

Módulo 5  
Clase 11

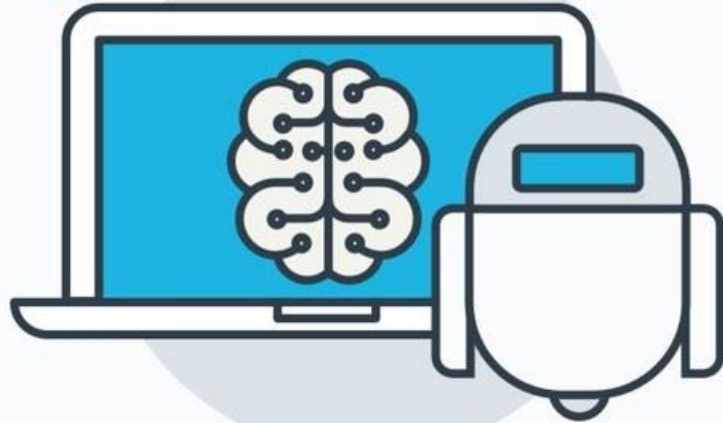
# Aprendizaje de Máquina Supervisado



## Optimización del Modelo

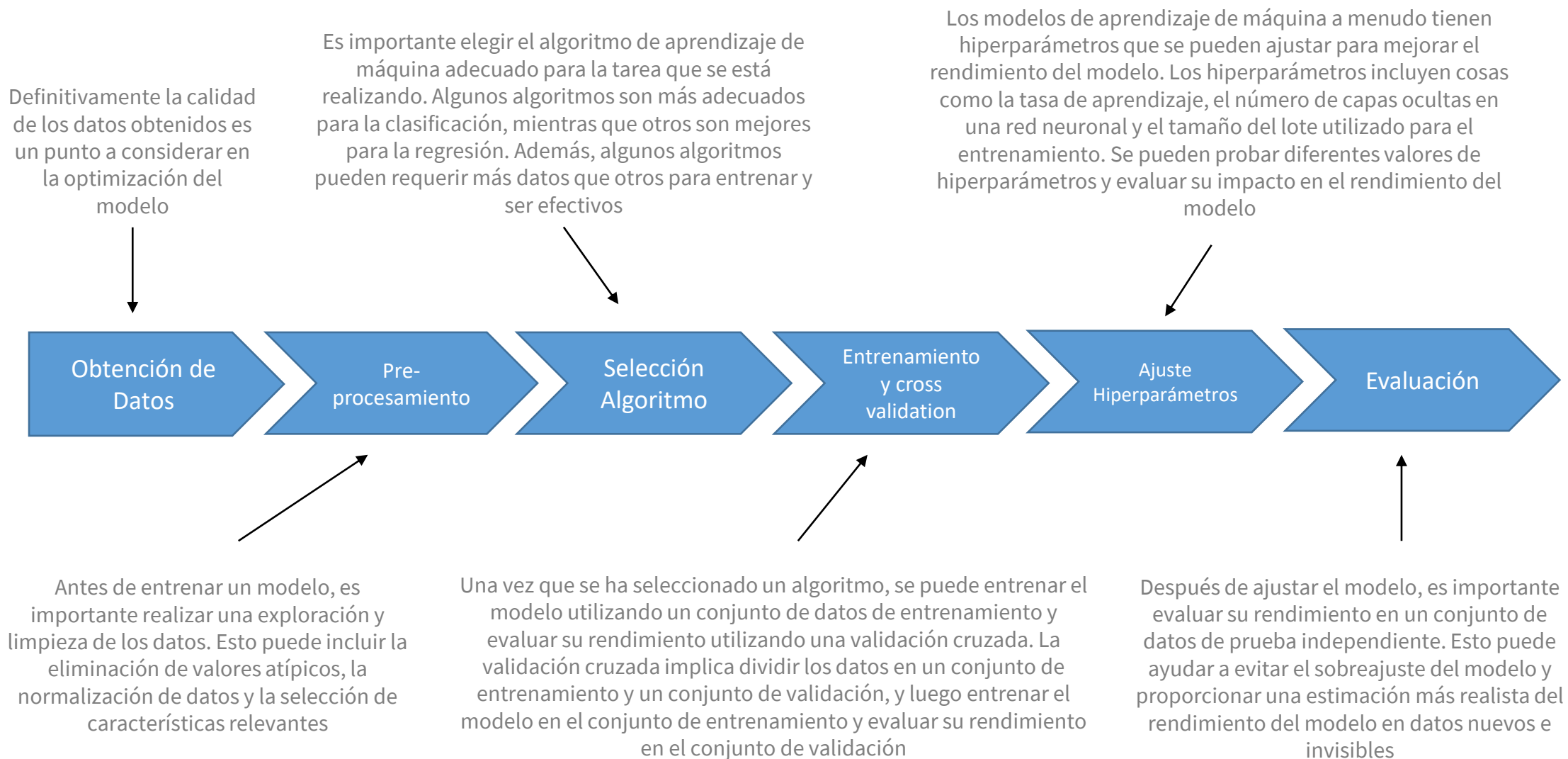
# Optimización del modelo

La **optimización de un modelo de aprendizaje de máquina** es un proceso importante para mejorar su rendimiento y precisión en la tarea que se está realizando. En general, la optimización de un modelo de aprendizaje de máquina es un proceso iterativo que implica probar diferentes enfoques y ajustar parámetros hasta que se alcance el rendimiento deseado. Aquí hay algunos pasos comunes para optimizar un modelo de aprendizaje de máquina:





# Optimización del modelo

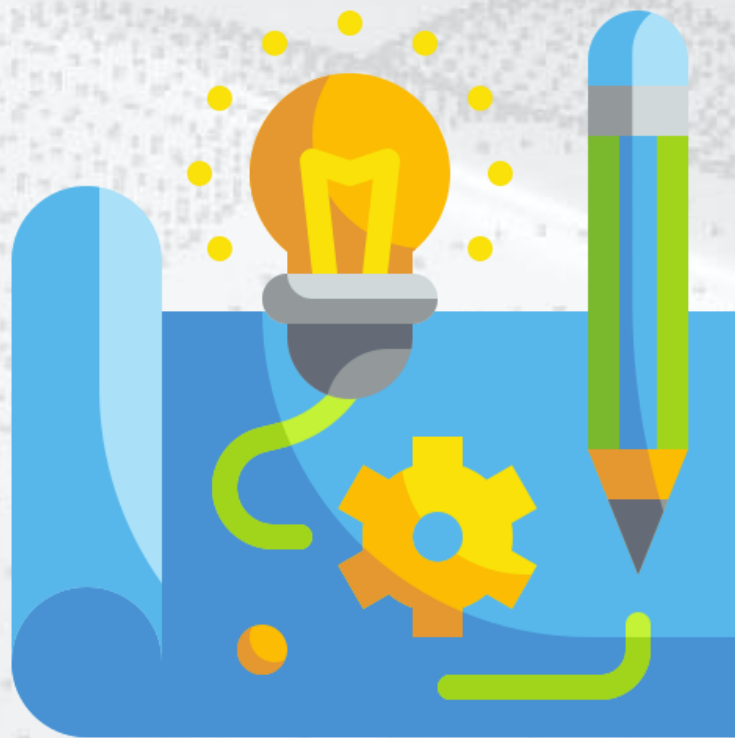


# Optimización del modelo

Los modelos de ML podrían mejorar realizando alguna de estas acciones:

- Agregando más ejemplos al dataset
- Incorporar nuevos features al modelo
- Haciendo Feature Engineering
- Optimizando el algoritmo (tunning hiperparámetros)





## Feature Engineering

# ¿Qué es Feature Engineering?

- Feature engineering (ingeniería de características) es el proceso de seleccionar y transformar las variables o características (features) de un conjunto de datos para mejorar el rendimiento y precisión de un modelo de aprendizaje de máquina.
- Las características son las variables que se utilizan para entrenar el modelo y hacer predicciones. El objetivo de la ingeniería de características es seleccionar las características más relevantes y útiles para la tarea en cuestión, y transformarlas en una forma que el modelo pueda usar de manera efectiva.
- La ingeniería de características es un paso crítico en el proceso de construcción de modelos de aprendizaje de máquina, ya que la calidad y relevancia de las características utilizadas en el modelo tienen un impacto significativo en su rendimiento y capacidad de generalización. Un buen feature engineering puede ayudar a reducir el sobreajuste (overfitting) del modelo y mejorar su capacidad para generalizar a nuevos datos



# ¿Qué es Feature Engineering?

Existen muchas técnicas de ingeniería de características que se pueden utilizar para mejorar el rendimiento de los modelos de aprendizaje de máquina. Algunas de las técnicas más comunes incluyen:

- **Creación de características derivadas:** a partir de las características existentes, se pueden crear nuevas características que contengan información adicional y sean más relevantes para la tarea en cuestión. Por ejemplo, si se tiene un conjunto de datos que incluye la fecha de nacimiento de una persona, se puede crear una nueva característica que indique la edad de la persona en lugar de la fecha de nacimiento.
- **Normalización de características:** esta técnica se utiliza para escalar las características a un rango común, lo que ayuda a asegurar que cada característica tenga un impacto similar en el modelo. Por ejemplo, si se tiene un conjunto de datos que incluye alturas de personas en centímetros, se puede normalizar las alturas dividiéndolas por la altura media en el conjunto de datos.
- **Codificación de características categóricas:** las características categóricas son aquellas que representan categorías, como el género o el tipo de vehículo. Estas características se pueden codificar en una forma numérica para que sean utilizadas por el modelo. Algunas técnicas de codificación de características categóricas incluyen la codificación one-hot y la codificación de frecuencia.



# ¿Qué es Feature Engineering?

- **Selección de características:** esta técnica implica seleccionar un subconjunto de características que sean las más relevantes para la tarea en cuestión. Esto puede ayudar a reducir el ruido en los datos y mejorar la capacidad de generalización del modelo. Algunas técnicas de selección de características incluyen análisis de correlación y selección de características basada en modelos.
- **Reducción de dimensionalidad:** esta técnica se utiliza para reducir la cantidad de características en un conjunto de datos al mismo tiempo que se conserva la mayor cantidad posible de información. Esto puede ayudar a reducir el tiempo de entrenamiento del modelo y mejorar la capacidad de generalización. Algunas técnicas de reducción de dimensionalidad incluyen PCA (análisis de componentes principales) y t-SNE (embeddings de vecinos estocásticos distribuidos en t).
- **Manejo de data no balanceada:** se refiere a las técnicas utilizadas para abordar la situación en la que hay una proporción desigual entre las clases en un conjunto de datos. Esto puede afectar la capacidad del modelo para aprender y generalizar a nuevos datos. Algunas técnicas comunes incluyen el submuestreo y el sobremuestreo de la clase minoritaria, y el uso de técnicas de clasificación basadas en costos o de ensemble para equilibrar la precisión en ambas clases.

En general, la selección y transformación adecuada de las características puede tener un gran impacto en el rendimiento y precisión de los modelos de aprendizaje de máquina. La ingeniería de características es un proceso iterativo y requiere una buena comprensión del conjunto de datos y de la tarea que se está realizando.

# Creación de características derivadas

Podemos utilizar el conocimiento del dominio (negocio) para la extracción de nuevos features desde la data cruda. Un buen feature engineering puede generar mejores resultados en los modelos de machine learning.

Survived	Pclass	Name	Sex	Age
0	3	Braund, Mr. Owen Harris	male	22.0
1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0
1	3	Heikkinen, Miss. Laina	female	26.0
1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0
0	3	Allen, Mr. William Henry	male	35.0



Nótese que el nombre del pasajero viene en un formato estándar, se podría intentar ingeniería de feature


# Creación de características derivadas

Como se puede observar, se ha creado una nueva columna título a partir del nombre. Esto produjo una mejora de algunos puntos porcentuales en el accuracy del modelo.

```
array([' Mr', ' Mrs', ' Miss', ' Master', ' Don', ' Rev', ' Dr', ' Mme',  
      ' Ms', ' Major', ' Lady', ' Sir', ' Mlle', ' Col', ' Capt',  
      ' the Countess', ' Jonkheer'], dtype=object)
```

Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked	Title
0	3	Sirayanian, Mr. Orsen	male	22.0	0	0	2669	7.2292	NaN	C	Mr
1	1	Duff Gordon, Sir. Cosmo Edmund ("Mr Morgan")	male	49.0	1	0	PC 17485	56.9292	A20	C	Sir
0	3	Sirota, Mr. Maurice	male	NaN	0	0	392092	8.0500	NaN	S	Mr

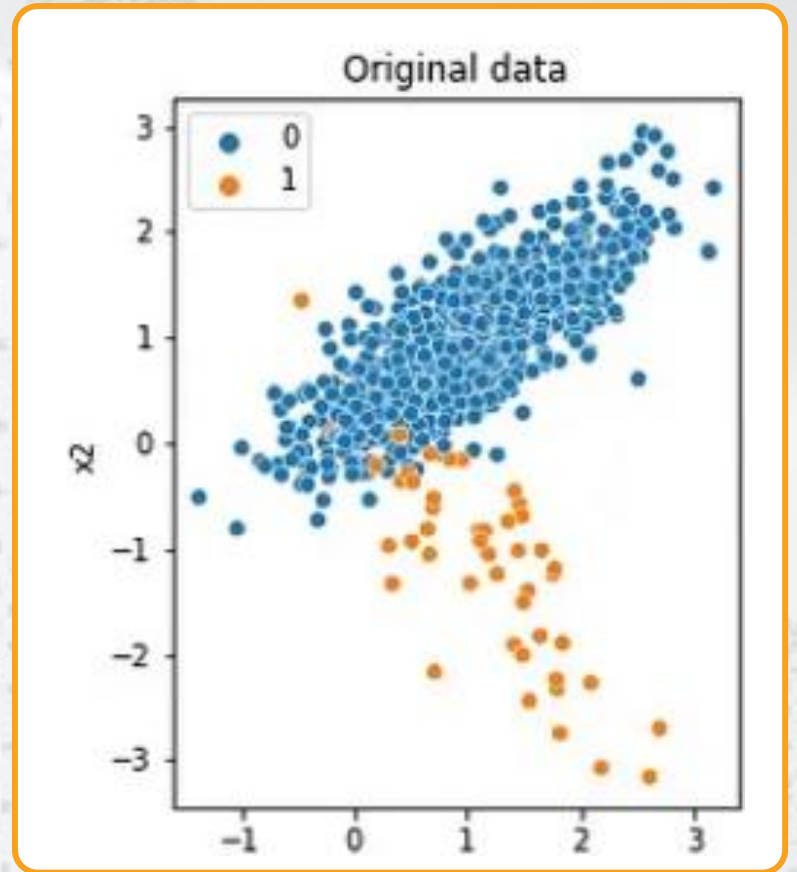
Nueva columna creada con feature engineering



# Manejo de datos desbalanceados

El desbalanceo de datos es un problema común en el aprendizaje automático donde los datos de una clase son significativamente menos representativos que los datos de otra clase en un conjunto de datos. Esto puede dificultar el entrenamiento de un modelo preciso, ya que el modelo puede tener una tendencia a favorecer la clase más representativa y, por lo tanto, no ser capaz de reconocer adecuadamente los casos de la clase minoritaria.

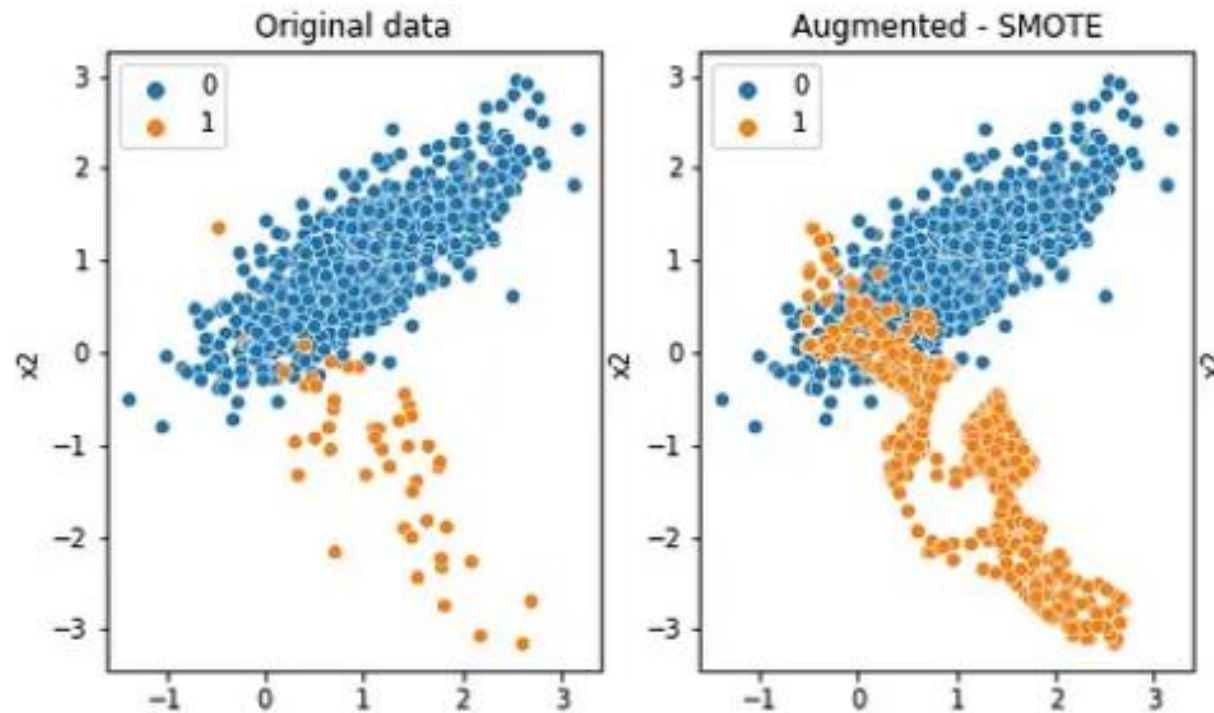
Por ejemplo, si un conjunto de datos tiene 1000 ejemplos, de los cuales 900 pertenecen a la clase A y solo 100 pertenecen a la clase B, el modelo puede tener dificultades para aprender correctamente la clase B debido a la falta de ejemplos representativos. Esto puede dar lugar a una clasificación sesgada y un bajo rendimiento en la detección de la clase minoritaria.





# Manejo de datos desbalanceados

Existen varios enfoques para enfrentar este tipo de situaciones, una de ellas es aplicar una técnica de Upsampling, es decir, aumentar la data de forma sintética para equiparar los ejemplos de cada clase. SMOTE (Synthetic Minority Over-Sampling Technique) es una técnica que crea nueva data a partir de la clase minoritaria usando combinaciones lineales de vectores de features.



# Manejo de datos desbalanceados

- La implementación en Python se puede llevar a cabo incorporando la librería imblearn (no viene en la distribución anaconda).
- Nótese cómo la clase minoritaria se equipará con la mayoritaria.

```
from imblearn.over_sampling import SMOTE  
from collections import Counter
```

[30] ✓ 0.0s

Python

```
Counter(y)
```

[29] ✓ 0.0s

Python

.. Counter({0: 64, 1: 17})

```
# transform the dataset  
oversample = SMOTE()  
X, y = oversample.fit_resample(X, y)
```

[32] ✓ 0.0s

Python

```
Counter(y)
```

[33] ✓ 0.0s

Python

.. Counter({0: 64, 1: 64})

# Manejo de datos desbalanceados

Después de volver a entrenar el algoritmo con data balanceada, comparamos el performance antes y después.

## Datos Desbalanceados

```
confusion_matrix(y_test,y_pred)
```

✓ 0.0s

Python

```
array([[17,  2],  
       [ 5,  1]], dtype=int64)
```

```
accuracy_score(y_test,y_pred)
```

✓ 0.0s

Python

0.72

## Datos Balanceados

```
confusion_matrix(y_test,y_pred)
```

✓ 0.0s

Python

```
array([[16,  2],  
       [ 1, 20]], dtype=int64)
```

```
accuracy_score(y_test,y_pred)
```

✓ 0.0s

Python

0.9230769230769231

Resultados obtenidos con set de datos Kyphosis, utilizando regresión logística como clasificador, sin optimización de hiperparámetros.



## Optimización de Hiperparámetros



# Optimización de Hiperparámetros

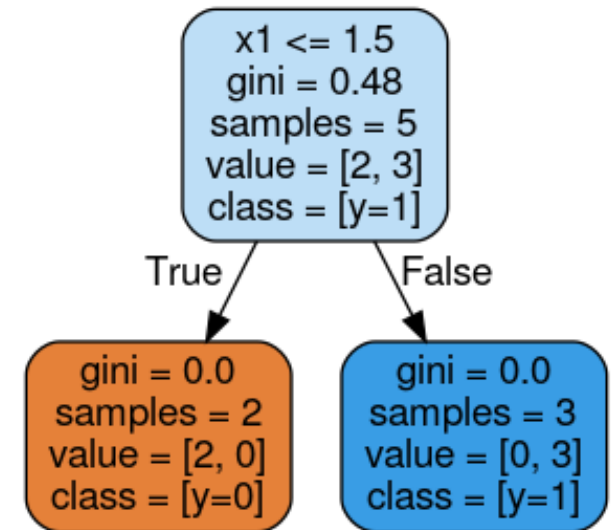
- En aprendizaje automático, un hiperparámetro es un parámetro que no se aprende directamente del conjunto de datos, sino que se establece antes del entrenamiento del modelo. Los hiperparámetros afectan la forma en que se aprenden los parámetros del modelo a partir de los datos y, por lo tanto, pueden tener un impacto significativo en el rendimiento del modelo.
- Los hiperparámetros se establecen antes del entrenamiento del modelo y se ajustan durante el proceso de entrenamiento para mejorar el rendimiento del modelo. Los ejemplos comunes de hiperparámetros incluyen la tasa de aprendizaje, el número de capas ocultas en una red neuronal, el número de árboles en un bosque aleatorio, la profundidad máxima de un árbol de decisión, entre otros.



# Hiperparámetros de árboles

Los principales hiperparámetros de un algoritmo de árbol de decisión incluyen:

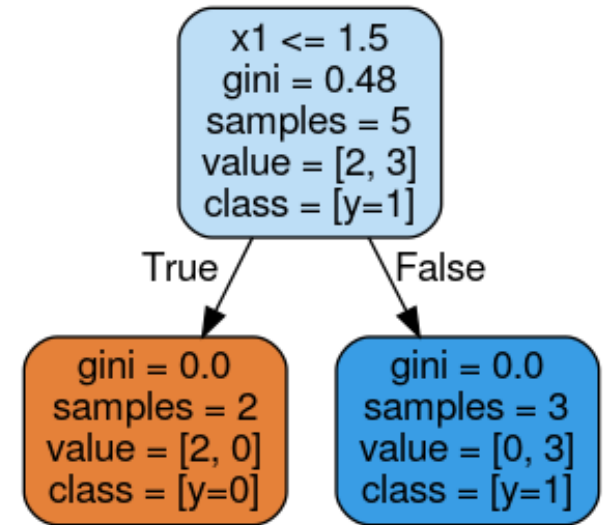
- **max\_depth:** Profundidad máxima del árbol. Limita la profundidad máxima del árbol. Un árbol demasiado profundo puede sobreajustarse al conjunto de datos de entrenamiento y no generalizar bien para nuevos datos.
- **min\_samples\_split:** Número mínimo de muestras requeridas para dividir un nodo. Establece el número mínimo de muestras que se requieren para dividir un nodo. Si el número de muestras en un nodo es menor que el valor establecido, el nodo no se dividirá.
- **min\_samples\_leaf:** Número mínimo de muestras requeridas para estar en una hoja. Establece el número mínimo de muestras que se requieren para formar una hoja. Si una división resulta en una hoja con un número menor de muestras que el valor establecido, la división se descarta.



# Hiperparámetros de Árboles

- **max\_features:** Número máximo de características a considerar para cada división. Limita el número máximo de características que se pueden considerar para cada división. Esto puede ayudar a reducir la complejidad del modelo y mejorar el rendimiento.
- **criterion:** Criterio de división. Establece el criterio utilizado para medir la calidad de una división. Los dos criterios comunes son "gini" y "entropía".

Estos son algunos de los hiperparámetros más comunes que se pueden ajustar en un algoritmo de árbol de decisión. Sin embargo, hay muchos otros hiperparámetros que también se pueden ajustar según el algoritmo específico que se esté utilizando. La optimización adecuada de los hiperparámetros puede ser crítica para el rendimiento y la precisión del modelo de árbol de decisión.



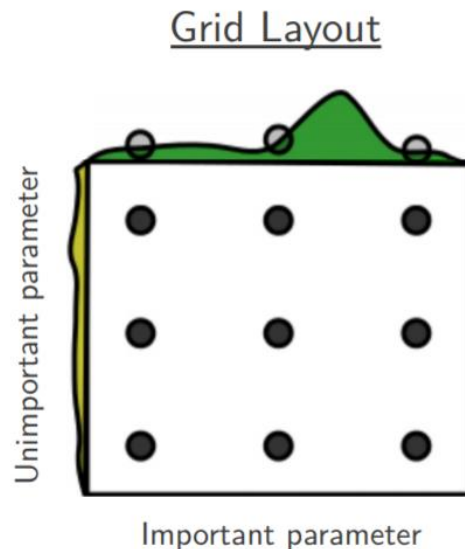
Puede obtener la información detallada desde la siguiente web:

<https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html>

# Métodos de Optimización

La optimización de hiperparámetros es un paso importante en el proceso de entrenamiento de modelos de aprendizaje automático, ya que determina los valores óptimos de los hiperparámetros que afectan el rendimiento del modelo. A continuación, se presentan algunos métodos comunes de optimización de hiperparámetros:

**Grid Search:** Búsqueda en cuadrícula. En este método, se define un conjunto de valores para cada hiperparámetro y se entrena y evalúa el modelo para todas las combinaciones posibles de valores de hiperparámetros. Es un enfoque exhaustivo pero puede ser computacionalmente costoso.



```
#Set the parameters by cross-validation
tuned_parameters = [{'max_depth': range(20,60),
                    'n_estimators': range(10,40),
                    'max_features': ['sqrt', 'log2', None]}]

clf = GridSearchCV(RandomForestRegressor(n_estimators=30), tuned_parameters, cv=5,
scoring='mean_squared_error')
clf.fit(self.X_train, self.y_train.ravel())

print "Best parameters set found on development set:\n"
print clf.best_params_

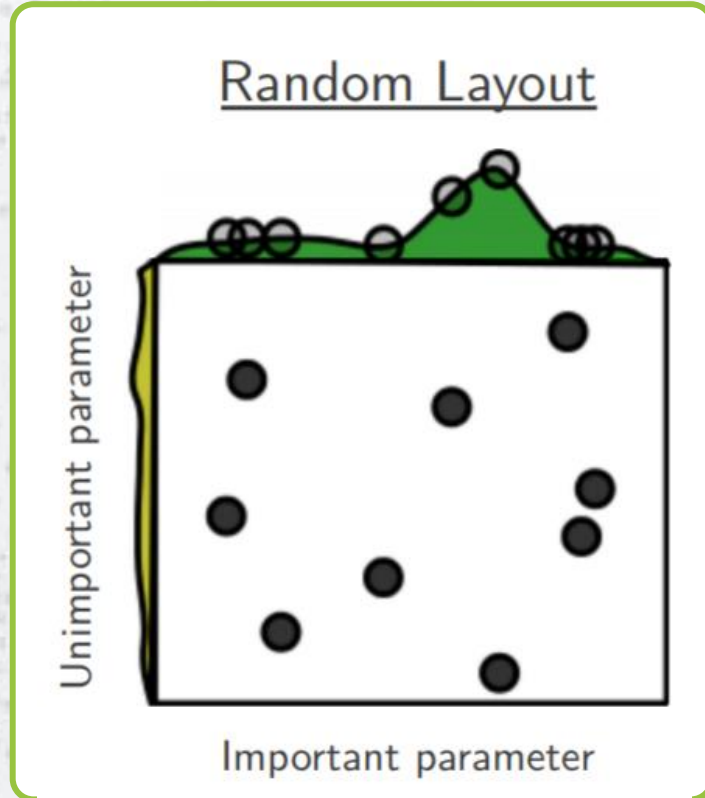
print "Grid scores on development set:\n"
for params, mean_score, scores in clf.grid_scores_:
    print "%.3f (+/-%.03f) for %r\n" % (mean_score, scores.std() * 2, params)

print "MSE for test data set:\n"
y_true, y_pred = self.y_test, clf.predict(self.X_test)
print mean_squared_error(y_true, y_pred)
```



# Métodos de Optimización

**Random Search:** Búsqueda Aleatoria. En este método, se define un espacio de búsqueda para cada hiperparámetro y se seleccionan valores aleatorios dentro de ese espacio para entrenar y evaluar el modelo. Es menos costoso computacionalmente que la búsqueda en cuadrícula, pero puede requerir más iteraciones para encontrar los valores óptimos.



# Métodos de Optimización

- **Bayesian Optimization:** Este método utiliza un modelo de probabilidad para modelar la relación entre los hiperparámetros y la métrica de rendimiento del modelo. A partir de esta relación, se realiza una búsqueda en el espacio de hiperparámetros para encontrar los valores que maximizan la métrica de rendimiento.
- **Particle Swarm Optimization:** Optimización por enjambre de partículas. Este método utiliza un conjunto de partículas que se mueven en el espacio de hiperparámetros para encontrar los valores óptimos. Las partículas se mueven hacia las regiones del espacio de hiperparámetros que han producido mejores resultados de rendimiento.
- **Differential Evolution:** Optimización por evolución diferencial. Este método utiliza una población de soluciones candidatas y las combina mediante operadores genéticos, como la mutación y la recombinación, para producir nuevas soluciones candidatas. Las soluciones candidatas que producen mejores resultados de rendimiento se mantienen en la población y el proceso se repite hasta que se alcanza un criterio de parada.

Cada uno de estos métodos tiene sus propias ventajas y desventajas, y la elección del método de optimización de hiperparámetros dependerá del problema específico que se esté abordando y de las limitaciones computacionales disponibles.



Regularización

# Regularización

La Regularización **es** una técnica utilizada para disminuir el nivel de sobreajuste de un modelo, atenuando la magnitud de los parámetros de forma que el modelo sea menos “sensible” a la data de entrenamiento y así tener mejor capacidad de generalización.

La regularización se realiza agregando un término adicional a la función de costo. Por ejemplo, en una regresión lineal:

- **Regularización L2**

Penaliza la función de costo con un término adicional equivalente a la suma de los cuadrados de los coeficientes.

Función de Costo =  $RSS + \alpha * (\text{suma del cuadrado de los coeficientes})$

- **Regularización L1**

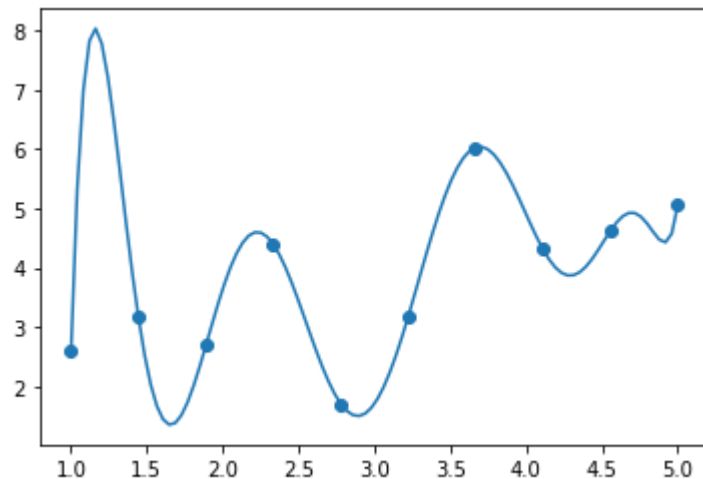
Penaliza la función de costo con un término adicional equivalente a la suma de los valores absolutos de los coeficientes.

Función de Costo =  $RSS + \alpha * (\text{suma de los valores absolutos de los coeficientes})$

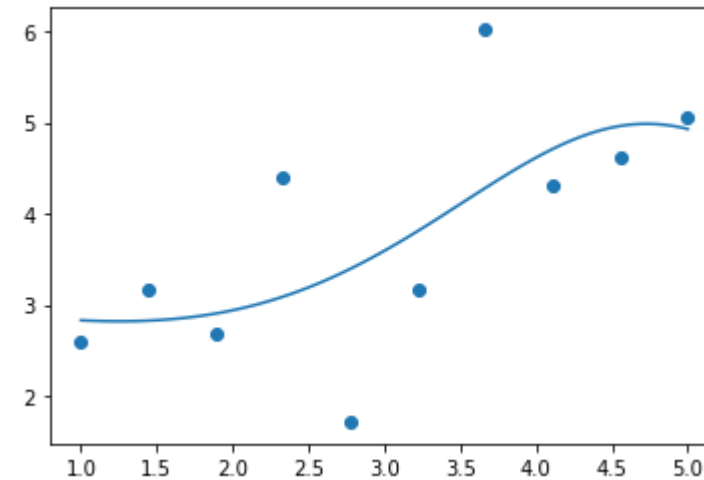


# Regularización

En los siguientes diagramas se aprecia un modelo sobreajustado (figura izquierda) y un modelo regularizado (figura derecha) el cual tiene un menor error de varianza y por ende una mejor generalización.



Ajuste con regresión polinomial y OLS  
(sin regularización)



Ajuste con regresión Ridge  
(regularización L2) y  $\alpha=0.001$

The background of the slide is a grayscale image of a book cover. The cover features a repeating pattern of stylized, overlapping leaf or feather shapes. A solid green rectangular banner is positioned horizontally across the middle of the image, partially obscuring the book cover pattern.

Dudas y consultas



Gracias