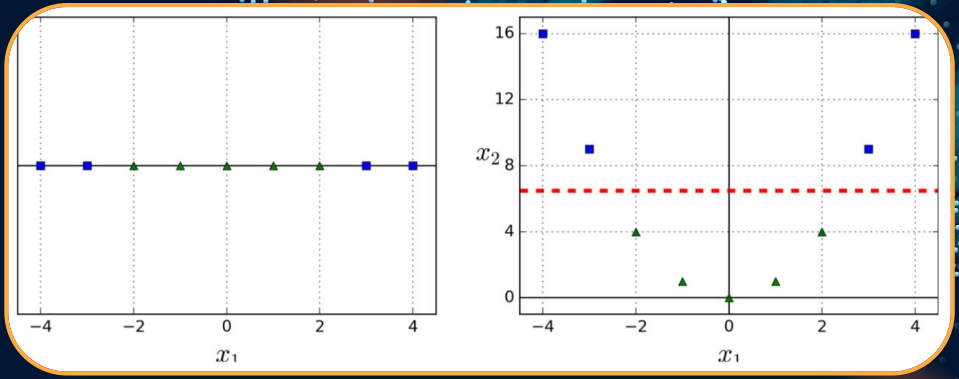
```
SVC(kernel="linear", C=1)
```



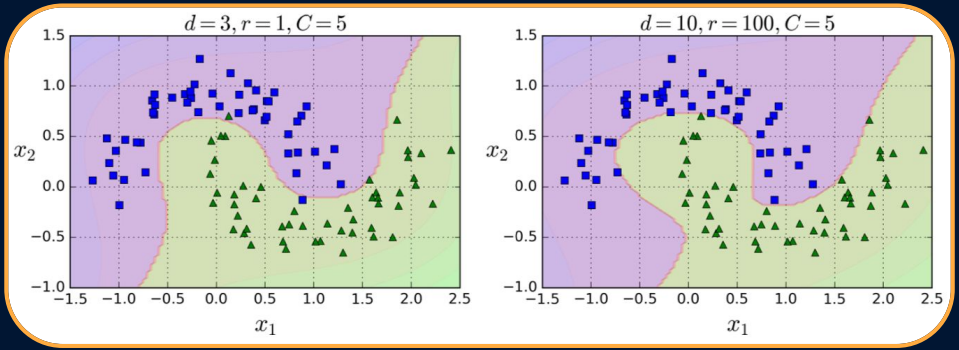
Hiperparámetros de las MSV

Kernels (kernel en sklearn)

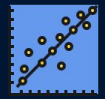
- Existen casos en que además de flexibilizar la regla, es necesario agregar una dimensión extra a los datos.
- Se controla la manera en que las variables de entrada serán proyectadas.
- Existen varios, pero: lineal, polinómico y RBF (Gaussian Radial Basis Function) son los más comunes.



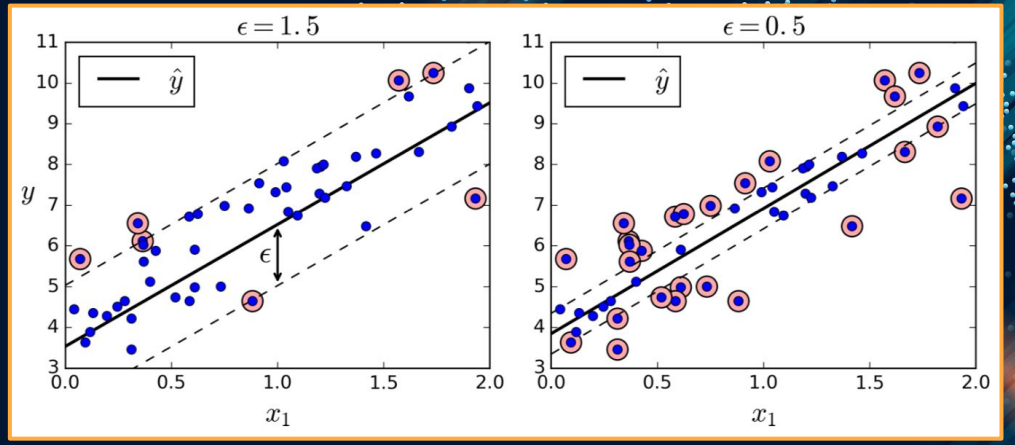

```python
from sklearn.svm import SVC
poly_kernel_svm_clf = Pipeline((
        ("scaler", StandardScaler()),
        ("svm_clf", SVC(kernel="poly", degree=3, coef0=1, C=5))
    ))
poly_kernel_svm_clf.fit(X, y)
```



Hiperparámetros de las MSV



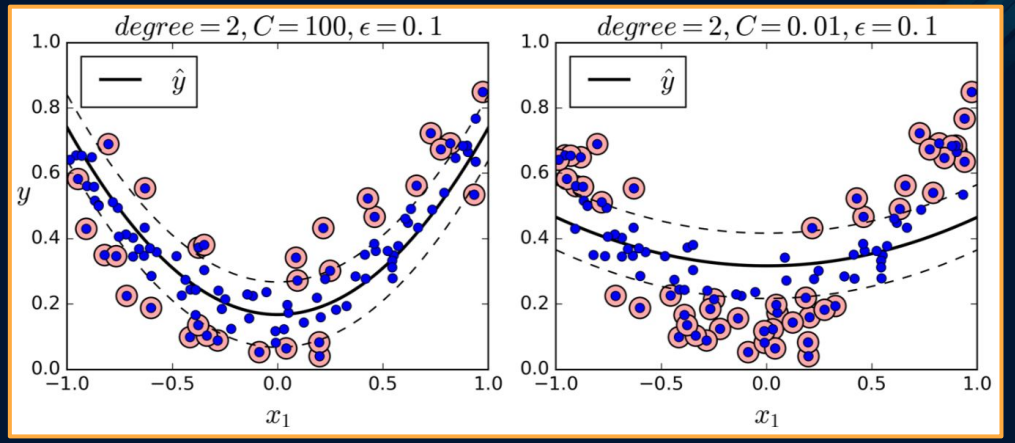
```
from sklearn.svm import LinearSVR

svm_reg = LinearSVR(epsilon=1.5)
svm_reg.fit(X, y)
```



Epsilon (epsilon en sklearn)

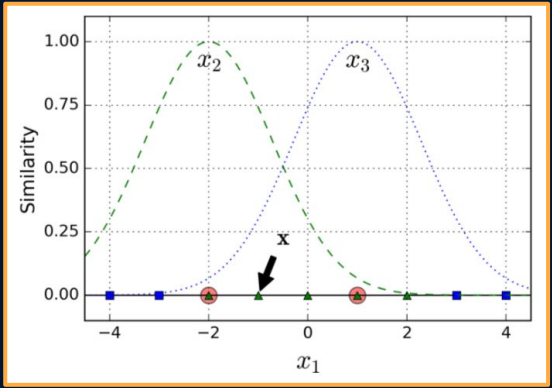
- Intenta colocar tantas instancias como sea posible dentro de los márgenes.

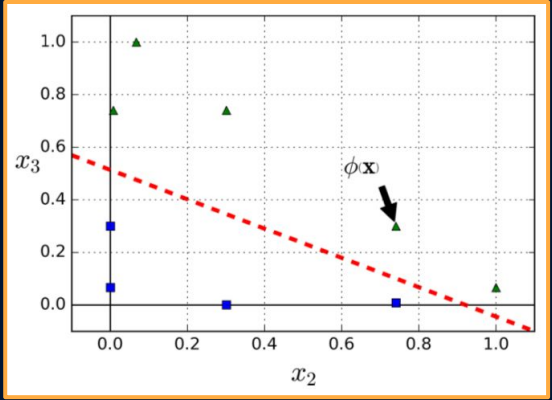
```
from sklearn.svm import SVR

svm_poly_reg = SVR(kernel="poly", degree=2, C=100, epsilon=0.1)
svm_poly_reg.fit(X, y)
```

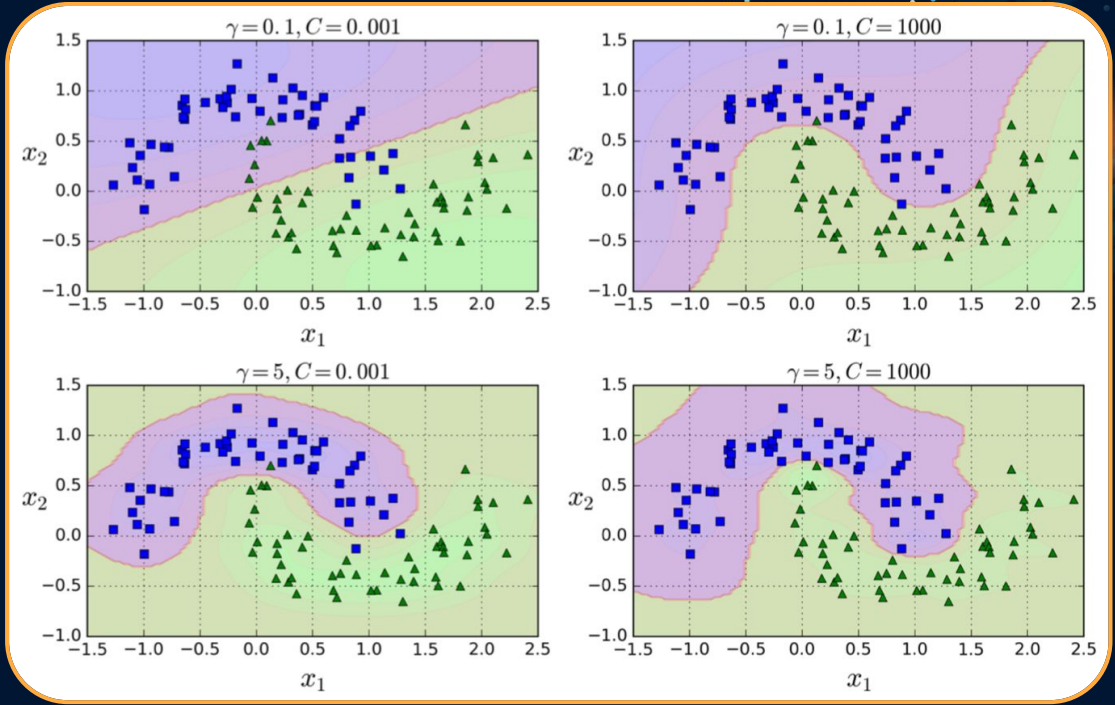


Hiperparámetros de las MSV

```
rbf_kernel_svm_clf = Pipeline((
        ("scaler", StandardScaler()),
        ("svm_clf", SVC(kernel="rbf", gamma=5, C=0.001))
    ))
rbf_kernel_svm_clf.fit(X, y)
```



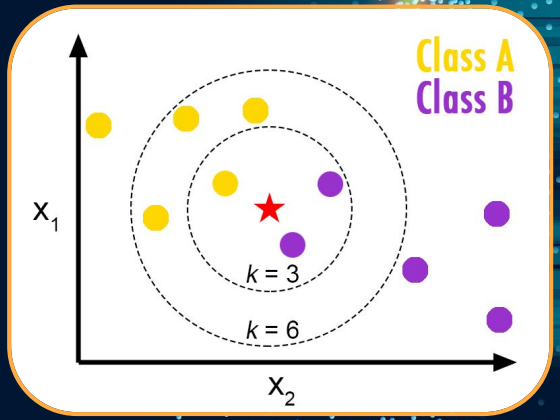




K-vecinos más cercanos

Se puede resumir en los siguientes pasos:

- Elegir el número de vecinos y una distancia.
- Encontrar los K-vecinos más cercanos a la muestra que queremos clasificar.
- Asignar una etiqueta en base a votación por mayoría.




```
from sklearn.neighbors import KNeighborsClassifier
model = KNeighborsClassifier()
model.fit(X, Y)
```

```
from sklearn.neighbors import KNeighborsRegressor
model = KNeighborsRegressor()
model.fit(X, Y)
```

Hiperparámetros en K-vecinos más cercanos

Cantidad de vecinos (n_neighbors en sklearn)

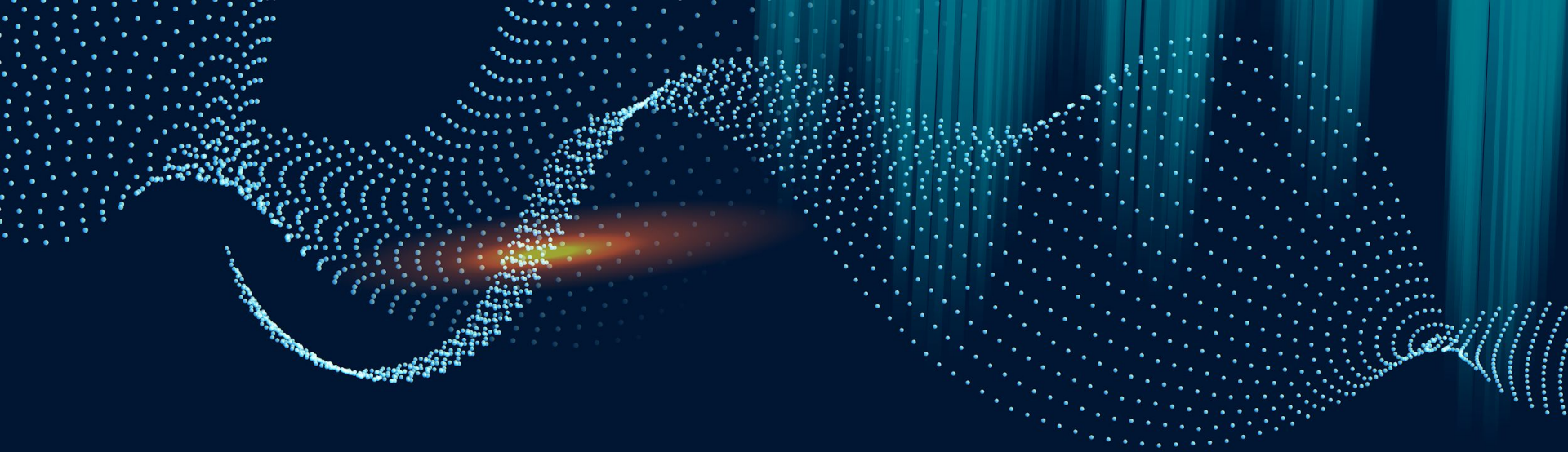
- El hiperparámetro más importante.
- Valores comunes entre 1 y 20.

Distancia (metric en sklearn)

- Se utilizan distintas métricas en lo referente a la cuantificación de la distancia.
- Las más comunes son euclidiana y Manhattan.

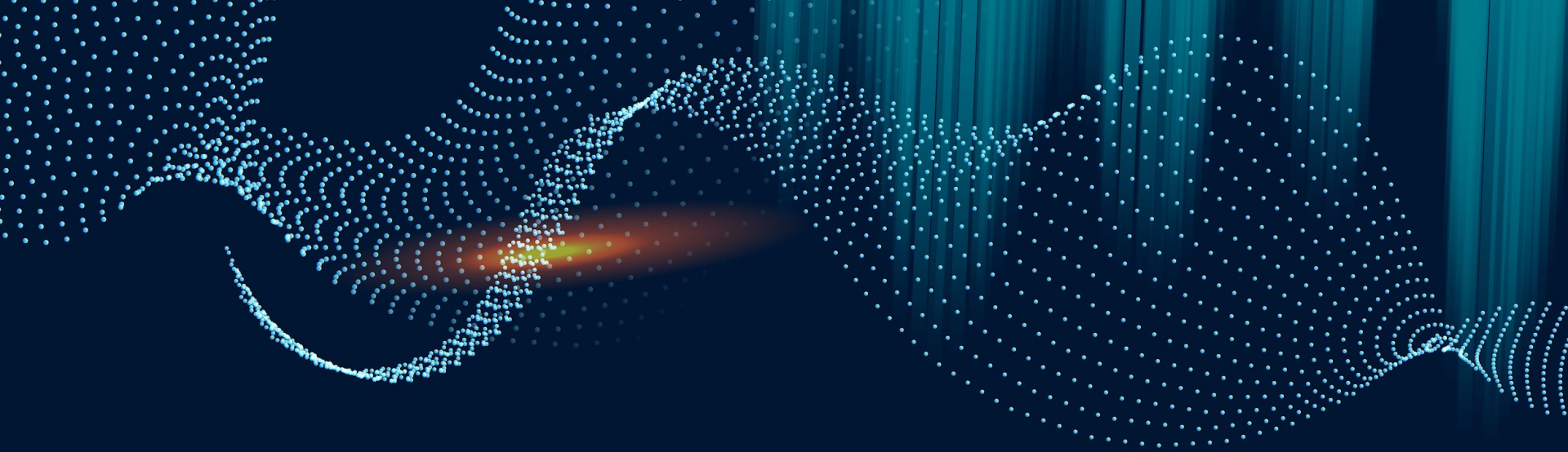
$$d(a, b) = \sqrt{\sum_{i=1}^{n} (a_i - b_i)^2}.$$

$$d(a, b) = \sum_{i=1}^{n} \left| a_i - b_i \right|$$



02 Estimación de propinas

Regresión mediante varios modelos de ML



03 Evaluación de modelos

Métricas y técnicas más utilizadas

Métricas y técnicas a utilizar para evaluar un modelo de Aprendizaje de Máquina

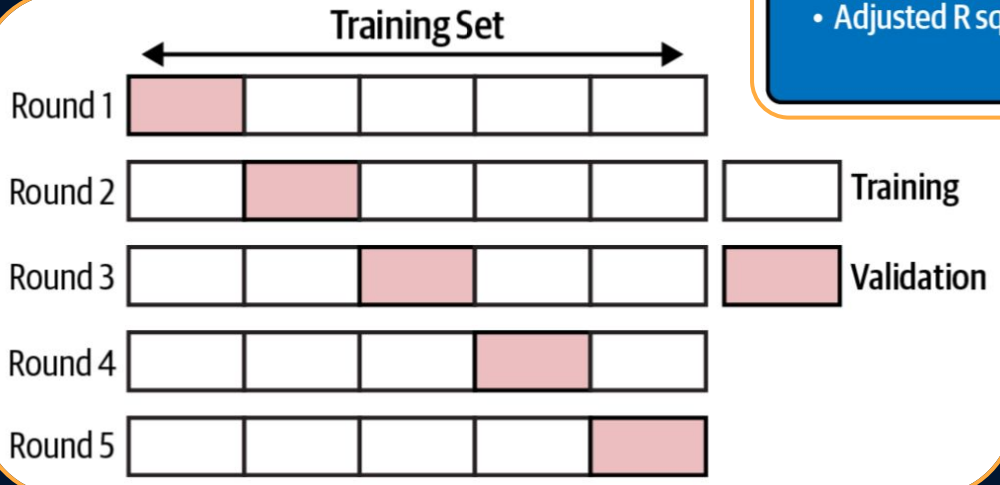
Métricas más comunes

Regression

- Mean absolute error (MAE)
- Mean squared error (MSE)
- R squared (R^2)
- Adjusted R squared (Adj-R^2)

Classification

- Accuracy
- Precision
- Recall
- Area under curve (AUC)
- Confusion matrix

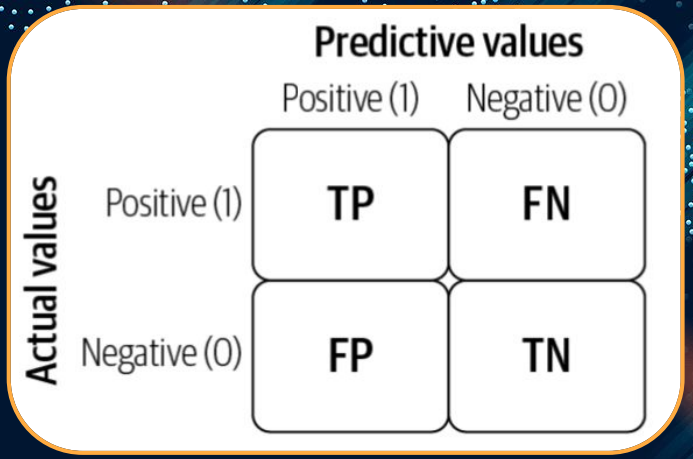


Validación cruzada

- Ventaja: encontrar el modelo que pueda generalizar mejor los datos (menos sobreajuste).
- Desventaja: tiempo y capacidad computacional.

Matriz de confusión y otros indicadores derivados

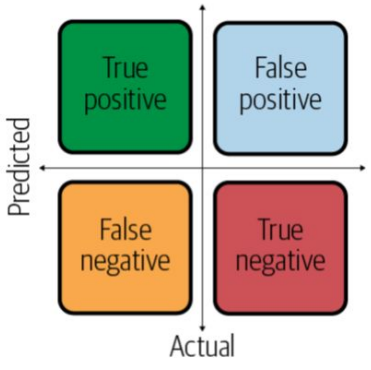
- En algunos escenarios será de mayor importancia un indicador que otro.
- Cuando las clases están balanceadas, entonces el Accuracy es muy útil, caso contrario no es de fiar.



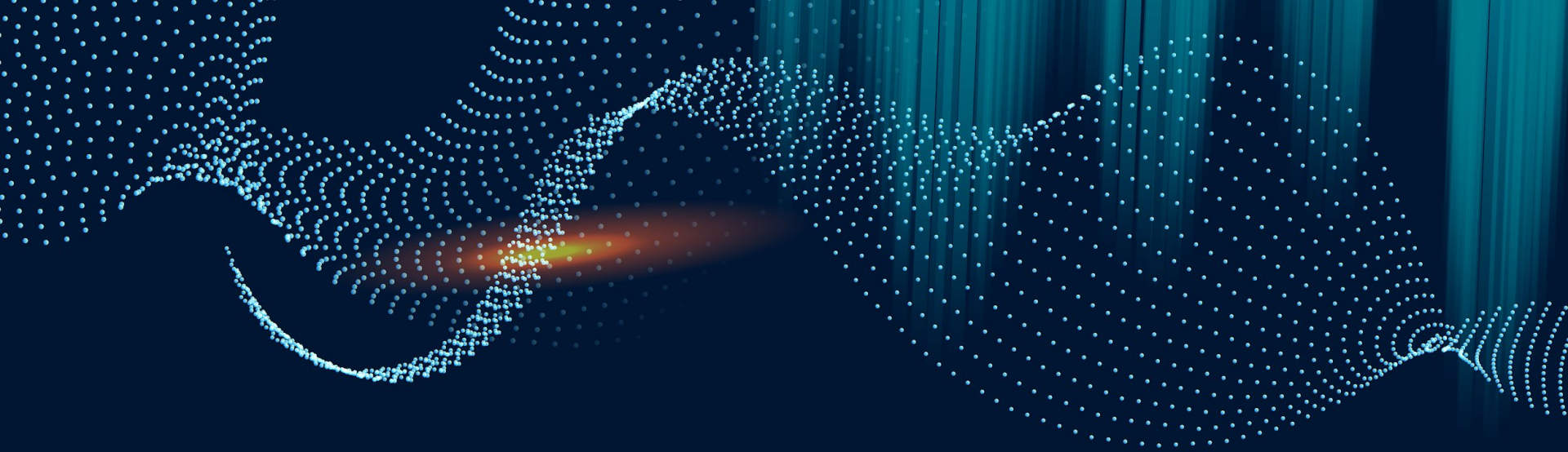
$$\text{Precision} = \frac{\text{True positive}}{\text{Actual results}} \quad \text{or} \quad \frac{\text{True positive}}{\text{True positive} + \text{False positive}}$$

$$\text{Recall} = \frac{\text{True positive}}{\text{Predictive results}} \quad \text{or} \quad \frac{\text{True positive}}{\text{True positive} + \text{False negative}}$$

$$\text{Accuracy} = \frac{\text{True positive} + \text{True negative}}{\text{Total}}$$



Cuando se trate de más de dos clases, las métricas deberán calcularse para cada una de ellas.



04 Predicción de abandono de clientes

Clasificación mediante varios modelos de ML

05 Aprendizaje supervisado

¡Manos a la obra! **[Práctica]**

¿Consultas,

dudas o comentarios?

Muchas gracias por su asistencia y atención

UNIVERSIDAD DE GRANADA

Universidad Centroamericana José Simeón Cañas