Universidad Popular del Cesar

Ciencia de Datos y BigData

Aprendizaje Supervisado

Dr. José Ramón Iglesias
DSP-ASIC BUILDER GROUP
Director Semillero TRIAC
Ingenieria Electronica
Universidad Popular del Cesar

Cuarta Clase

Universidad Popular del Cesar

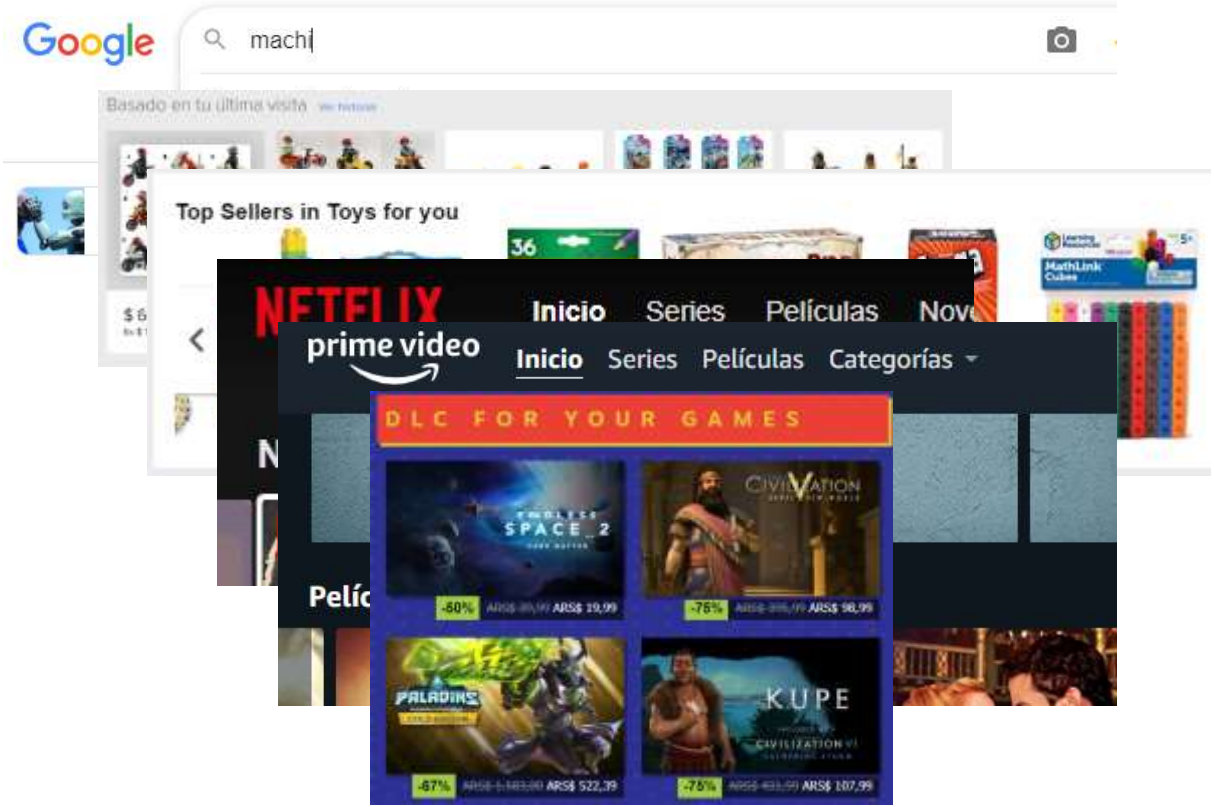
Ciencia de Datos y BigData

Cuarta Clase: 02/07/2022

- Introducción al ML
- Etapas en la aplicación del ML
- Aprendizaje supervisado.
  - Repaso: Regresión Lineal y Polinomial, Regresión Logística, Naive Bayes.
  - Repaso: Perceptrón.
- Support Vector Machines.
  - SVC/SVR. Datos no linealmente separables. Función de costo.
- Ensemble learning.
  - Repaso: Decision Trees
  - Random Forest, Bagging, Boosting, Voting.
- Redes neuronales.
  - Perceptrón multicapa.
- Sistemas de recomendación.
  - Filtrado colaborativo.
- "Buenas" prácticas en la aplicación de ML

Sistemas de recomendación (Introducción)

Sistemas de recomendación: ejemplos



- Google
- MercadoLibre
- Amazon
- Netflix
- PrimeVideo
- Steam
- Facebook
- Instagram
- …

Sistemas de Recomendación: objetivos

Los sistemas de recomendación se centran en:

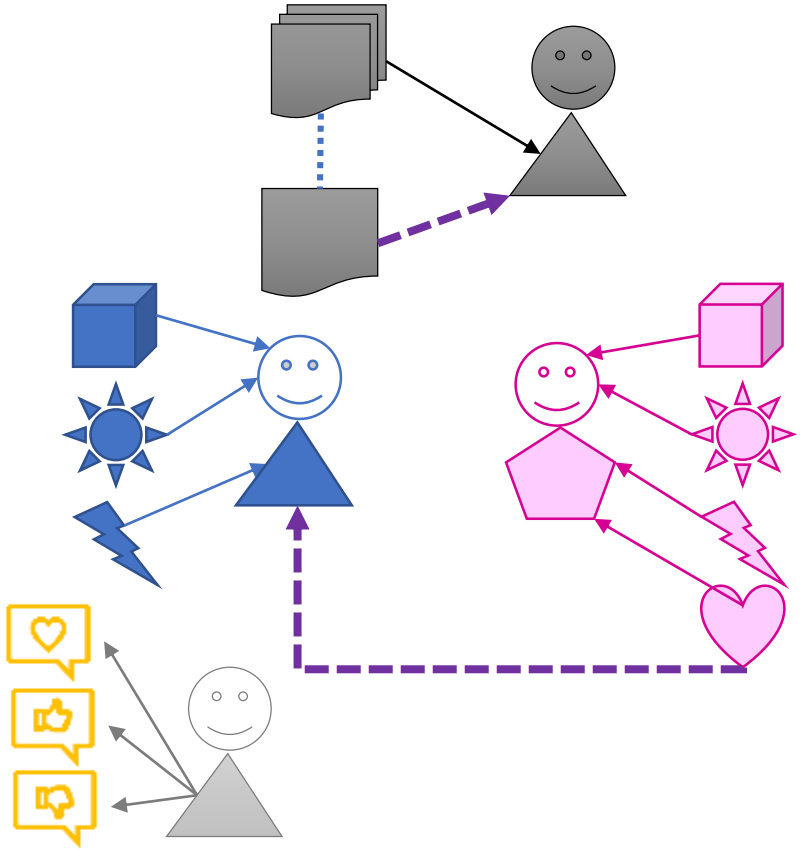
- *items,* para eCommerce
- *contenido,* para eLearning, noticias, streaming, etc.
- *links,* para navegación

en general, se habla de ítems como término genérico.

El objetivo general de estos sistemas es el de guiar a los usuarios a tomar decisiones.

Tipos de Sistemas de Recomendación

- **Basados en contenido:** tratan el problema de manera específica para cada usuario y "aprenden" una clasificación de lo que a un usuario le gusta basado en las características (contenido) de un ítem.
- **Filtros colaborativos:** se basan en la idea que los usuarios que coinciden en el pasado lo harán en el futuro y que les gustarán ítems similares.
- **Basados en el conocimiento:** el usuario proporciona explícitamente parámetros del producto que quiere
- **Basados en cuestiones demográficas**
- **Sistemas híbridos**



Filtros colaborativos

Combina usuarios con intereses similares:

- Se podría requerir de muchos usuarios para que las posibilidades de encontrar un par con intereses en común.
- Tiene que existir un método sencillo para reflejar los intereses de los usuarios.
- Se necesita de un algoritmo eficiente para combinar a los usuarios con gustos similares.

Filtros colaborativos: cómo funcionan?

INPUT

Usuarios generan puntuaciones a un conjunto de ítems (implícita o explícitamente): matriz de puntuaciones usuario-ítem

OUTPUT

- Predicción (numérica) indicando cuánto a un usuario le gusta un ítem
- Una lista con N ítems recomendados por usuario

Ciencia de Datos y BigData

Filtros colaborativos: Vecindarios basado en usuarios

	Item1	Item2	Item3	Item4	Item5
Claudia	5	3	4	4	
Usuario1	3	1	2	3	3
Usuario2	4	3	4	3	5
Usuario3	3	3	1	5	4
Usuario4	1	5	5	2	1

Similitud entre usuarios

Coeficiente de correlación (Pearson correlation)

$$sim(a,b) = \frac{\sum_{p \in P}(r_{a,p} - \bar{r}_a)(r_{b,p} - \bar{r}_b)}{\sqrt{\sum_{p \in P}(r_{a,p} - \bar{r}_a)^2}\sqrt{\sum_{p \in P}(r_{b,p} - \bar{r}_b)^2}}$$

donde:

- *a* y *b* son usuarios;
- $r_{a,p}$ es la puntuación del usuario *a* al ítem *p;*
- y *P* es el conjunto de ítems que ya han sido puntuados por *a* y *b* .

Similitud entre usuarios

$$sim(a,b) = \frac{\sum_{p \in P}(r_{a,p} - \bar{r}_a)(r_{b,p} - \bar{r}_b)}{\sqrt{\sum_{p \in P}(r_{a,p} - \bar{r}_a)^2}\sqrt{\sum_{p \in P}(r_{b,p} - \bar{r}_b)^2}}$$

	Item1	Item2	Item3	Item4	Item5
Claudia	5	3	4	4	
Usuario1	3	1	2	3	3
Usuario2	4	3	4	3	5
Usuario3	3	3	1	5	4
Usuario4	1	5	5	2	1

sim(Claudia, Usuario1) = 0.8392

sim(Claudia, Usuario2) = 0.6063

sim(Claudia, Usuario3) = 0

sim(Claudia, Usuario4) = -0.7681

Similitud entre usuarios

Coeficiente de correlación

$$pred(a,p) = \bar{r}_a + \frac{\sum_{b \in N} sim(a,b)\left(r_{b,p} - \bar{r}_b\right)}{\sum_{b \in N} sim(a,b)}$$

donde:

- N es el conjunto de "vecinos"
- p es un ítem

$$pred(Claudia, item5) = 4 + \frac{0{,}85\ (3 - 2{,}4) + 0{,}7\ (5 - 3{,}8)}{0{,}85 + 0{,}7} = 4{,}87 \cong 5$$

Filtros colaborativos: Vecindarios basado en ítems

Se usan las semejanzas entre los ítems (y no entre los usuarios) para hacer las predicciones.

Por ejemplo, busquemos los ítems parecidos al *Item5*. Ahora usamos las puntuaciones que Claudia le asignó a tales ítems para darle un valor al *Item5*.

	Item1	Item2	Item3	Item4	Item5
Claudia	5	3	4	4	
Usuario1	3	1	2	3	3
Usuario2	4	3	4	3	5
Usuario3	3	3	1	5	4
Usuario4	1	5	5	2	1

Similitud entre ítems

Distancia (ajustada) del coseno

$$sim(a,b) = \frac{\sum_{u \in U}(r_{u,a} - \bar{r}_u)(r_{u,b} - \bar{r}_u)}{\sqrt{\sum_{u \in U}(r_{u,a} - \bar{r}_u)^2}\sqrt{\sum_{u \in U}(r_{u,b} - \bar{r}_u)^2}}$$

donde:

- *a* y *b* son ítems;
- $r_{u,a}$ es la puntuación del usuario *u* al ítem *a;*
- y *U* es el conjunto de usuarios que han puntuado a los ítems *a* y *b*.

Similitud entre ítems

$$pred(u, item_j) = \bar{r}_u + \frac{\sum_{item_i \in N} sim(item_i, item_j)\, r_{u,item_i}}{\sum_{item_i \in N} sim(item_i, item_j)}$$

donde:

- N es el conjunto de "vecinos"
- u es un usuario

$$pred(Claudia, item_5) = \bar{r}_{Claudia} + \frac{\sum_{item_i \in N} sim(item_i, item_5)\, r_{Claudia,item_i}}{\sum_{item_i \in N} sim(item_i, item_j)}$$

Filtros colaborativos: Preprocesamiento

Para que estos sistemas de recomendación sean escalables, es necesario aprender el modelo "offline".

- Calcular offline todos las similitudes entre pares de objetos
- Calcular la predicción en tiempo real (pred de la slide anterior)
- El vecindario (N) suele ser bastante pequeño (el usuario puntúa pocos objetos)

Este preprocesamiento funciona en CF por ítems (y no en CF por usuarios): las similitudes entre ítems suelen ser más estables que entre usuarios.

Filtros colaborativos: Preprocesamiento

Requerimientos de memoria: si consideramos *n* ítems, tendremos n^2 similitudes.

En la práctica, la matriz es dispersa (muchos pares de ítems que no tienen similitud).

Se suelen usar reducciones fijando un umbral mínimo de "co-ratings": se eliminan ítems que tienen pocas puntuaciones comunes, usando al menos n' usuarios.

Se puede también limitar el tamaño del vecindario (*N*).

Filtros colaborativos: Problemas

Cold start problem:

- ¿Cómo recomendamos nuevos ítems?
- ¿Qué le recomendamos a usuarios nuevos?

Soluciones inmediatas:

- Forzar a los usuarios a puntuar un conjunto de objetos (-1)
- Emplear otro método para la estos casos (basado en contenido, información demográfica o recomendaciones no personalizadas)

Filtros colaborativos: Problemas

Problemas con los vecindarios:

El conjunto de usuarios ítems similares puede ser muy pequeño. Lo que produce malas predicciones.

Alternativas:

- CF recursivo
  - Transitividad entre vecinos

Demo time (demo_11_recommender_systems)

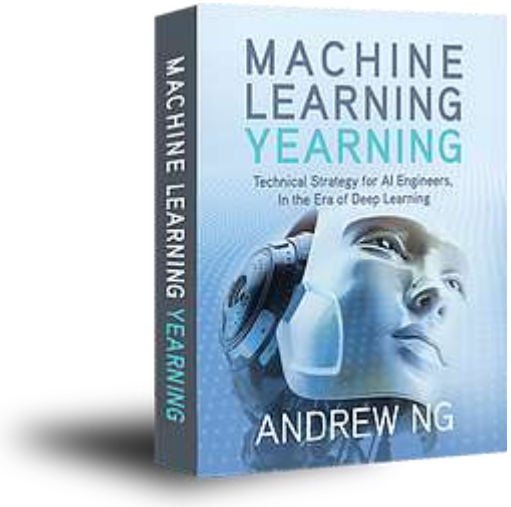
Estrategias para Machine Learning

Estrategias para Machine Learning

Referencias:

- Andrew Ng. “Machine Learning Yearning”. Draft, 2018.

- Experiencia



Honest Machine Learning

Google:

- Cantidades astronómicas de datos
- Ejércitos de ingenieros
- Hectáreas de GPUs, memoria, etc.

Vos:

- 1500 datos ruidosos
- Una fracción de tu tiempo
- Una notebook del año 2016

Estrategias para Machine Learning

Estrategias para Machine Learning

Queremos aplicar ML sobre un problema, de manera rápida y exitosa.

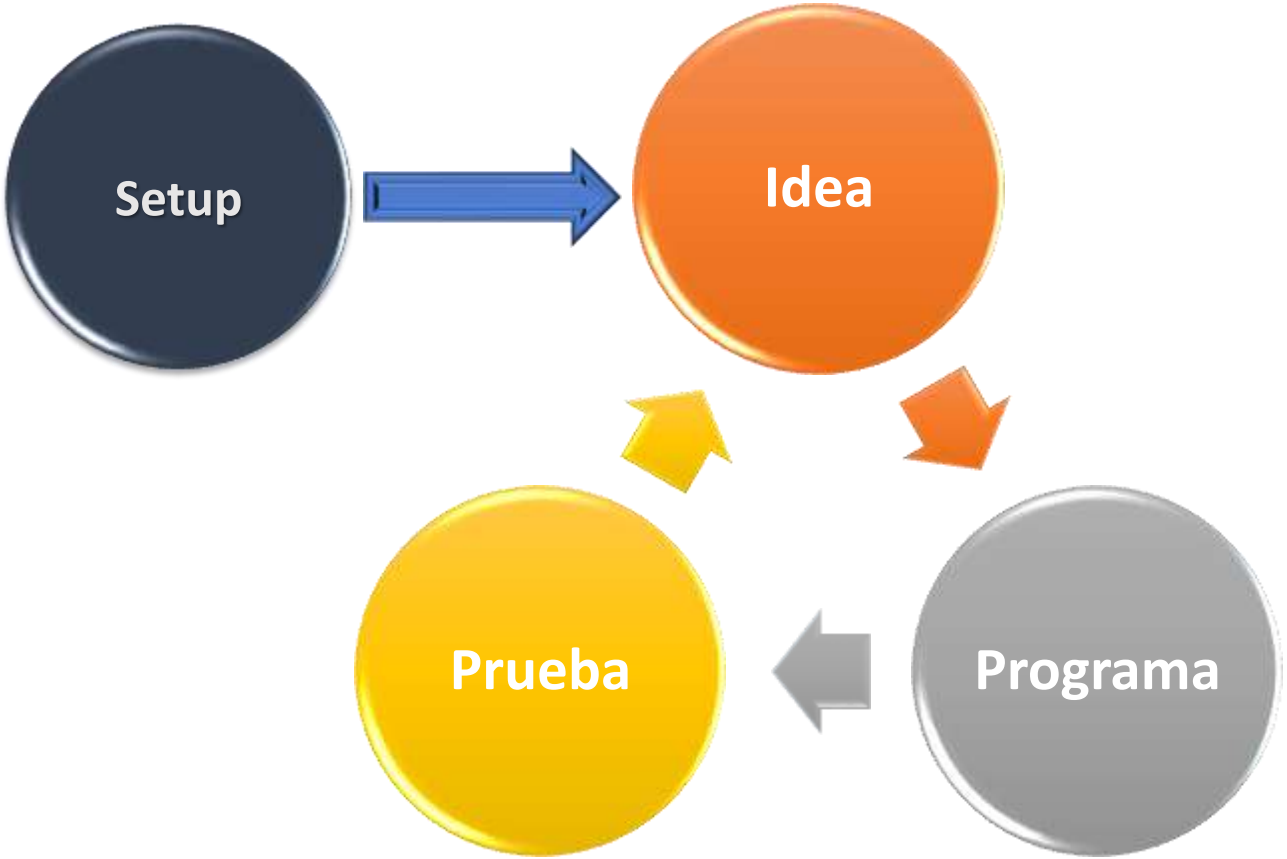
Lamentablemente, nuestro algoritmo anda mal. ¿Qué hacer?

Opciones:

- Recolectar más datos, o datos más diversos
- Preprocesamiento: ingeniería de features, reducción de dimensionalidad, normalización, etc.
- Modelos de clasificación
- Parámetros/arquitectura: modelos más simples, o más complejos
- Entrenamiento

Hay que saber elegir!!

Método iterativo



Setup: Preparación de los Conjuntos de Datos

- **Training**: Entrenamiento
- **Validación**: Para ajustar hiperparámetros, seleccionar features, analizar errores, etc.
- **Prueba**: Para obtener números finales de evaluación. **Nunca** para tomar decisiones.

Validación y prueba **deben** responder a la misma distribución.

Entrenamiento: no necesariamente.

Setup: Tamaño de los datasets

- ✓ Machine Learning clásico:
  - ✓ División de los datos ~70/10/20
- ✓ Grandes cantidades de datos:
  - ✓ unos pocos miles para desarrollo/prueba.
- ✓ Resolución: El tamaño del dataset indica la “resolución” de la precisión
  - ✓ 100 elementos: 1%
  - ✓ 500: 0.2%
  - ✓ 1000: 0.1%
  - ✓ 10000: 0.01%

Setup: Métricas

- ***Precisión***:
  - Poco informativa para problemas desbalanceados
- ***Precision/recall/F1***:
  - ***Binaria***: Focalizar el problema en una de las dos clases.
  - ***Multiclase***: Permite regular la importancia de cada clase (weighted macro-average).
- ***Balanced Accuracy (Recall Macro Average)***
- ***ROC AUC y AUCPR***:
  - Más expresiva: evalúa probs/scores asignadas a todas las clases, no la predicción.

Setup: Métricas de optimización vs. satisfacción

- Establecer una **única** métrica numérica, cuyo objetivo es optimizar.
- Métricas secundarias:
  - Velocidad
  - Instancias sensibles que no pueden ser etiquetadas incorrectamente.
  - Valores mínimos de precision/recall para clases específicas.
- Definir criterios de “satisfacción” para las métricas secundarias.

Setup: Baselines

- Clasificadores “bobos” para calcular valores mínimos para las métricas.
  - Clase mayoritaria
  - Random uniforme
  - Random respetando distribución

- A veces también se pueden calcular upper bounds teóricas.

Setup: Rápido!

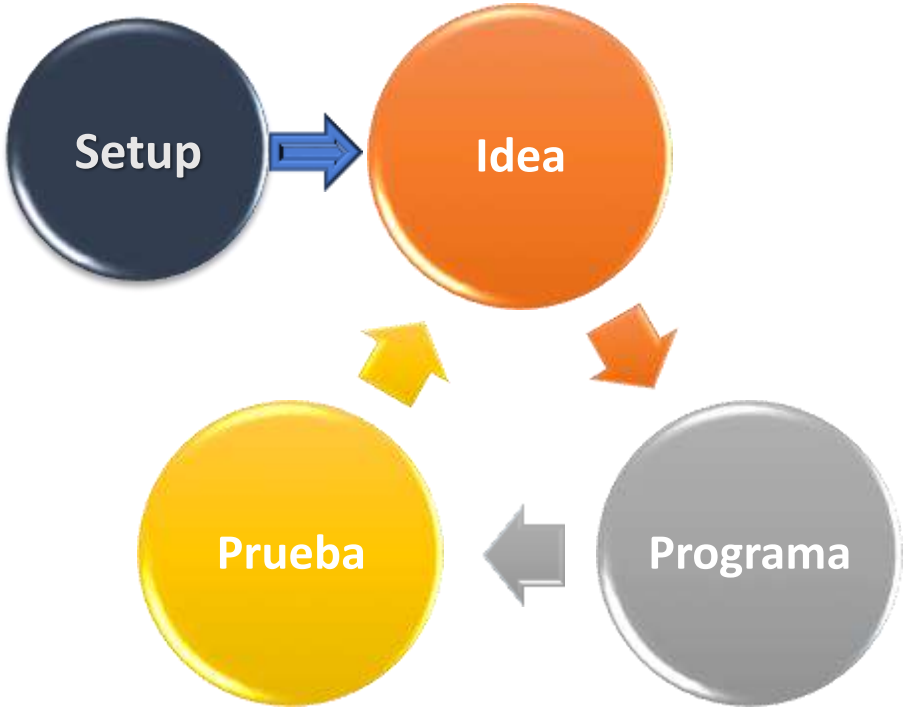
- Definir rápidamente los conjuntos de datos y las métricas objetivo.
- Permite iniciar el ciclo iterativo.
- Luego, los resultados y su análisis pueden indicar la necesidad de modificaciones:
  a. Los datos no reflejan la aplicación real: actualizar dev/test
  b. Overfitting en dev: se iteró muchas veces, actualizar dev.
  c. Las métricas no reflejan los objetivos.

Setup: Registro de Experimentos

- Historial de experimentos realizados.
- Registrar información necesaria para la reproducibilidad:
  - Fecha del experimento
  - Configuración del modelo
  - Resultado de las evaluación

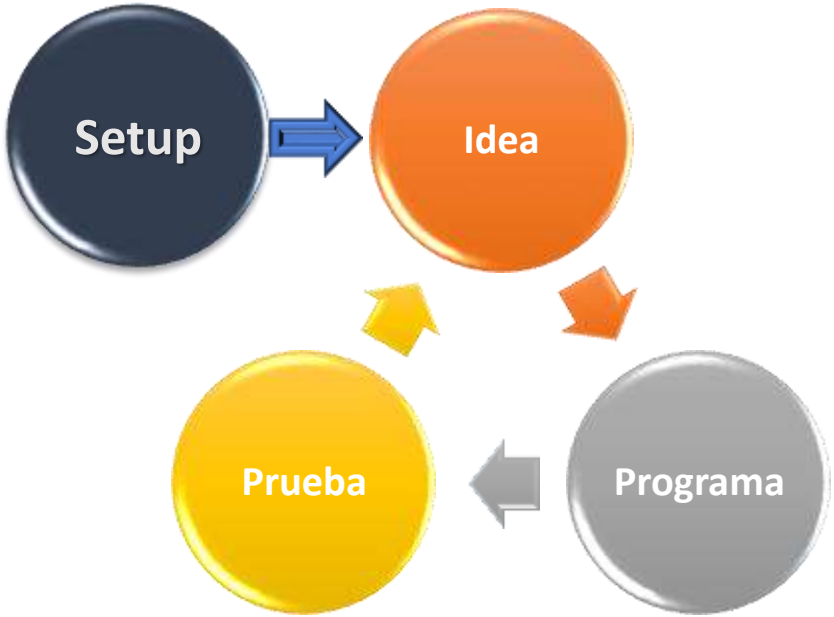
Método Iterativo
Primera Iteración: Sistema Básico

- No empezar tratando de construir el sistema perfecto.
- Construir y entrenar un sistema básico lo más rápido posible.
- Evaluarlo y estudiarlo para decidir en qué direcciones avanzar.



Primera Iteración: Modelo de Clasificación

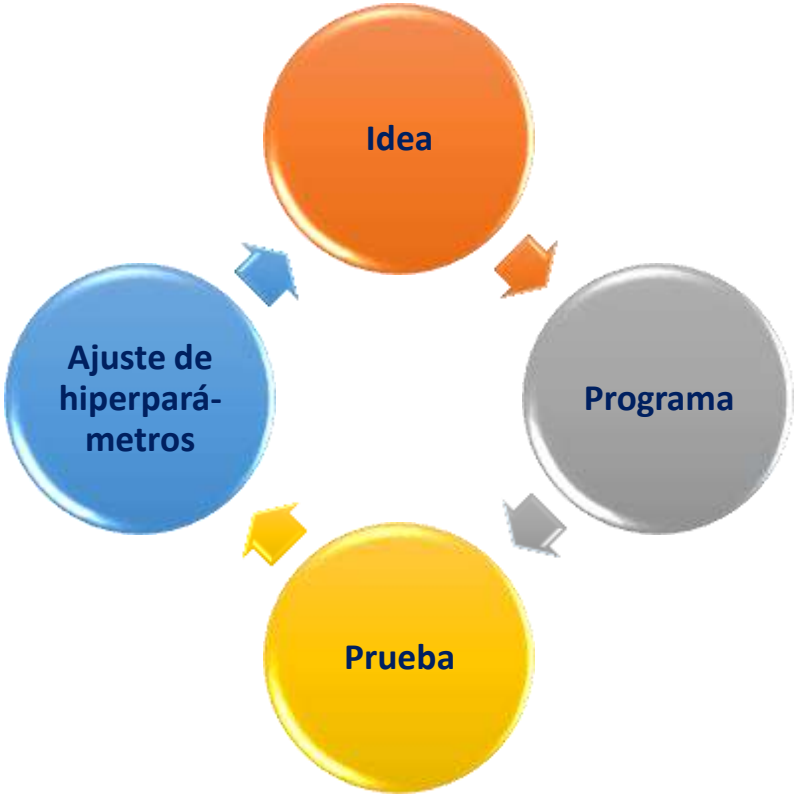
- Se pueden probar varios modelos de clasificación
- Empezar eligiendo el que mejor ande sin ninguna configuración.
- No empezar **NUNCA** con redes neuronales.
- No casarse con un único modelo.



Ajuste de Hiperparámetros

Opciones:

- Búsqueda manual.
- Búsqueda exhaustiva (grid-search): todas las combinaciones posibles de valores.
- Aleatoria (randomized): sampleando valores o combinaciones.
- Development vs. Cross Validation



Ajuste de Hiperparámetros

Estrategias:

- ¡Leer documentación!
- Empezar con búsqueda manual. Elegir parámetros más relevantes.
- Búsqueda aleatoria, con un espectro mayor.
- Seguir con búsqueda exhaustiva. Probar pocas combinaciones.
- Iterar.
- Guardar mejores configuraciones (no sólo la mejor).

Evaluación: Sesgo y Varianza

- **Sesgo (bias):** Error en el conjunto de entrenamiento.
- **Varianza (variance):** Error en el conjunto de development.
- **Error total:** bias + variance.
- Hacer Machine Learning = Bajar el error total.

Evaluación: Sesgo

- **Sesgo alto:** el clasificador **ni siquiera** es capaz de aprender los datos de entrenamiento.
  - Anda peor que un sistema que memoriza los puntos de entrenamiento.
  - ¿Cuánto quiere decir alto?
  - Depende del problema y de los valores a los que aspiramos.
  - Normalmente el sesgo **se puede reducir a cero. Se puede pero no necesariamente se quiere.**

PRIMER OBJETIVO DEL ML: CONTROLAR EL SESGO.

Evaluación: Reducción de Sesgo

- El sistema no logra aprender el conjunto de entrenamiento. No es lo suficientemente “expresivo” (underfitting).

- Soluciones:
  - **Modelo más grande**: agregar parámetros, capas, componentes, etc.
  - **Modelo menos regularizado**: salir del underfitting.
  - **Características más expresivas**: más dimensiones.
  - **Modelo nuevo**: clasificador diferente, otra arquitectura.

Evaluación: Varianza

- **Sesgo bajo control:** Puedo hacerlo tan bajo como quiera.

- **Varianza alta:** No generaliza. No “aprende”. Memoriza. (overfitting)

- ¿Cuánto es varianza alta?
  - Nuevamente, depende del problema y de nuestros objetivos.
  - Con el sesgo controlado, la varianza es directamente proporcional al error total.
  - Con el sesgo controlado, **varianza cero = sistema perfecto.**

- **NUEVO OBJETIVO: Baja**r la varianza tanto como se pueda = **HACER ML.**

Evaluación: Reducción de Varianza

- ✓ El sistema no logra generalizar a partir del conjunto de entrenamiento.
- ✓ Posibles soluciones:
  - ✓ **Más datos de entrenamiento**. No hay de dónde aprender.
  - ✓ **Mejores features**: Facilitar al modelo el acceso a información valiosa.
  - ✓ **Bajar expresividad**: Regularización, early stopping, menos params., etc.
  - ✓ **Modelo nuevo**: clasificador diferente, otra arquitectura.

Análisis de Error

- ¿En qué se equivoca el modelo?
- Inspeccionar elementos mal clasificados.
- ¿Porqué se clasifica mal?
  - Ver la probabilidad / score de la clase correcta.
  - Ver features activos. Ver valores cercanos en instancias de entrenamiento.
  - Ver qué modificaciones del elemento hacen que se clasifique bien.
- Inspeccionar elementos **peor** clasificados (en

Análisis de Error

- Hacer una lista de ejemplos mal clasificados. (e.g., 50 de dev)
- Inspeccionar cada ejemplo. Identificar fuentes de error.
- Para cada fuente de error, identificar importancia y costo estimado.

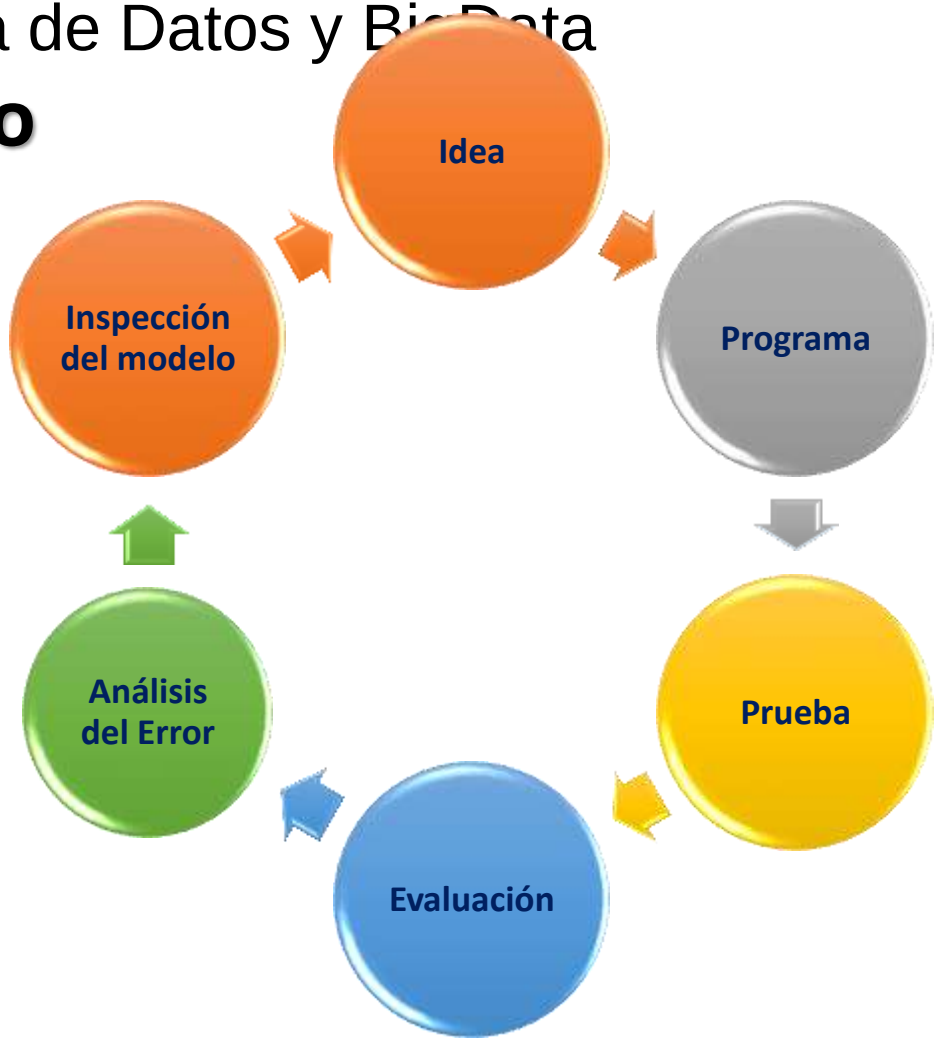
Audio clip	Loud background noise	User spoke quickly	Far from microphone	Comments
1	✔			Car noise
2	✔		✔	Restaurant noise
3		✔	✔	User shouting across living room?
4	✔			Coffeeshop
% of total	75%	25%	50%	

Análisis de Error (Error Analysis)

- Subdivisón de development:
  - Eyeball dev set (~100 instancias)
  - Blackbox dev set (el resto)
  - Rotar cada tanto!
- Errores en el dataset:
  - Evaluar su impacto.
  - Si es importante, corregir en **todos** los datasets.

Inspección del Modelo

- Estudiar los parámetros del modelo una vez aprendido.
- Features más influyentes para cada clase.
- Fronteras de decisión.

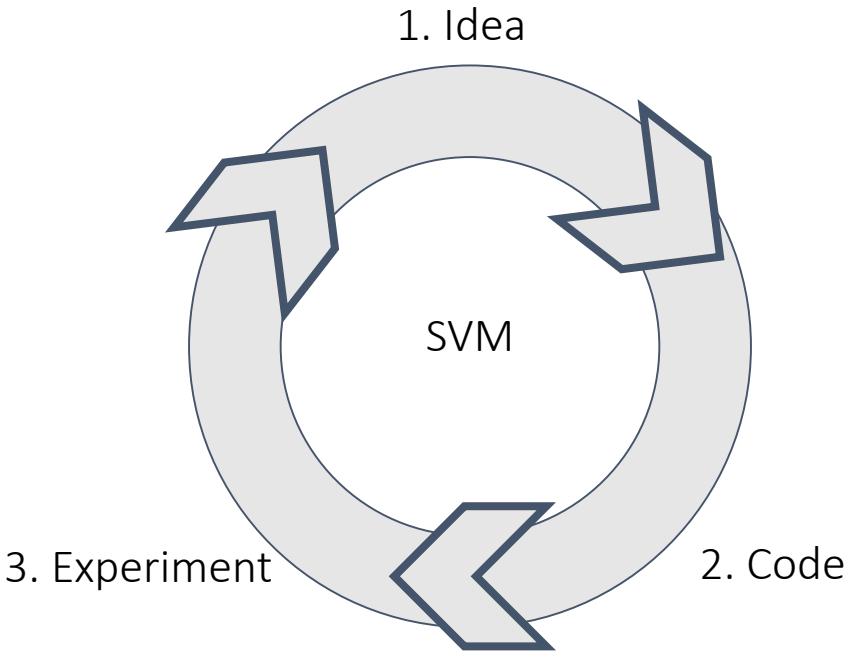


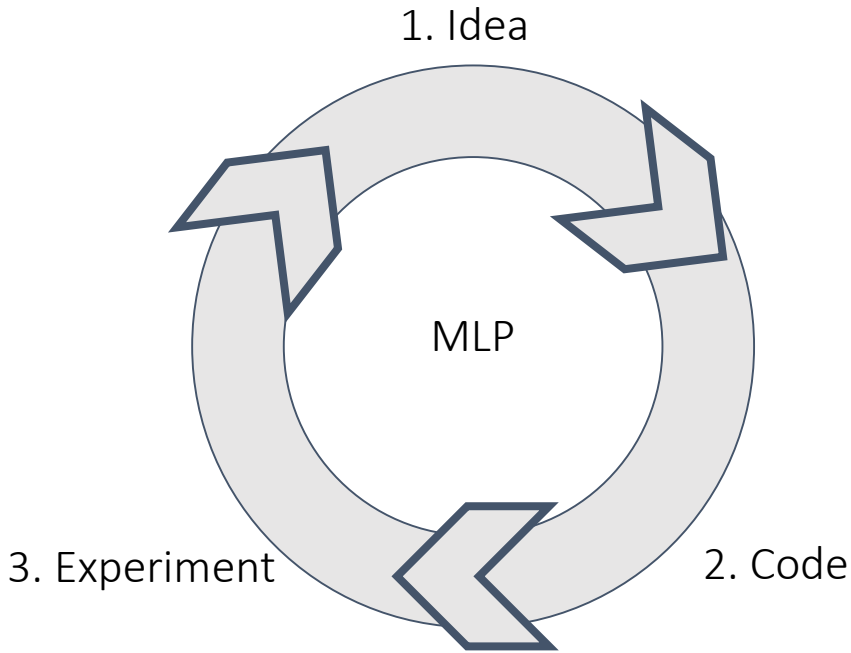
Inspección del Modelo

- Modelos fácilmente inspeccionables:
  - Decision Trees
  - Naive Bayes: probabilidad de cada feature dada la clase (y prior de la clase)
  - Logistic Regressions: score de cada feature para cada clase (y bias o intercept)

- Más complicado:
  - Random Forests: son muchos árboles para ver!
  - SVMs: ver con qué features está más alineado el hiperplano.
  - Redes Neuronales: usar inputs para ver cómo reacciona la red.

Fork (Bifurcación)





Bifurcación

- Empezar a mantener dos o más sistemas diferentes en simultáneo.
- Cada uno tiene su ciclo de experimentación.
- Con el tiempo, las configuraciones divergen.
- Ejemplo:
  - SVM / LR
  - Red neuronal: MLP / RNN / CNN

Retrospectivas

- **Revisar ideas previas**, tanto las aceptadas como las rechazadas.
- **Ablation Tests**: medir el impacto de cada componente del sistema actual.
- **Hiperparameter retuning**: Volver a hacer ajuste de hiperparámetros
- **Reconsiderar** viejas ideas

Aumentación de Datos (Data Augmentation)

- Generar datos artificiales en base a los datos que tenemos.
- Las transformaciones deben preservar las etiquetas
- Imágenes: rotación, escala, espejado, cambio de color, etc.
- Texto: más dificil! sinónimos, traducción bidireccional, etc.

Ingeniería del Software en Machine Learning

Diferencias con los sistemas tradicionales

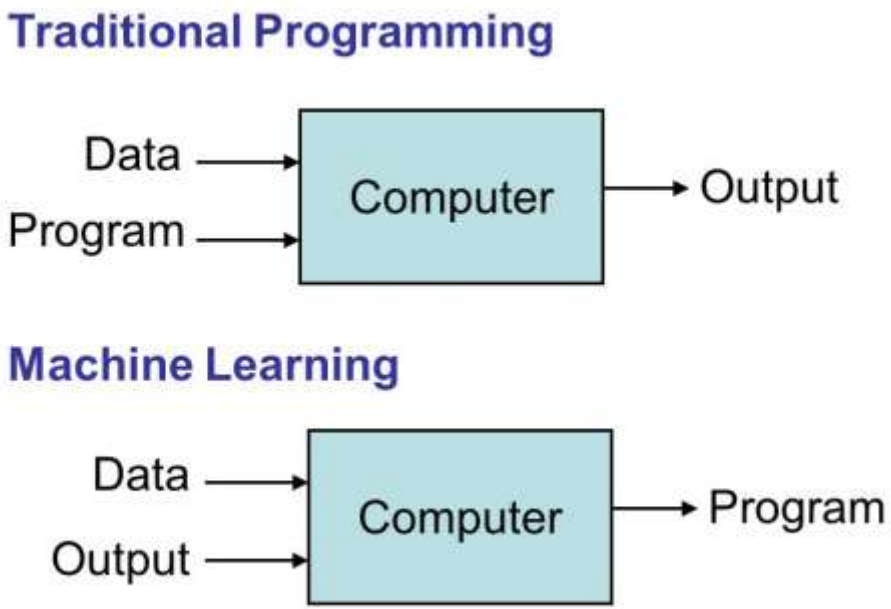
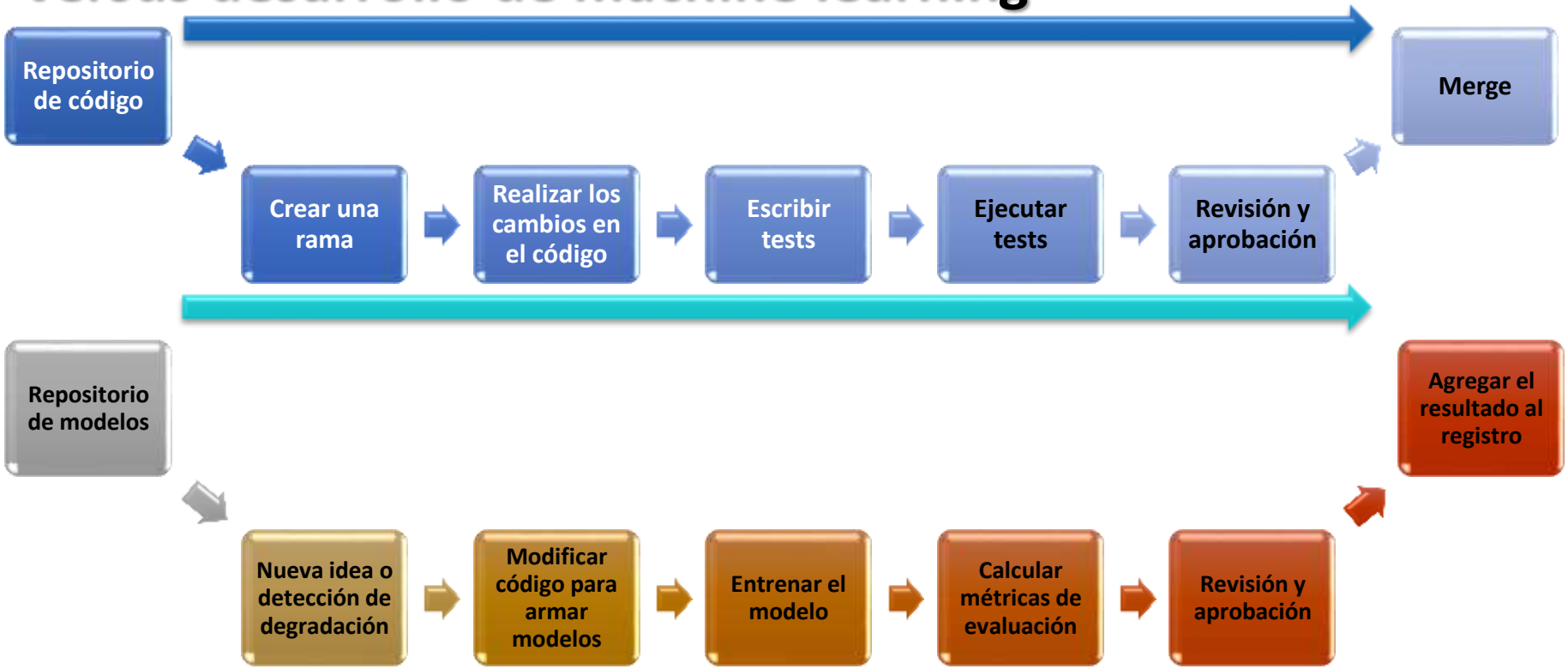


Image by Pedro Domingos

Workflow de desarrollo de software tradicional vérsus desarrollo de machine learning



Testing en machine learning?

- **Model evaluation:** métricas, gráficos, estadísticas que resumen el comportamiento del modelo en los conjuntos de dev/test.
- **Model testing:** verificación explícita de comportamientos que esperamos de nuestro modelo
  - **Pre-train tests**
  - **Post-train tests**

Testing del Modelo: **tests de pre-entenamiento**

Tests que se pueden ejecutar sin la necesidad de entrenar el modelo sobre todo el dataset.

- **Tamaño del output del modelo:** verificar que la longitud del vector? de salida del modelo esté alineado al tamaño de los labels del dataset
- **Rangos de salida:** validar tipos y rangos de valores según las expectativas. Por ejemplo, si el output es una probabilidad, asegurarse de que la suma será 1
- **La ejecución del modelo tiene sentido:** por ejemplo, asegurarse de que los pasos en gradient descent produzcan un descenso en el costo
- **Verificar data leakage**

Testing del Modelo: tests de post-entrenamiento

Tests sobre resultados del modelo ya entrenado.

Invariance Tests

- Describen perturbaciones en la entrada del modelo que no deberían modificar la salida.
- Similar al concepto de *data augmentation*, donde se modifica la entrada durante el training pero se preservan las etiquetas.
- Ejemplo: para un modelo de análisis de sentimiento:
  - Juan es un buen tipo
  - José es un buen tipo

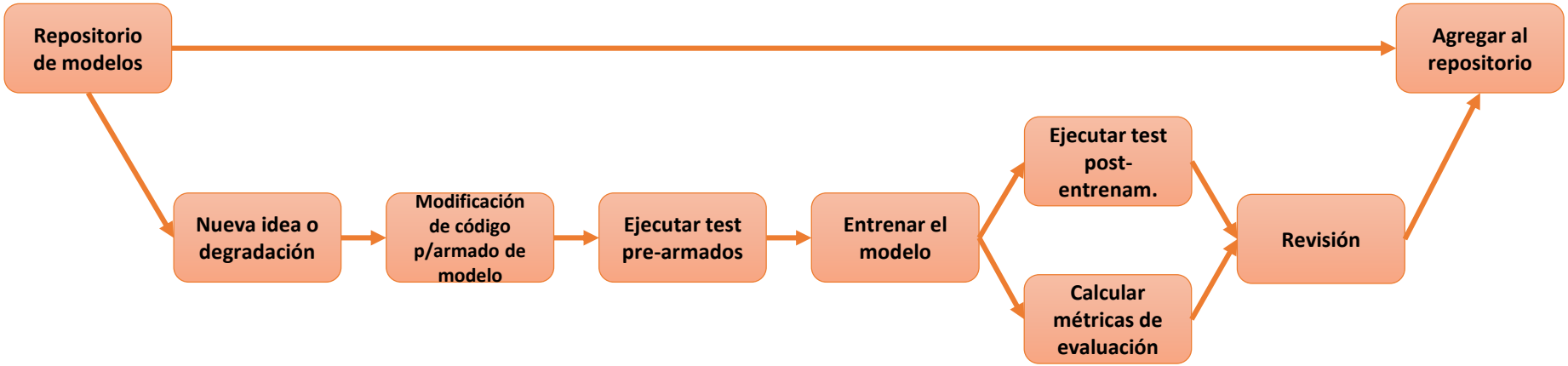
Testing del modelo: tests de post-entrenamiento

Tests sobre resultados del modelo ya entrenado.

Tests de expectativa direccional

- Nos permiten definir perturbaciones en la entrada que deberían afectar de cierta forma la salida del modelo.
- Por ejemplo, para un modelo que predice el precio de inmuebles:
  - Aumentar el número de habitaciones que tiene una casa, manteniendo constante el resto de sus atributos, no debería causar un descenso de su precio
  - Reducir la superficie cubierta (m^2) de una propiedad, no debería aumentarle su valor

Nuevo workflow de desarrollo de machine learning



Demo time (demo_12_ML_practices)

FIN