UNIVERSIDAD SANTO TOMÁS
PRIMER CLAUSTRO UNIVERSITARIO DE COLOMBIA
SECCIONAL TUNJA
VIGILADA MINEDUCACIÓN - SNIES 1732

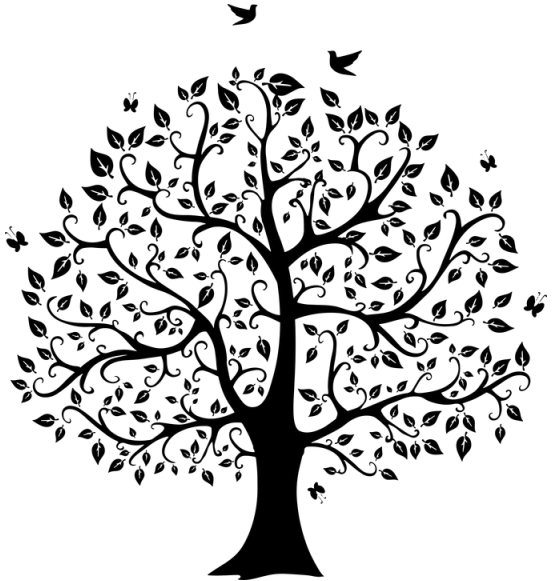
# APRENDIZAJE SUPERVISADO

# ÁRBOLES DE DECISIÓN

# ÁRBOLES DE DECISIÓN

Se usa esta analogía para describir una técnica de aprendizaje supervisado que permite predecir valores de respuestas utilizando reglas de decisión.

Esta técnica se puede usar tanto en un problema de Clasificación como en un problema de Regresión.



# ÁRBOLES DE DECISIÓN

| Árboles de decisión - Regresión | Árboles de decisión - Clasificación |
|---|---|
| Variable dependiente es continua | Variable dependiente es categórica |
| Valores de los nodos terminales u hojas corresponden a la media de las observaciones en esa región. | El valor en el nodo terminal u hoja corresponde a la moda de las observaciones de esa región. |

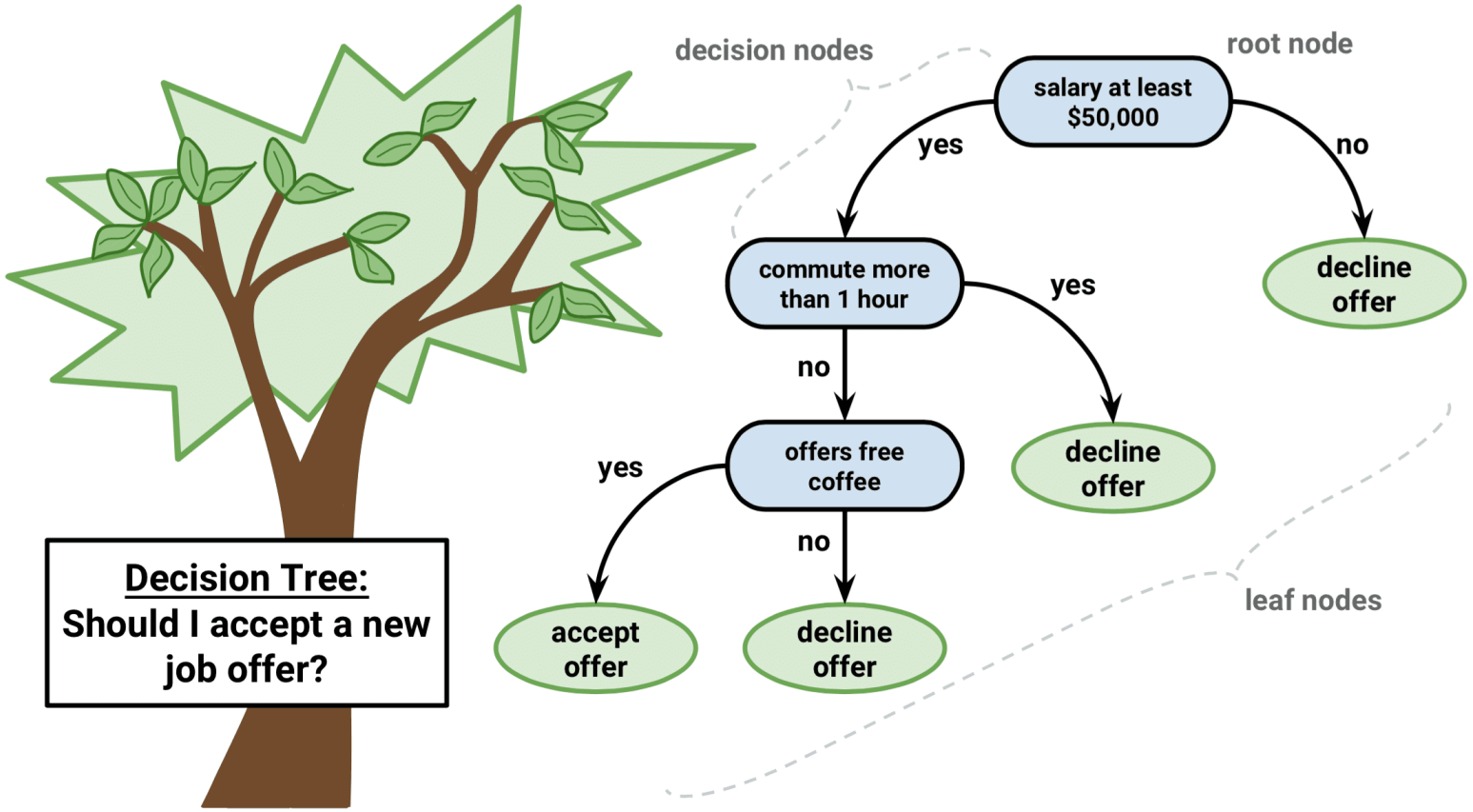
# ÁRBOLES DE DECISIÓN

Al aplicar esta técnica, se divide el espacio de las características en regiones rectangulares.

Los árboles de decisión son modelos predictivos que aplican reglas binarias (si/no) para repartir las observaciones en función de sus atributos y predecir así el valor de la variable respuesta.

Para ello, es como si el algoritmo generara una serie de preguntas para clasificar los datos.

# ÁRBOLES DE DECISIÓN

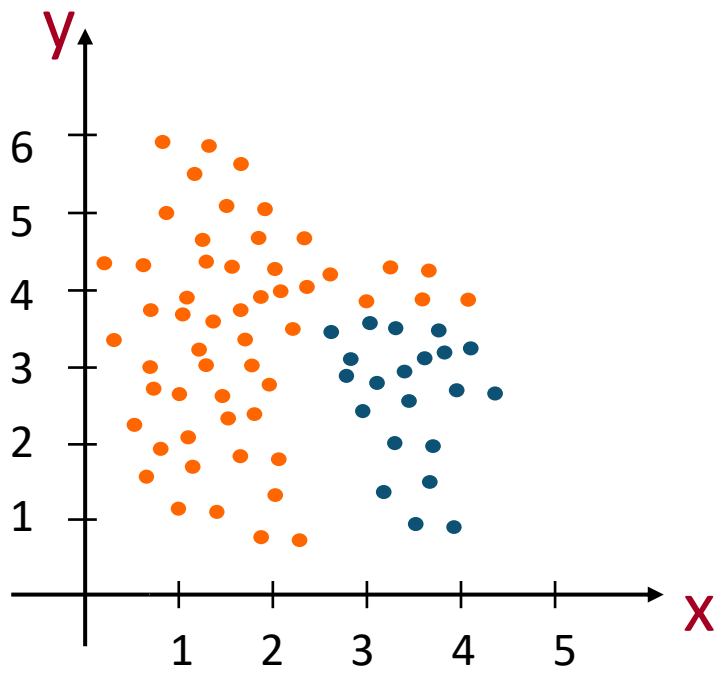



- Nodo raíz
- Ramificación
- Nodo de decisión
- Nodo terminal y hoja

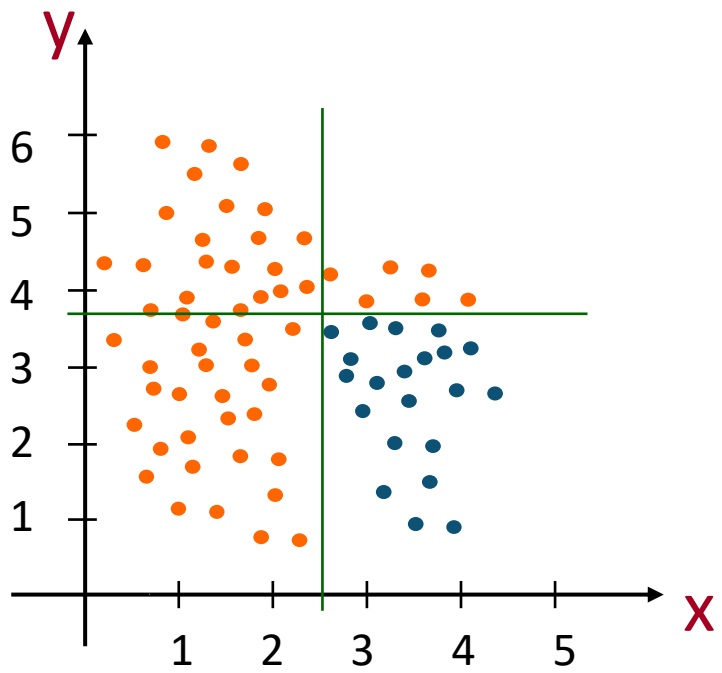
Imagen tomada de: https://bookdown.org/content/2031/images/terminology.png

¡Siempre hacia lo alto!

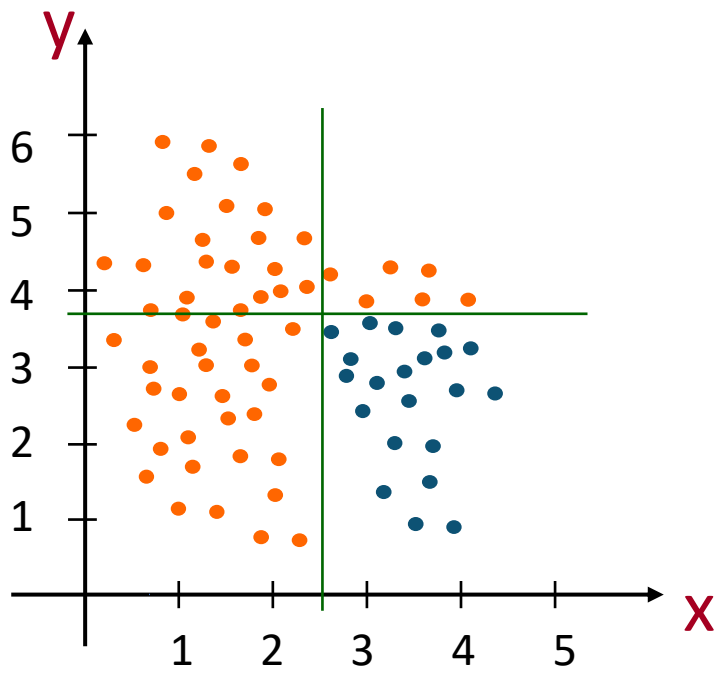
# ÁRBOLES DE DECISIÓN

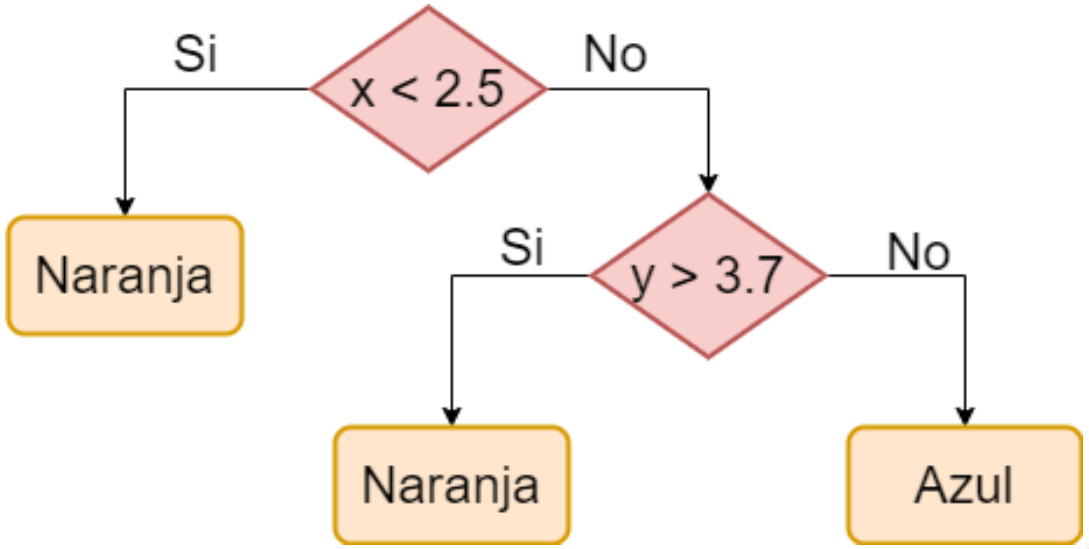


# ÁRBOLES DE DECISIÓN



# ÁRBOLES DE DECISIÓN




¡Siempre hacia lo alto!

# ÁRBOLES DE DECISIÓN

```
#POR ÁRBOLES DE DECISIÓN
from sklearn.tree import DecisionTreeClassifier
modelo2 = DecisionTreeClassifier()
modelo2.fit(x_train,y_train)
y_pred=modelo2.predict(x_test)
print("Precisión por Árboles de decisión: ",modelo2.score(x_train,y_train))
```

Es propenso a sobreajuste.

Si la precisión da como resultado 1,0 (significa un 100% de ajuste) se llama sobreajuste

# ÁRBOLES DE DECISIÓN

Para solucionar el sobreajuste:

Se puede solucionar deteniendo el crecimiento del árbol.

Parámetros:

- Criterion: default = “**mse**” (Media del error cuadrado). Para versiones nuevas está MAE (Criterio del error absoluto promedio).
- Splitter: división de cada nodo (**best** o random).
- **Max_depth**: si no se tiene un valor se deja crecer sin límite. Se puede establecer un límite para evitar el sobreajuste.

# ÁRBOLES DE DECISIÓN

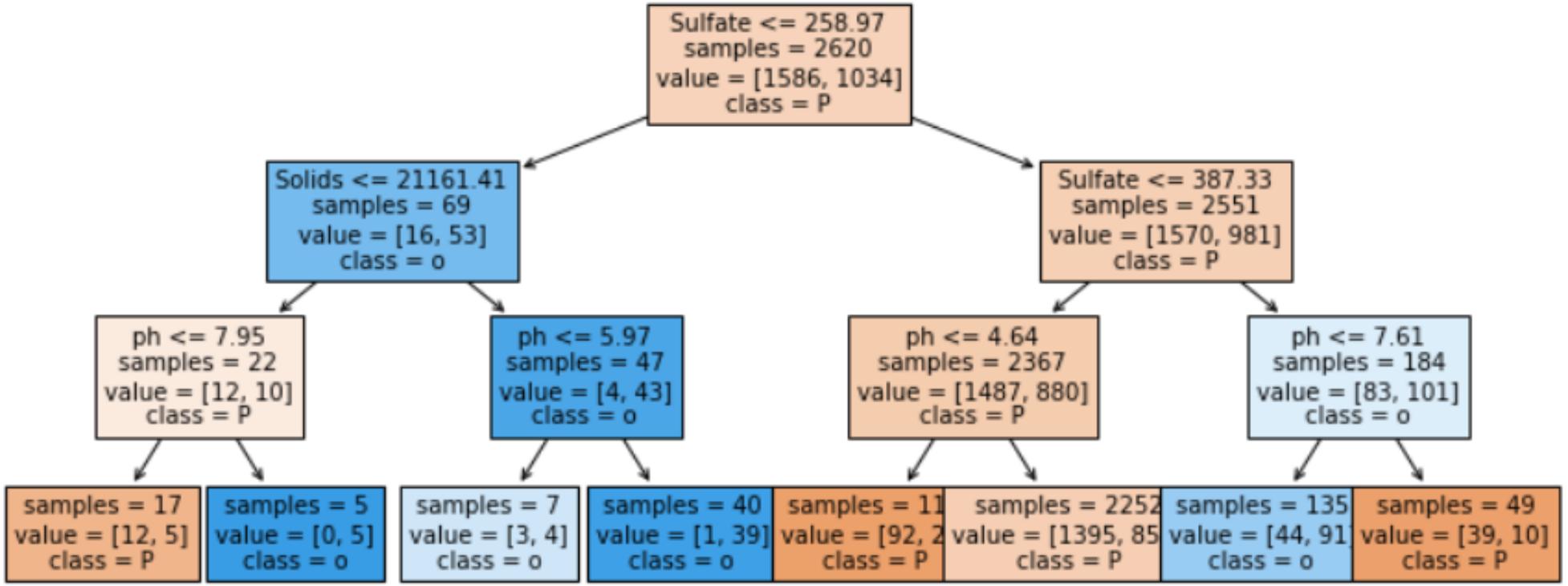
```
from sklearn import tree
fig, ax = plt.subplots(figsize=(12, 5))

print(f"Profundidad del árbol: {modelo2.get_depth()}")
print(f"Número de nodos terminales: {modelo2.get_n_leaves()}")

tree=tree.plot_tree(
            decision_tree = modelo2,
            feature_names = TrainData.drop(columns = "Potability").columns,
            class_names   = 'Potability',
            filled        = True,
            impurity      = False,
            fontsize      = 10,
            precision     = 2,
            ax            = ax
       )
```

# ÁRBOLES DE DECISIÓN

Profundidad del árbol: 3
Número de nodos terminales: 8



# Sobreajuste y subajuste (overfitting and underfitting)

# Sobrejuste y subajuste

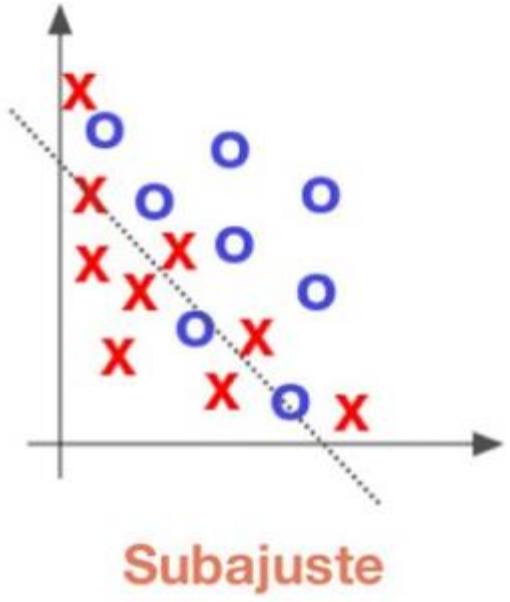
Cuando se hace entrenamiento de datos, se busca encontrar la precisión con la cual se ejecutan estas tareas.
Cuando no se obtiene una buena precisión (mayor a 0,8) se suele ajustar el número de las características que se han tenido en cuenta o ajustar a otro modelo.

# Sobrejuste y subajuste

Cuando se hace entrenamiento de datos, se busca encontrar la precisión con la cual se ejecutan estas tareas.
Cuando no se obtiene una buena precisión (mayor a 0,8) se suele ajustar el número de las características que se han tenido en cuenta o ajustar a otro modelo.

No obstante, algunas ocasiones se puede concluir que el modelo es muy simple para describir el objetivo o que es muy complejo para expresar el objetivo.

# Sobrejuste y subajuste




La línea de las predicciones no cubre todos los puntos.

Imagen tomada de: https://aprendeia.com/sobreajuste-y-subajuste-en-machine-learning/

# Sobrejuste y subajuste

La línea de las predicciones si cubre todos los puntos del gráfico, incluso los valores atípicos. Sin embargo, también cubre el ruido, por lo cual no es tan bueno.

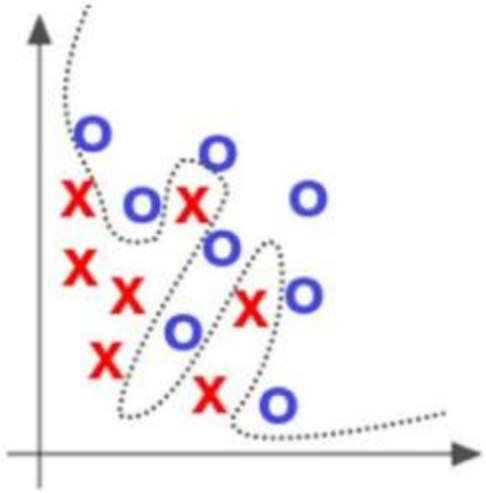



Imagen tomada de: https://aprendeia.com/sobreajuste-y-subajuste-en-machine-learning/

# Sobrejuste y subajuste

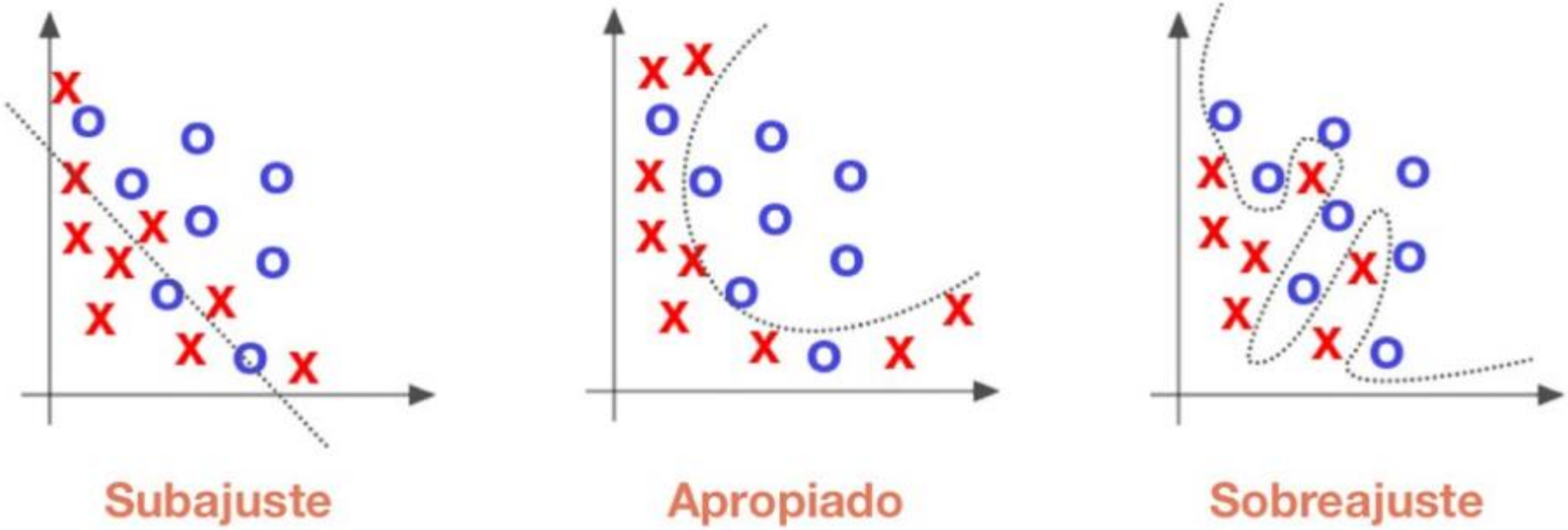



Imagen tomada de: https://aprendeia.com/sobreajuste-y-subajuste-en-machine-learning/

# Sobrejuste y subajuste

El sobreajuste en el aprendizaje automático se produce cuando un modelo encaja **demasiado bien** con los datos de entrenamiento y, como resultado, no puede realizar predicciones con precisión sobre datos de prueba desconocidos. En otras palabras, el modelo simplemente ha memorizado patrones y ruido específicos de los datos de entrenamiento, pero no es lo suficientemente flexible como para realizar predicciones sobre datos reales.

Tomado de: https://docs.microsoft.com/es-es/azure/machine-learning/concept-manage-ml-pitfalls

# Sobrejuste y subajuste

| Modelo | Precisión del entrenamiento | Precisión de la prueba | |
|---|---|---|---|
| A | 99,9 % | 95 % | |
| B | 87 % | 87 % | |
| C | 99,9 % | 45 % | Sobrejuste |

La precisión de la prueba siempre debe ser menor que el de entrenamiento. Sin embargo, si la diferencia entre estos valores es muy grande, se habla de sobreajuste.

Tomado de: https://docs.microsoft.com/es-es/azure/machine-learning/concept-manage-ml-pitfalls

# K-Nearest Neighbor(KNN)

# K-Nearest Neighbor(KNN)

El algoritmo K-NN almacena todos los datos disponibles y clasifica un nuevo punto de datos en función de la similitud.

El algoritmo K-NN se puede utilizar tanto para la regresión como para la clasificación, pero sobre todo se utiliza para los problemas de clasificación.

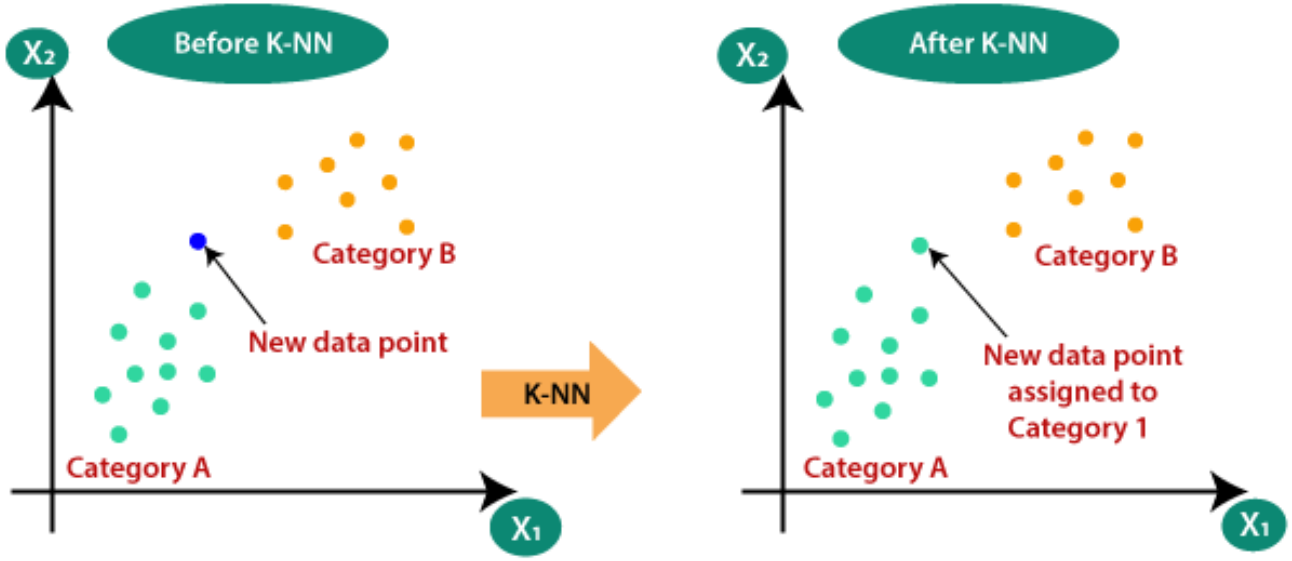



Imagen tomada de: https://static.javatpoint.com/tutorial/machine-learning/images/k-nearest-neighbor-algorithm-for-machine-learning2.png

# K-Nearest Neighbor(KNN)

El algoritmo busca el vecino más cercano, calculando la distancia con cada etiqueta de las clasificaciones existentes.

- Se debe definir el valor de K (número de vecinos).
- Se recomienda un número impar si el número de clases es par.

```
#vecinos más cercanos KNN
from sklearn.neighbors import KNeighborsClassifier
modelo4 = KNeighborsClassifier(n_neighbors = 3)
modelo4.fit(x_train, y_train)
y_pred =modelo4.predict(x_test)
print("Precisión por Vecino más cercano: ",modelo4.score(x_train,y_train))
```

```
Precisión por Vecino más cercano:  0.7645038167938931
```

# K-Nearest Neighbor(KNN)

**Parámetros**:

- N_neighbors (K): int (default = 5).
- Para la distancia euclidiana: p=2 y metric = “minkowski”

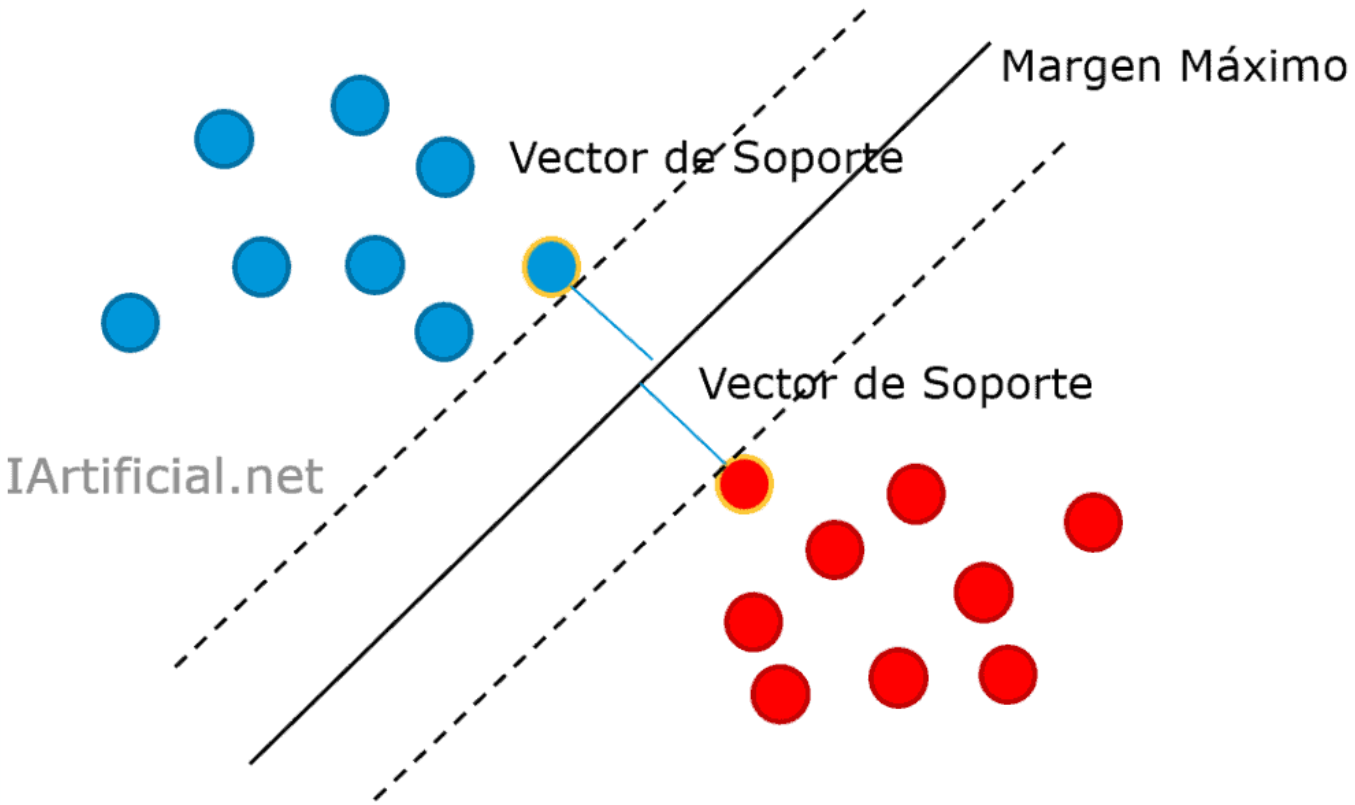
```
#vecinos más cercanos KNN
from sklearn.neighbors import KNeighborsClassifier
modelo4 = KNeighborsClassifier(n_neighbors = 3 , metric = 'minkowski' , p = 2)
modelo4.fit(x_train, y_train)
y_pred =modelo4.predict(x_test)
print("Precisión por Vecino más cercano: ",modelo4.score(x_train,y_train))
```

```
Precisión por Vecino más cercano:  0.7645038167938931
```

# Máquinas de Vectores de soporte

# Máquinas de vectores de soporte

**SVM** (por sus siglas en inglés) permite lograr un muy buen ajuste




- Vectores de soporte: puntos de datos más cercanos al hiperplano.
- Hiperplano: plano de decisión que separa los datos.
- Margen: espacio entre las dos líneas en los puntos más cercanos de la clase.

Imagen tomada de: https://www.iartificial.net/wp-content/uploads/2019/04/Clasificacion-SVM-768x436.webp

# Máquinas de vectores de soporte

```
# Support Vector Machines
from sklearn.svm import SVC
modelo3 = SVC()
modelo3.fit(x_train, y_train)
y_pred =modelo3.predict(x_test)
print("Precisión por Support Vector Machines: ",modelo3.score(x_train,y_train))
```

```
Precisión por Support Vector Machines:  0.6053435114503817
```

**Parámetros**:

- Kernel: string (default=“rbf”). Rbf hace referencia a radial.
- C (default=1): es un parámetro de regularización y representa un término de error (máximo permitido).

# REFERENCIAS


- https://www.cienciadedatos.net/documentos/py07_arboles_decision_python.html
- https://bookdown.org/content/2031/arboles-de-decision-parte-i.html
- https://docs.microsoft.com/es-es/azure/machine-learning/concept-manage-ml-pitfalls