Clase 1 - Introducción a Machine Learning

Facundo González

18 de abril de 2024

Outline

Cómo resolvemos un problema tradicional

Problema: decidir si un número es par.
Por ejemplo: `Entrada: 2, Salida: True; Entrada: 3, Salida: False`
¿Cómo resolvemos este problema?

Cómo resolvemos un problema tradicional

Problema: decidir si un número es par.

`Entrada: 2, Salida: True; Entrada: 3, Salida: False`

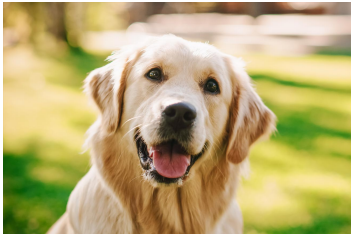
¿Cómo resolvemos este problema?

```
1: function esPar(n)
2:     if n es par then
3:         return verdadero
4:     else
5:         return falso
6:     end if
7: end function
```

Cómo resolvemos un problema no tradicional

Problema: decidir si una una imagen es de un gato.

`Entrada:`  `, Salida: True`

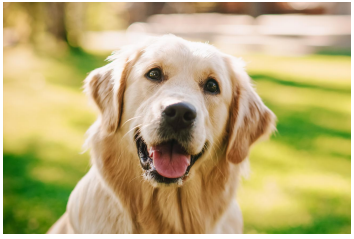
`Entrada:`  `, Salida: False`

¿Cómo resolvemos este problema?

Cómo resolvemos un problema no tradicional

Problema: decidir si una una imagen es de un gato.

`Entrada:`  `, Salida: True`

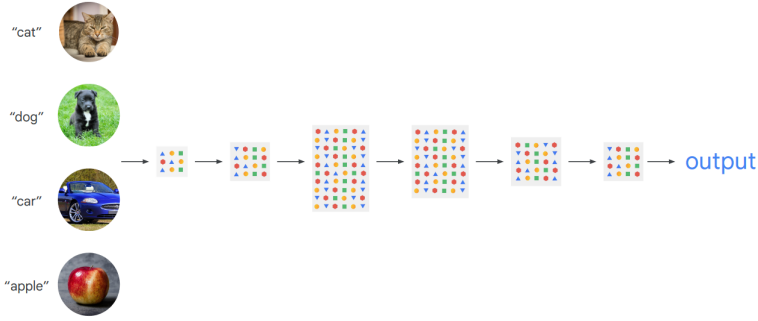
`Entrada:`  `, Salida: False`

¿Cómo resolvemos este problema?

¿Podemos establecer una serie de pasos para decidir si la imagen es de un gato o no?

Cómo resolvemos un problema no tradicional

Enfoque de Machine Learning: tomamos muchos datos **etiquetados**, y entrenamos un modelo para que pueda distinguir si en una foto hay gato o no.



En este caso los datos van a ser imagenes, y la etiqueta va a decir si hay un gato en la imagen o no.

Introducción a Machine Learning

Vamos a querer modelos que **generalicen** el comportamiento de ciertos datos para luego poder hacer predicciones o inferencias.

- **Inteligencia Artificial:** creación de máquinas que imiten la inteligencia humana para realizar tareas, y que pueden mejorar conforme recopilen información

Introducción a Machine Learning

Vamos a querer modelos que **generalicen** el comportamiento de ciertos datos para luego poder hacer predicciones o inferencias.

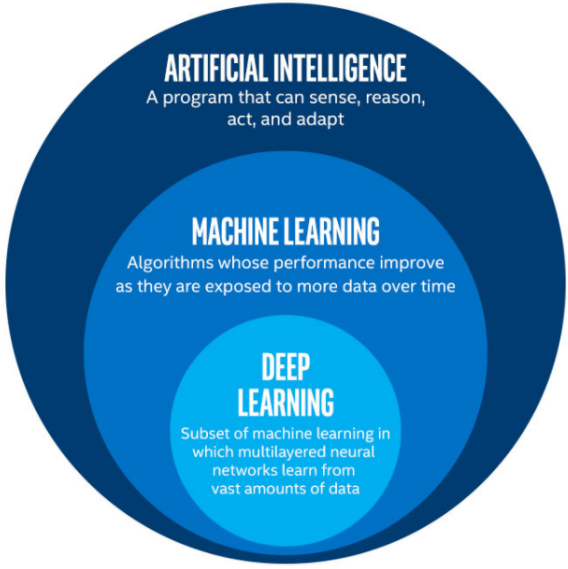
- **Inteligencia Artificial:** creación de máquinas que imiten la inteligencia humana para realizar tareas, y que pueden mejorar conforme recopilen información
- **Machine Learning:** subconjunto de Inteligencia Aritificial, que se ocupa de desarrollar algoritmos que puedan generalizar y “aprender”de un dataset.

Introducción a Machine Learning

Vamos a querer modelos que **generalicen** el comportamiento de ciertos datos para luego poder hacer predicciones o inferencias.

- **Inteligencia Artificial:** creación de máquinas que imiten la inteligencia humana para realizar tareas, y que pueden mejorar conforme recopilen información
- **Machine Learning:** subconjunto de Inteligencia Aritificial, que se ocupa de desarrollar algoritmos que puedan generalizar y "aprender"de un dataset.
- **Deep Learning:** o Aprendizaje Profundo, es un subconjunto de Machine Learning que utiliza abstracciones más complejas para desarrollar modelos, por ejemplo Redes Neuronales.

Introducción a Machine Learning



Más sobre Machine Learning

Vamos a distinguir distintipo tipos de modelos de Machine Learning:

- **Aprendizaje Supervisado:** tomamos datos etiquetados para entrenar el modelo.

Más sobre Machine Learning

Vamos a distinguir distintipo tipos de modelos de Machine Learning:

- **Aprendizaje Supervisado:** tomamos datos etiquetados para entrenar el modelo.
  - Clasificación: clasificar entre N categorías.

Más sobre Machine Learning

Vamos a distinguir distintipo tipos de modelos de Machine Learning:

- **Aprendizaje Supervisado:** tomamos datos etiquetados para entrenar el modelo.
  - Clasificación: clasificar entre N categorías.
  - Regresión: obtener un valor continuo (número real).

Más sobre Machine Learning

Vamos a distinguir distintipo tipos de modelos de Machine Learning:

- **Aprendizaje Supervisado:** tomamos datos etiquetados para entrenar el modelo.
  - Clasificación: clasificar entre N categorías.
  - Regresión: obtener un valor continuo (número real).
- **Aprendizaje No-Supervisado:** no hay una etiqueta o variable objetivo.

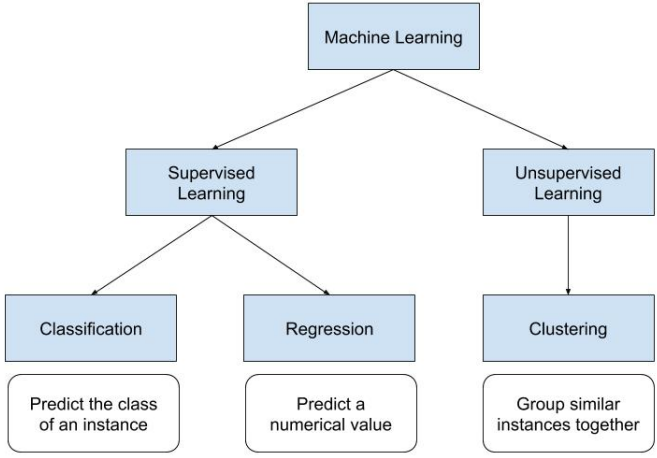
Más sobre Machine Learning

Vamos a distinguir distintipo tipos de modelos de Machine Learning:

- **Aprendizaje Supervisado:** tomamos datos etiquetados para entrenar el modelo.
  - Clasificación: clasificar entre N categorías.
  - Regresión: obtener un valor continuo (número real).
- **Aprendizaje No-Supervisado:** no hay una etiqueta o variable objetivo.
  - Clustering: agrupar los datos encontrar similaridades.

Tipos de Machine Learning

Figura: Tipos de Machine Learning.



ML vs Estadística

Los modelos de Machine Learning tienen una base fuerte de estadística, pero son dos áreas con enfoques distintos:

- Estadística: se utilizan los datos que se tienen, descartando *outliers* para analizar el comportamiento de los mismos. Se demustra que alguna hipótesis sobre los datos está suficientemente respaldada.

ML vs Estadística

Los modelos de Machine Learning tienen una base fuerte de estadística, pero son dos áreas con enfoques distintos:

- Estadística: se utilizan los datos que se tienen, descartando *outliers* para analizar el comportamiento de los mismos. Se demustra que alguna hipótesis sobre los datos está suficientemente respaldada.
- Machine Learning: se utilizan **todos** los datos para poder generalizar el comporamiento (mediante un modelo) y realizar predicciones iterativamente. Modelos complicados para poder predecir el comportamiento de un dato nuevo.

Modelos de Clasifiación

Vamos a usar **Modelos de Clasificación** cuando busquemos predecir en qué categoria se va a encontrar un dato nuevo. Es decir, buscamos clasificar un dato.
Ejemplo: modelo para detectar si un mail es spam o no. En este caso la clasificación es binaria (entre dos opciones). Modelos de Clasificación:

- Logistic Regression: para clasificación binaria, es un modelo más simple que los demas.

Modelos de Clasifiación

Vamos a usar **Modelos de Clasificación** cuando busquemos predecir en qué categoria se va a encontrar un dato nuevo. Es decir, buscamos clasificar un dato.
Ejemplo: modelo para detectar si un mail es spam o no. En este caso la clasificación es binaria (entre dos opciones). Modelos de Clasificación:

- Logistic Regression: para clasificación binaria, es un modelo más simple que los demas.
- XGBoost: algoritmo que ensambla otros modelos para mejorar performance.

Modelos de Clasifiación

Vamos a usar **Modelos de Clasificación** cuando busquemos predecir en qué categoria se va a encontrar un dato nuevo. Es decir, buscamos clasificar un dato.
Ejemplo: modelo para detectar si un mail es spam o no. En este caso la clasificación es binaria (entre dos opciones). Modelos de Clasificación:

- Logistic Regression: para clasificación binaria, es un modelo más simple que los demas.
- XGBoost: algoritmo que ensambla otros modelos para mejorar performance.
- Decision Trees: árbol con decisiones para clasificar los datos. Pueden generar overfitting y no generalizar bien.

Modelos de Clasifiación

Vamos a usar **Modelos de Clasificación** cuando busquemos predecir en qué categoria se va a encontrar un dato nuevo. Es decir, buscamos clasificar un dato.
Ejemplo: modelo para detectar si un mail es spam o no. En este caso la clasificación es binaria (entre dos opciones). Modelos de Clasificación:

- Logistic Regression: para clasificación binaria, es un modelo más simple que los demas.
- XGBoost: algoritmo que ensambla otros modelos para mejorar performance.
- Decision Trees: árbol con decisiones para clasificar los datos. Pueden generar overfitting y no generalizar bien.
- Random forest: utiliza muchos árboles de decisión para evitar overfitting y generalizar mejor.

Modelos de Clasifiación

Vamos a usar **Modelos de Clasificación** cuando busquemos predecir en qué categoria se va a encontrar un dato nuevo. Es decir, buscamos clasificar un dato.
Ejemplo: modelo para detectar si un mail es spam o no. En este caso la clasificación es binaria (entre dos opciones). Modelos de Clasificación:

- Logistic Regression: para clasificación binaria, es un modelo más simple que los demas.
- XGBoost: algoritmo que ensambla otros modelos para mejorar performance.
- Decision Trees: árbol con decisiones para clasificar los datos. Pueden generar overfitting y no generalizar bien.
- Random forest: utiliza muchos árboles de decisión para evitar overfitting y generalizar mejor.
- DNN clasifiers: Red neuronal profunda para clasificación.

Modelos de Clasifiación

Vamos a usar **Modelos de Clasificación** cuando busquemos predecir en qué categoria se va a encontrar un dato nuevo. Es decir, buscamos clasificar un dato.
Ejemplo: modelo para detectar si un mail es spam o no. En este caso la clasificación es binaria (entre dos opciones). Modelos de Clasificación:

- Logistic Regression: para clasificación binaria, es un modelo más simple que los demas.
- XGBoost: algoritmo que ensambla otros modelos para mejorar performance.
- Decision Trees: árbol con decisiones para clasificar los datos. Pueden generar overfitting y no generalizar bien.
- Random forest: utiliza muchos árboles de decisión para evitar overfitting y generalizar mejor.
- DNN clasifiers: Red neuronal profunda para clasificación.

  Algunas implementaciones: DecisionTree, RandomForest

Modelos de Regresión

Los modelos de **regresión** nos van a servir cuando necesitemos predecir o inferir un número continuo. En este caso, la variable objetivo (o *label*) va a ser continua.
Modelos de Regresión:

- Linear Regression (y Polynomial Regression): busca ajustar los datos con la función lineal (o polinomio) que minimice error.

Modelos de Regresión

Los modelos de **regresión** nos van a servir cuando necesitemos predecir o inferir un número continuo. En este caso, la variable objetivo (o *label*) va a ser continua.
Modelos de Regresión:

- Linear Regression (y Polynomial Regression): busca ajustar los datos con la función lineal (o polinomio) que minimice error.
- XGBoost: se puede usar también para regresión.

Modelos de Regresión

Los modelos de **regresión** nos van a servir cuando necesitemos predecir o inferir un número continuo. En este caso, la variable objetivo (o *label*) va a ser continua.
Modelos de Regresión:

- Linear Regression (y Polynomial Regression): busca ajustar los datos con la función lineal (o polinomio) que minimice error.
- XGBoost: se puede usar también para regresión.
- DNN regressos: red neuronal profunda para regresión.

Modelos de Regresión

Los modelos de **regresión** nos van a servir cuando necesitemos predecir o inferir un número continuo. En este caso, la variable objetivo (o *label*) va a ser continua.
Modelos de Regresión:

- Linear Regression (y Polynomial Regression): busca ajustar los datos con la función lineal (o polinomio) que minimice error.
- XGBoost: se puede usar también para regresión.
- DNN regressos: red neuronal profunda para regresión.
- Decision trees (regression): árboles de decisión que tienen como output una variable continua.

Modelos de Regresión

Los modelos de **regresión** nos van a servir cuando necesitemos predecir o inferir un número continuo. En este caso, la variable objetivo (o *label*) va a ser continua.
Modelos de Regresión:

- Linear Regression (y Polynomial Regression): busca ajustar los datos con la función lineal (o polinomio) que minimice error.
- XGBoost: se puede usar también para regresión.
- DNN regressos: red neuronal profunda para regresión.
- Decision trees (regression): árboles de decisión que tienen como output una variable continua.
- Random forest: funcionan como los bosques para clasificación, utilizando la media de lo que devuelve cada árbol.

Modelos de Regresión

Los modelos de **regresión** nos van a servir cuando necesitemos predecir o inferir un número continuo. En este caso, la variable objetivo (o *label*) va a ser continua.
Modelos de Regresión:

- Linear Regression (y Polynomial Regression): busca ajustar los datos con la función lineal (o polinomio) que minimice error.
- XGBoost: se puede usar también para regresión.
- DNN regressos: red neuronal profunda para regresión.
- Decision trees (regression): árboles de decisión que tienen como output una variable continua.
- Random forest: funcionan como los bosques para clasificación, utilizando la media de lo que devuelve cada árbol.

  Algunas implementaciones: DecisionTreeRegressor, RandomForestRegressor

Mini Ejercicio

Decidir si los siguientes problemas se resuelven con problemas de regresión o clasificación:

1. Predecir el precio de una casa en dólares.

Mini Ejercicio

Decidir si los siguientes problemas se resuelven con problemas de regresión o clasificación:

1. Predecir el precio de una casa en dólares.
   → **Regresión**

Mini Ejercicio

Decidir si los siguientes problemas se resuelven con problemas de regresión o clasificación:

1. Predecir el precio de una casa en dólares.
   → **Regresión**
2. Predecir si un nuevo mail es spam.

Mini Ejercicio

Decidir si los siguientes problemas se resuelven con problemas de regresión o clasificación:

1. Predecir el precio de una casa en dólares.
   → **Regresión**
2. Predecir si un nuevo mail es spam.
   → **Clasificación**

Mini Ejercicio

Decidir si los siguientes problemas se resuelven con problemas de regresión o clasificación:

1. Predecir el precio de una casa en dólares.
   → **Regresión**
2. Predecir si un nuevo mail es spam.
   → **Clasificación**
3. Predecir cuántos mm de lluvia va a haber en un mes.

Mini Ejercicio

Decidir si los siguientes problemas se resuelven con problemas de regresión o clasificación:

1. Predecir el precio de una casa en dólares.
   → **Regresión**
2. Predecir si un nuevo mail es spam.
   → **Clasificación**
3. Predecir cuántos mm de lluvia va a haber en un mes.
   → **Regresión**

Mini Ejercicio

Decidir si los siguientes problemas se resuelven con problemas de regresión o clasificación:

1. Predecir el precio de una casa en dólares.
   → **Regresión**
2. Predecir si un nuevo mail es spam.
   → **Clasificación**
3. Predecir cuántos mm de lluvia va a haber en un mes.
   → **Regresión**
4. Predecir cuántos minutos va a tardar una persona en auto en hacer un recorrido.

Mini Ejercicio

Decidir si los siguientes problemas se resuelven con problemas de regresión o clasificación:

1. Predecir el precio de una casa en dólares.
   → **Regresión**
2. Predecir si un nuevo mail es spam.
   → **Clasificación**
3. Predecir cuántos mm de lluvia va a haber en un mes.
   → **Regresión**
4. Predecir cuántos minutos va a tardar una persona en auto en hacer un recorrido.
   → **Regresión**

Mini Ejercicio

Decidir si los siguientes problemas se resuelven con problemas de regresión o clasificación:

1. Predecir el precio de una casa en dólares.
   → **Regresión**
2. Predecir si un nuevo mail es spam.
   → **Clasificación**
3. Predecir cuántos mm de lluvia va a haber en un mes.
   → **Regresión**
4. Predecir cuántos minutos va a tardar una persona en auto en hacer un recorrido.
   → **Regresión**
5. Predecir si un comentario en una red social tiene una connotación negativa, positiva o neutra.

Mini Ejercicio

Decidir si los siguientes problemas se resuelven con problemas de regresión o clasificación:

1. Predecir el precio de una casa en dólares.
   → **Regresión**
2. Predecir si un nuevo mail es spam.
   → **Clasificación**
3. Predecir cuántos mm de lluvia va a haber en un mes.
   → **Regresión**
4. Predecir cuántos minutos va a tardar una persona en auto en hacer un recorrido.
   → **Regresión**
5. Predecir si un comentario en una red social tiene una connotación negativa, positiva o neutra.
   → **Clasificación**

Dataset

El dataset es el conjunto de todos los datos que poseemos para crear el modelo. En caso de aprendizaje supervisado, estos datos deben estar etiquetados y debemos saber cual es la etiqueta o *label*.

Dataset

El dataset es el conjunto de todos los datos que poseemos para crear el modelo. En caso de aprendizaje supervisado, estos datos deben estar etiquetados y debemos saber cual es la etiqueta o *label*.

Para poder trabajar en un modelo de ML vamos a necesitar separar el dataset en (al menos) dos grupos:

Dataset

El dataset es el conjunto de todos los datos que poseemos para crear el modelo. En caso de aprendizaje supervisado, estos datos deben estar etiquetados y debemos saber cual es la etiqueta o *label*.

Para poder trabajar en un modelo de ML vamos a necesitar separar el dataset en (al menos) dos grupos:

- Dataset de entrenamiento: los datos que va a recibir el modelo para ser entrenado.

Dataset

El dataset es el conjunto de todos los datos que poseemos para crear el modelo. En caso de aprendizaje supervisado, estos datos deben estar etiquetados y debemos saber cual es la etiqueta o *label*.

Para poder trabajar en un modelo de ML vamos a necesitar separar el dataset en (al menos) dos grupos:

- Dataset de entrenamiento: los datos que va a recibir el modelo para ser entrenado.
- Dataset de testeo: datos que vamos a utilizar como muestra real para testear el modelo.

Dataset

El dataset es el conjunto de todos los datos que poseemos para crear el modelo. En caso de aprendizaje supervisado, estos datos deben estar etiquetados y debemos saber cual es la etiqueta o *label*.

Para poder trabajar en un modelo de ML vamos a necesitar separar el dataset en (al menos) dos grupos:

- Dataset de entrenamiento: los datos que va a recibir el modelo para ser entrenado.
- Dataset de testeo: datos que vamos a utilizar como muestra real para testear el modelo.

Esta separación se debe hacer correctamente para que ambos datasets tengan el mismo "comportamiento" estadísticamente. Para eso podemos utilizar librerías de Python que hacen el trabajo: PyTorch, TensorFlow, SKlearn.

Testear el modelo

Ahora tenemos una idea de cómo testear el modelo, utilizando el dataset de testeo. ¿Cómo podemos hacer esto? Hay varias métricas que podemos tener en cuenta:

- Regresión

Testear el modelo

Ahora tenemos una idea de cómo testear el modelo, utilizando el dataset de testeo. ¿Cómo podemos hacer esto? Hay varias métricas que podemos tener en cuenta:

- Regresión
  - **Root Mean Squared Error:** $RMSE = \sqrt{\frac{1}{n}\sum(t_i - y_i)^2}$.

Testear el modelo

Ahora tenemos una idea de cómo testear el modelo, utilizando el dataset de testeo. ¿Cómo podemos hacer esto? Hay varias métricas que podemos tener en cuenta:

- Regresión
  - **Root Mean Squared Error:** $RMSE = \sqrt{\frac{1}{n}\sum(t_i - y_i)^2}$.
  - **Pearson:** R^2, va entre 0 y 1 donde más cercano a 1 es mejor.

Testear el modelo

Ahora tenemos una idea de cómo testear el modelo, utilizando el dataset de testeo. ¿Cómo podemos hacer esto? Hay varias métricas que podemos tener en cuenta:

- Regresión
  - **Root Mean Squared Error:** $RMSE = \sqrt{\frac{1}{n}\sum(t_i - y_i)^2}$.
  - **Pearson:** R^2, va entre 0 y 1 donde más cercano a 1 es mejor.
- Clasificación

Testear el modelo

Ahora tenemos una idea de cómo testear el modelo, utilizando el dataset de testeo. ¿Cómo podemos hacer esto? Hay varias métricas que podemos tener en cuenta:

- Regresión
  - **Root Mean Squared Error:** $RMSE = \sqrt{\frac{1}{n}\sum(t_i - y_i)^2}$.
  - **Pearson:** R^2, va entre 0 y 1 donde más cercano a 1 es mejor.
- Clasificación
  - **Precisión:** $\frac{TP}{TP+FP}$ qué porcentaje de predicciones positivas fueron correctas.

Testear el modelo

Ahora tenemos una idea de cómo testear el modelo, utilizando el dataset de testeo. ¿Cómo podemos hacer esto? Hay varias métricas que podemos tener en cuenta:

- Regresión
  - **Root Mean Squared Error:** $RMSE = \sqrt{\frac{1}{n}\sum(t_i - y_i)^2}$.
  - **Pearson:** R^2, va entre 0 y 1 donde más cercano a 1 es mejor.
- Clasificación
  - **Precisión:** $\frac{TP}{TP+FP}$ qué porcentaje de predicciones positivas fueron correctas.
  - **Recall:** $\frac{TP}{TP+FN}$ qué porcentaje de positivos fue identificado correctamente.

Testear el modelo

Ahora tenemos una idea de cómo testear el modelo, utilizando el dataset de testeo. ¿Cómo podemos hacer esto? Hay varias métricas que podemos tener en cuenta:

- Regresión
  - **Root Mean Squared Error:** $RMSE = \sqrt{\frac{1}{n}\sum(t_i - y_i)^2}$.
  - **Pearson:** R^2, va entre 0 y 1 donde más cercano a 1 es mejor.
- Clasificación
  - **Precisión:** $\frac{TP}{TP+FP}$ qué porcentaje de predicciones positivas fueron correctas.
  - **Recall:** $\frac{TP}{TP+FN}$ qué porcentaje de positivos fue identificado correctamente.
  - **Confusion Matrix:** visualizar los posibles errores y acieros del modelo: True Positives, False Positives, True Negatives, False negatives.

Testear el modelo

Ahora tenemos una idea de cómo testear el modelo, utilizando el dataset de testeo. ¿Cómo podemos hacer esto? Hay varias métricas que podemos tener en cuenta:

- Regresión
  - **Root Mean Squared Error:** $RMSE = \sqrt{\frac{1}{n}\sum(t_i - y_i)^2}$.
  - **Pearson:** R^2, va entre 0 y 1 donde más cercano a 1 es mejor.
- Clasificación
  - **Precisión:** $\frac{TP}{TP+FP}$ qué porcentaje de predicciones positivas fueron correctas.
  - **Recall:** $\frac{TP}{TP+FN}$ qué porcentaje de positivos fue identificado correctamente.
  - **Confusion Matrix:** visualizar los posibles errores y acieros del modelo: True Positives, False Positives, True Negatives, False negatives.
  - **Otros:** PR-AUC, ROC-AUC, log-loss, F1 Score.

Testear el modelo

Ahora tenemos una idea de cómo testear el modelo, utilizando el dataset de testeo. ¿Cómo podemos hacer esto? Hay varias métricas que podemos tener en cuenta:

- Regresión
  - **Root Mean Squared Error:** $RMSE = \sqrt{\frac{1}{n}\sum(t_i - y_i)^2}$.
  - **Pearson:** R^2, va entre 0 y 1 donde más cercano a 1 es mejor.
- Clasificación
  - **Precisión:** $\frac{TP}{TP+FP}$ qué porcentaje de predicciones positivas fueron correctas.
  - **Recall:** $\frac{TP}{TP+FN}$ qué porcentaje de positivos fue identificado correctamente.
  - **Confusion Matrix:** visualizar los posibles errores y acieros del modelo: True Positives, False Positives, True Negatives, False negatives.
  - **Otros:** PR-AUC, ROC-AUC, log-loss, F1 Score.

  Dependiendo el caso se elige qué optimizar. Ejemplo: modelo para detectar spam. Si se busca maximizar la cantidad de spam efectivos encontrados, se maximiza **Recall**, aunque esto pueda resultar en la clasificación errónea de algunos correos legítimos como spam. Si se busca minimizar los falsos positivo, teniendo en cuenta que algún spam podría no ser detectado, maximizar **Precision**.

Hiper Parámetros

Los **hiper parámetros** son ajustes que se le pueden hacer a los modelos. Afectan al rendimiento y funcionamiento de los modelos. Se tienen que definir al declarar el modelo, por lo que es necesario saber qué hay que poner para no perder tiempo entrenando con hiper parámetros incorrectos.
Algunos ejemplos son:

Hiper Parámetros

Los **hiper parámetros** son ajustes que se le pueden hacer a los modelos. Afectan al rendimiento y funcionamiento de los modelos. Se tienen que definir al declarar el modelo, por lo que es necesario saber qué hay que poner para no perder tiempo entrenando con hiper parámetros incorrectos.
Algunos ejemplos son:

- Polynomial Regression: *degree*, grado del polinomio que va a ajustar.

Hiper Parámetros

Los **hiper parámetros** son ajustes que se le pueden hacer a los modelos. Afectan al rendimiento y funcionamiento de los modelos. Se tienen que definir al declarar el modelo, por lo que es necesario saber qué hay que poner para no perder tiempo entrenando con hiper parámetros incorrectos.
Algunos ejemplos son:

- Polynomial Regression: *degree*, grado del polinomio que va a ajustar.
- Decision Tree: *max depth*, profundidad del árbol de decisión.

Hiper Parámetros

Los **hiper parámetros** son ajustes que se le pueden hacer a los modelos. Afectan al rendimiento y funcionamiento de los modelos. Se tienen que definir al declarar el modelo, por lo que es necesario saber qué hay que poner para no perder tiempo entrenando con hiper parámetros incorrectos.
Algunos ejemplos son:

- Polynomial Regression: *degree*, grado del polinomio que va a ajustar.
- Decision Tree: *max depth*, profundidad del árbol de decisión.
- Redes Neuronales: *learning rate*, *hidden layers*, *activation function*.

Hiper Parámetros

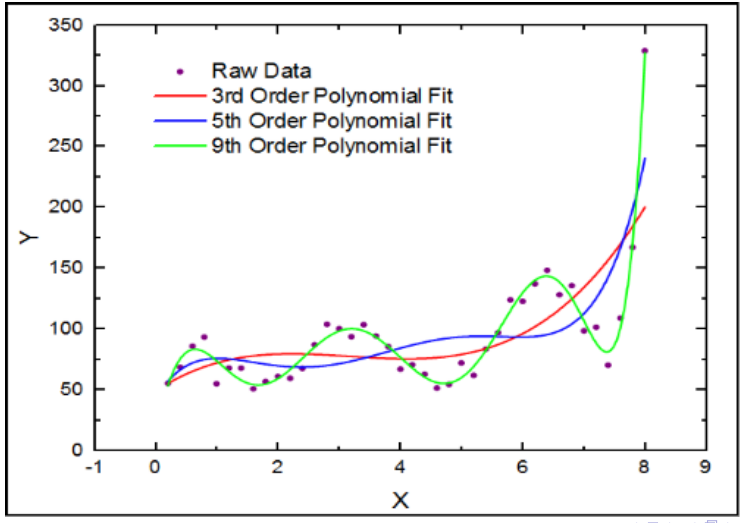
Los **hiper parámetros** son ajustes que se le pueden hacer a los modelos. Afectan al rendimiento y funcionamiento de los modelos. Se tienen que definir al declarar el modelo, por lo que es necesario saber qué hay que poner para no perder tiempo entrenando con hiper parámetros incorrectos.
Algunos ejemplos son:

- Polynomial Regression: *degree*, grado del polinomio que va a ajustar.
- Decision Tree: *max depth*, profundidad del árbol de decisión.
- Redes Neuronales: *learning rate*, *hidden layers*, *activation function*.

Hyperparameter Tunning: técninca para hacer una búsqueda de los mejores hiper parámetros para un modelo. Ejemplos: búsqueda aleatoria, búsqueda en grilla, optimización bayesiana.

Hiper Parámetros

Figura: Cómo afecta el grado del polinomio al ajuste.



Overfitting

Se dice que un modelo hace *overfitting* cuando ajusta muy bien a los datos de entrenamiento, pero no tanto a los de testeo. Esto es un problema, ya que el modelo no logra generalizar el comportamiento de los datos y no va a funcionar con datos nuevos.

- **Regularización:** evitar que los parámetros del modelo crezcan arbitrariamente. Por ejemplo limitar los coeficientes del polinomio.

Overfitting

Se dice que un modelo hace *overfitting* cuando ajusta muy bien a los datos de entrenamiento, pero no tanto a los de testeo. Esto es un problema, ya que el modelo no logra generalizar el comportamiento de los datos y no va a funcionar con datos nuevos.

- **Regularización:** evitar que los parámetros del modelo crezcan arbitrariamente. Por ejemplo limitar los coeficientes del polinomio.
- **Early Stopping:** validar durante el entrenamiento que el error en los datos de test no aumente, y frenar cuando esto suceda.

Notebook en Google Colab

Vamos a trabajar en un Notebook en Google Colab.

La idea del ejercicio de hoy es ver el funcionamiento de algunos modelos con un dataset de "juguete".

Notebook en Google Colab

Vamos a trabajar en un Notebook en Google Colab.

La idea del ejercicio de hoy es ver el funcionamiento de algunos modelos con un dataset de "juguete".

Vamos a probar los modelos **LinearRegression**, **PolynomialRegression**, **DecisionTreeRegressor** con la librería SKlearn

Notebook en Google Colab

Vamos a trabajar en un Notebook en Google Colab.

La idea del ejercicio de hoy es ver el funcionamiento de algunos modelos con un dataset de "juguete".

Vamos a probar los modelos **LinearRegression**, **PolynomialRegression**, **DecisionTreeRegressor** con la librería SKlearn

Ejercicios:

Notebook en Google Colab

Vamos a trabajar en un Notebook en Google Colab.

La idea del ejercicio de hoy es ver el funcionamiento de algunos modelos con un dataset de "juguete".

Vamos a probar los modelos **LinearRegression**, **PolynomialRegression**, **DecisionTreeRegressor** con la librería SKlearn

Ejercicios:

1. Graficar los datos del dataset que estamos creando.

Notebook en Google Colab

Vamos a trabajar en un Notebook en Google Colab.

La idea del ejercicio de hoy es ver el funcionamiento de algunos modelos con un dataset de "juguete".

Vamos a probar los modelos **LinearRegression**, **PolynomialRegression**, **DecisionTreeRegressor** con la librería SKlearn

Ejercicios:

1. Graficar los datos del dataset que estamos creando.
2. Dividir el dataset en train y test, graficar la división.

Notebook en Google Colab

Vamos a trabajar en un Notebook en Google Colab.

La idea del ejercicio de hoy es ver el funcionamiento de algunos modelos con un dataset de "juguete".

Vamos a probar los modelos **LinearRegression**, **PolynomialRegression**, **DecisionTreeRegressor** con la librería SKlearn

Ejercicios:

1. Graficar los datos del dataset que estamos creando.
2. Dividir el dataset en train y test, graficar la división.
3. Analizar el modelo **Linear Regression**, calcular *RMSE* en train y test.

Notebook en Google Colab

Vamos a trabajar en un Notebook en Google Colab.

La idea del ejercicio de hoy es ver el funcionamiento de algunos modelos con un dataset de "juguete".

Vamos a probar los modelos **LinearRegression**, **PolynomialRegression**, **DecisionTreeRegressor** con la librería SKlearn

Ejercicios:

1. Graficar los datos del dataset que estamos creando.
2. Dividir el dataset en train y test, graficar la división.
3. Analizar el modelo **Linear Regression**, calcular *RMSE* en train y test.
4. Analizar el modelo **Polynomial Regression**, calcular *RMSE* en train y test.

Notebook en Google Colab

Vamos a trabajar en un Notebook en Google Colab.

La idea del ejercicio de hoy es ver el funcionamiento de algunos modelos con un dataset de "juguete".

Vamos a probar los modelos **LinearRegression**, **PolynomialRegression**, **DecisionTreeRegressor** con la librería SKlearn

Ejercicios:

1. Graficar los datos del dataset que estamos creando.
2. Dividir el dataset en train y test, graficar la división.
3. Analizar el modelo **Linear Regression**, calcular *RMSE* en train y test.
4. Analizar el modelo **Polynomial Regression**, calcular *RMSE* en train y test.
5. Experimentar con el grado del polinomio, analizar cómo cambian los errores.

Notebook en Google Colab

Vamos a trabajar en un Notebook en Google Colab.

La idea del ejercicio de hoy es ver el funcionamiento de algunos modelos con un dataset de "juguete".

Vamos a probar los modelos **LinearRegression**, **PolynomialRegression**, **DecisionTreeRegressor** con la librería SKlearn

Ejercicios:

1. Graficar los datos del dataset que estamos creando.
2. Dividir el dataset en train y test, graficar la división.
3. Analizar el modelo **Linear Regression**, calcular *RMSE* en train y test.
4. Analizar el modelo **Polynomial Regression**, calcular *RMSE* en train y test.
5. Experimentar con el grado del polinomio, analizar cómo cambian los errores.
6. Analizar el modelo **Decision Tree Regressor**, calcular *RMSE* en train y test.