Aprendizaje Supervisado

Cristian Cardellino

Cuarta Clase

Temario de la Clase

- ¿Qué es aprendizaje supervisado?
- Aprendizaje supervisado.
  - Repaso: Regresión Lineal y Polinomial, Regresión Logística, Naive Bayes.
- Support Vector Machines.
  - Repaso: Perceptrón.
  - 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.
- Prácticas de reproducibilidad

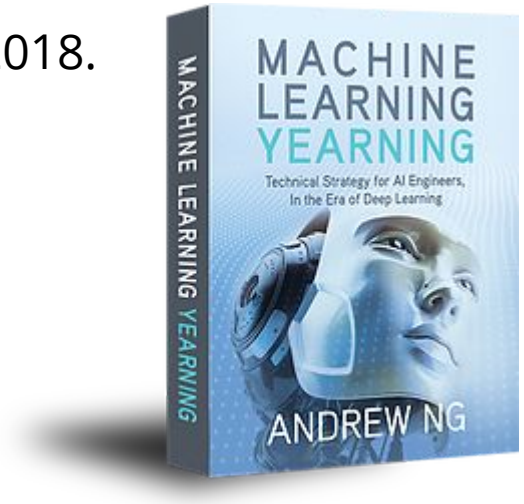
Estrategias para Machine Learning

Estrategias para Machine Learning

Referencias:

- Andrew Ng. “Machine Learning Yearning”. Draft, 2018. http://www.mlyearning.org/

- Experiencia personal.



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

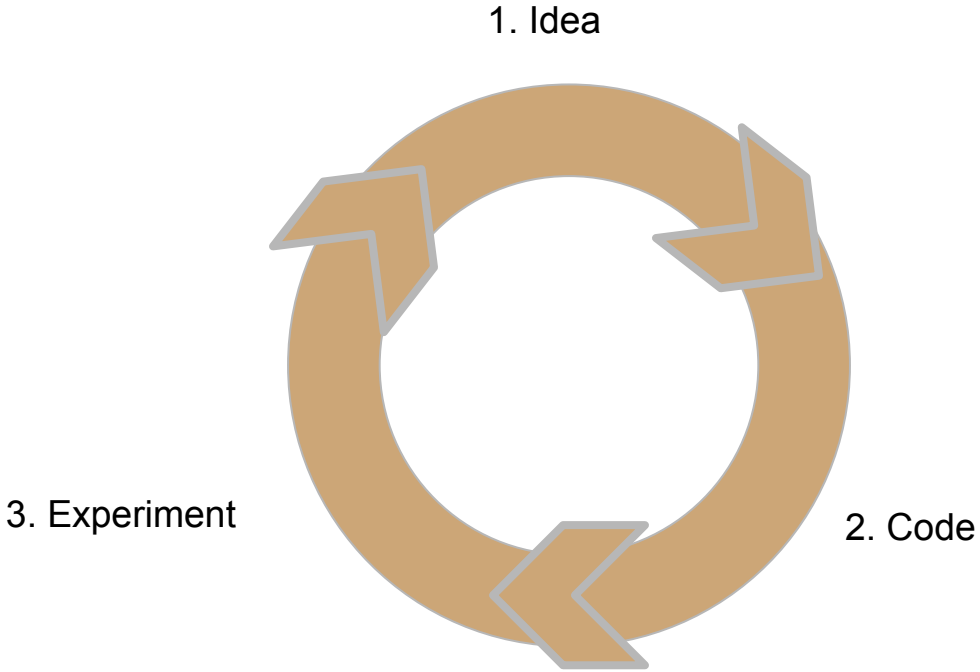
Queremos aplicar ML sobre un problema, de manera rápida y exitosa.

Lamentablemente, nuestro algoritmo anda mal. ¿Qué hacer? Opciones:

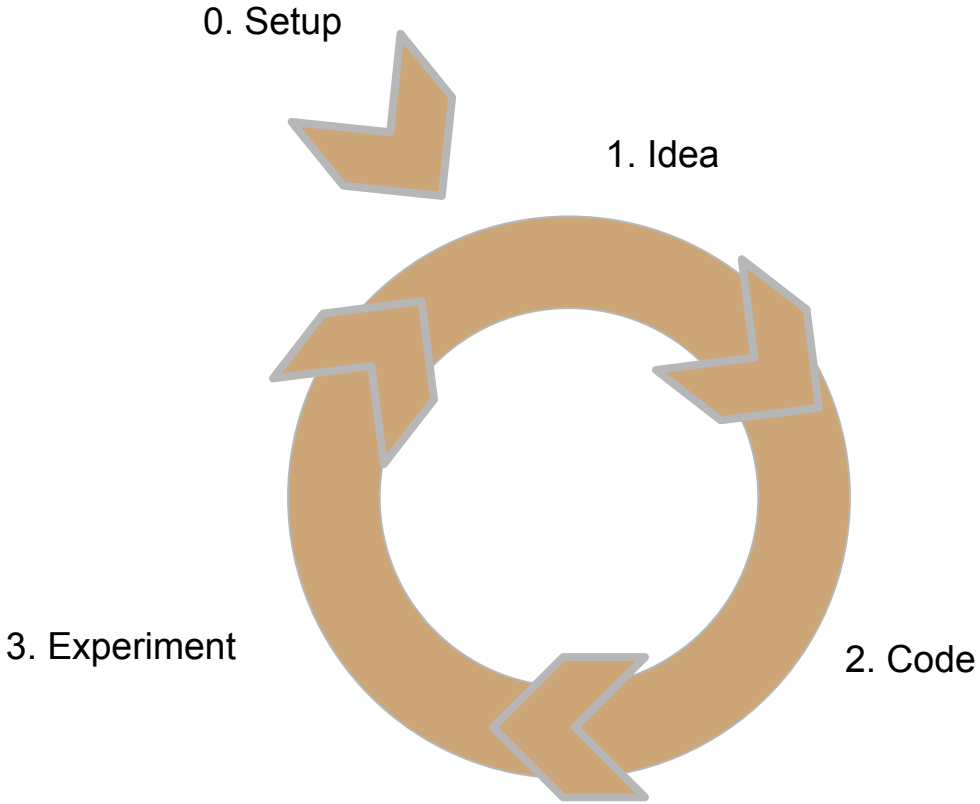
- Recolectar más datos, o datos más diversos
- Preprocesamiento: ingeniería de features, reducción de dimensionalidad, normalización, etc.
- Modelos de clasificación
- Parámetros / arquitectura: modelos más simples, o más complejos
- Entrenamiento

Hay que saber elegir!!

Método iterativo



Setup



Setup: Preparación de los Conjuntos de Datos

- Training: Entrenamiento
- Development (validación): Para ajustar hiperparámetros, seleccionar features, analizar errores, etc.
- Test: Para obtener números finales de evaluación. **Nunca** para tomar decisiones.
- Dev y test **deben** responder a la misma distribución.
- Train no necesariamente.

Setup: Tamaño de los datasets

- Machine Learning clásico:
  - Split ~70/10/20
- Grandes cantidades de datos:
  - unos pocos miles para dev/test.
- Resolución: El tamaño del dataset indica la “resolución” de la accuracy
  - 100 elementos: 1%
  - 500: 0.2%
  - 1000: 0.1%
  - 10000: 0.01%

Setup: Métricas

- Accuracy:
    - 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.

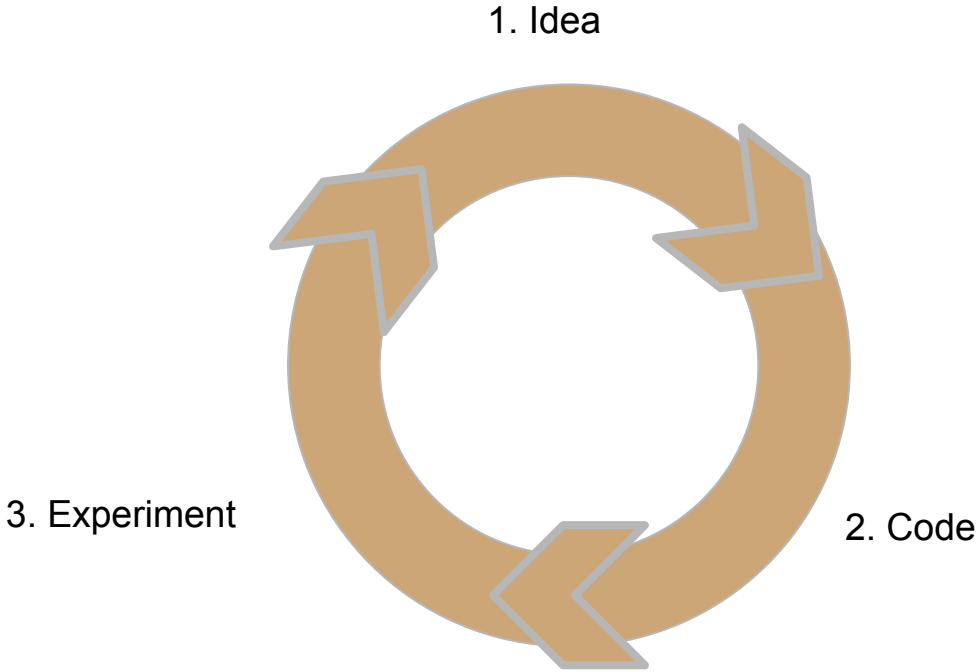
Setup: Rápido!

- 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

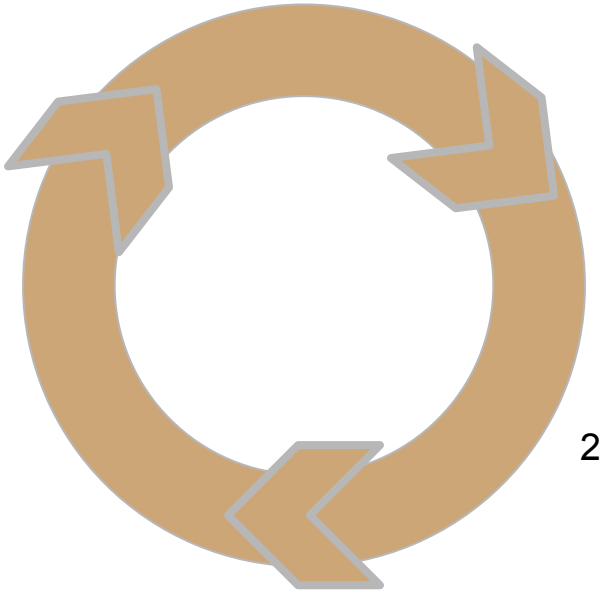
- No empezar tratando de construir el sistema perfecto.
- Construir y entrenar un sistema básico lo más rápido posible.
- Evaluarlo y estudiarlo para decidir en qué direcciones avanzar.

Primera Iteración: Modelo de Clasificación

- Se pueden probar varios modelos de clasificación (DT, MNB, LR, SVM, etc.)
- Empezar eligiendo el que mejor ande sin ninguna configuración.
- No empezar **NUNCA** con redes neuronales.
- No casarse con un único modelo.

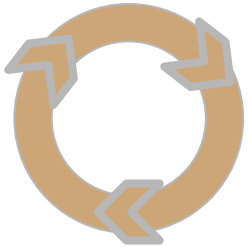
Ajuste de Hiperparámetros

1. Idea



2. Code

3. Experiment
3.1. **Hyperparameter tuning**



Ajuste de Hiperparámetros

Opciones:

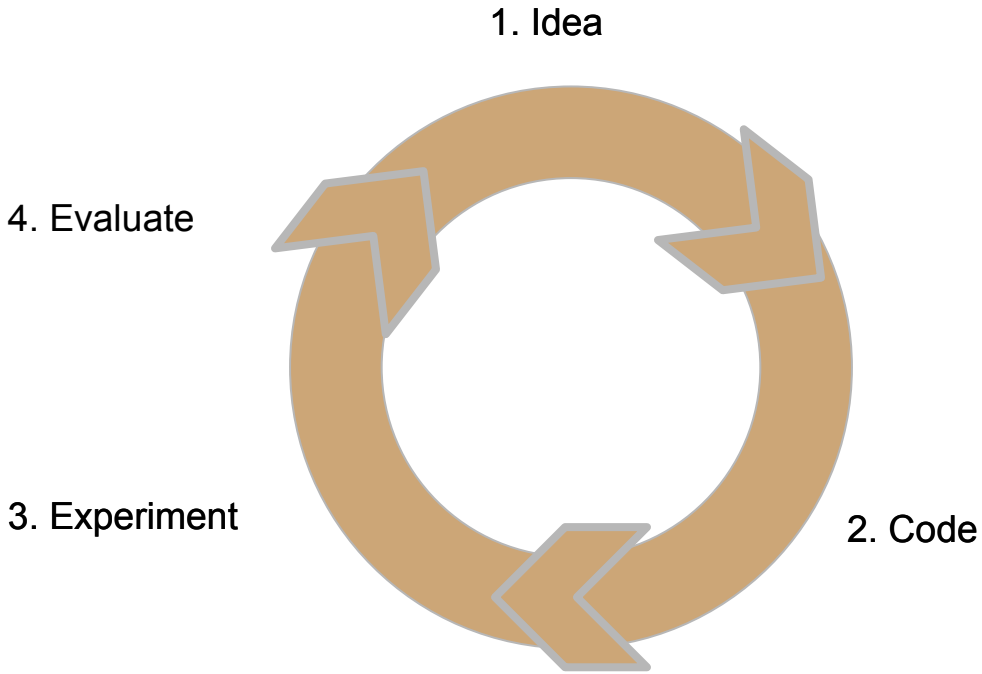
- Búsqueda manual.
- Búsqueda exhaustiva (grid-search): todas las combinaciones posibles de valores.
- Aleatoria (randomized): sampleando valores o combinaciones.
- Development vs. Cross Validation

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



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.
  - Features más expresivos: 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.**

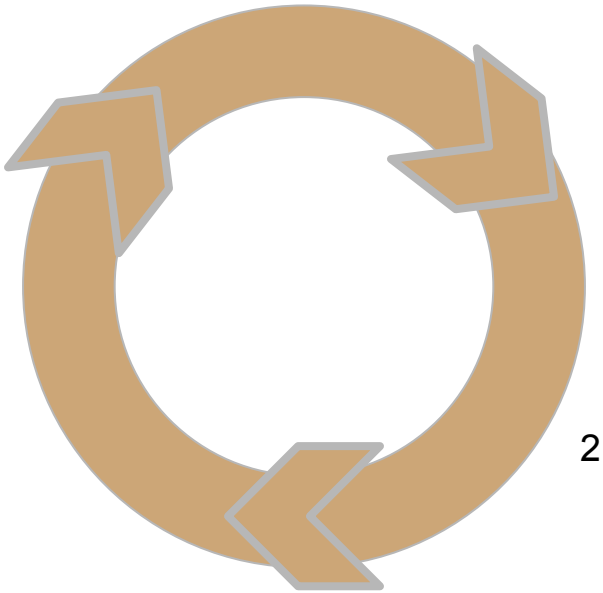
Evaluación: Reducción de Varianza

- 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

1. Idea

5. Error analysis



4. Evaluate

3. Experiment

2. Code

Análisis de Error (Error Analysis)

- ¿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 base a prob/score)

Análisis de Error (Error Analysis)

- 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%	

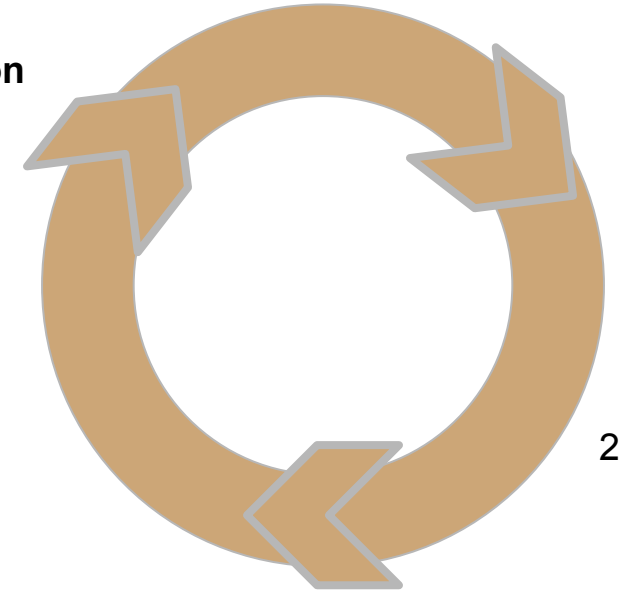
Análisis de Error (Error Analysis)

- 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

1. Idea

5.1. Error analysis
5.2. Model inspection



4. Evaluate

3. Experiment

2. Code

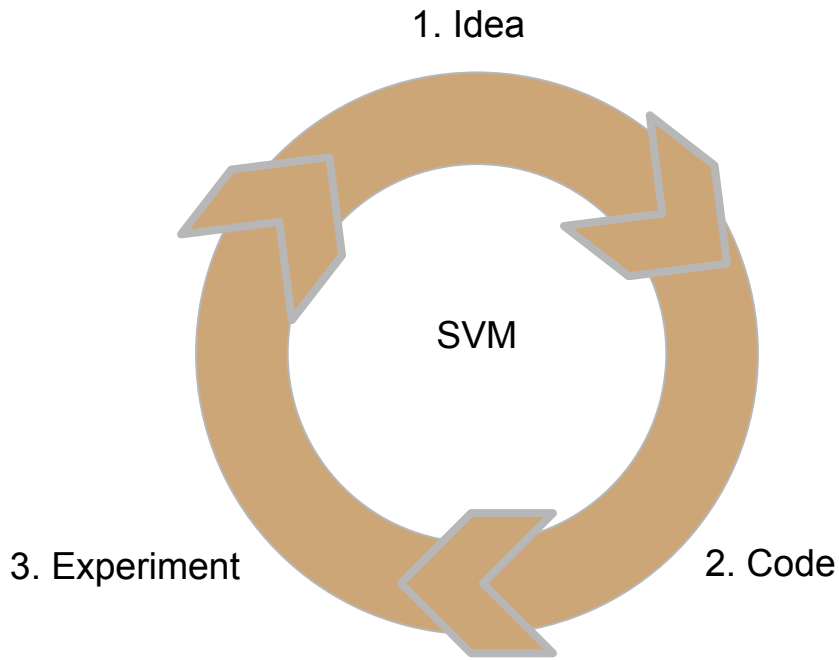
Inspección del Modelo

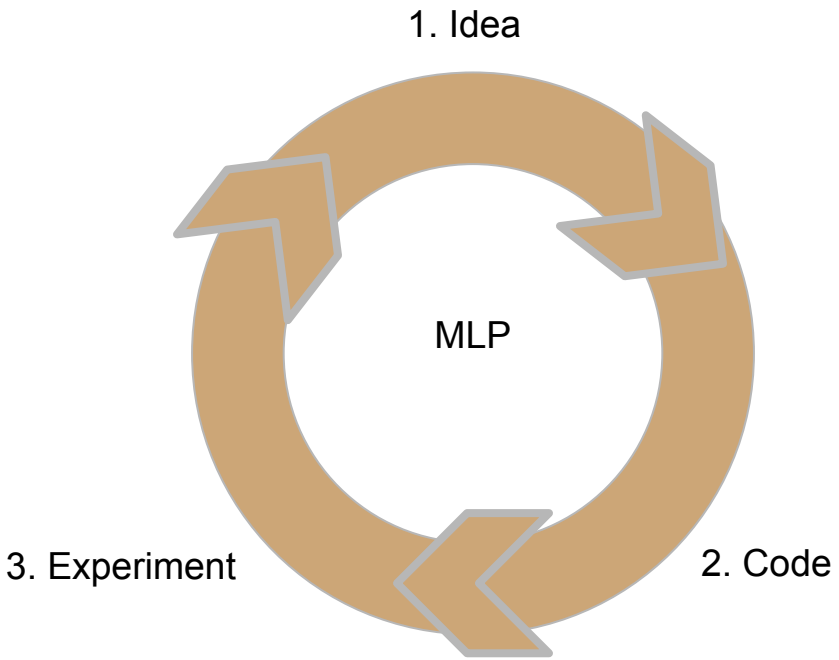
- Estudiar los parámetros del modelo una vez aprendido.
- Features más influyentes para cada clase.
- Fronteras de decisión.

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)



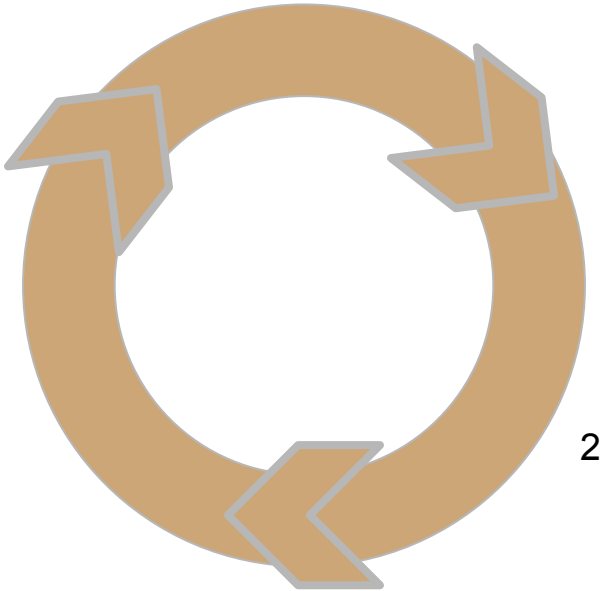


Bifurcación

- 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

1. Idea:
   - Ablation Tests
   - Retune hiperparameters
   - Reconsider Old Ideas



2. Code

3. Experiment

4. Evaluate

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

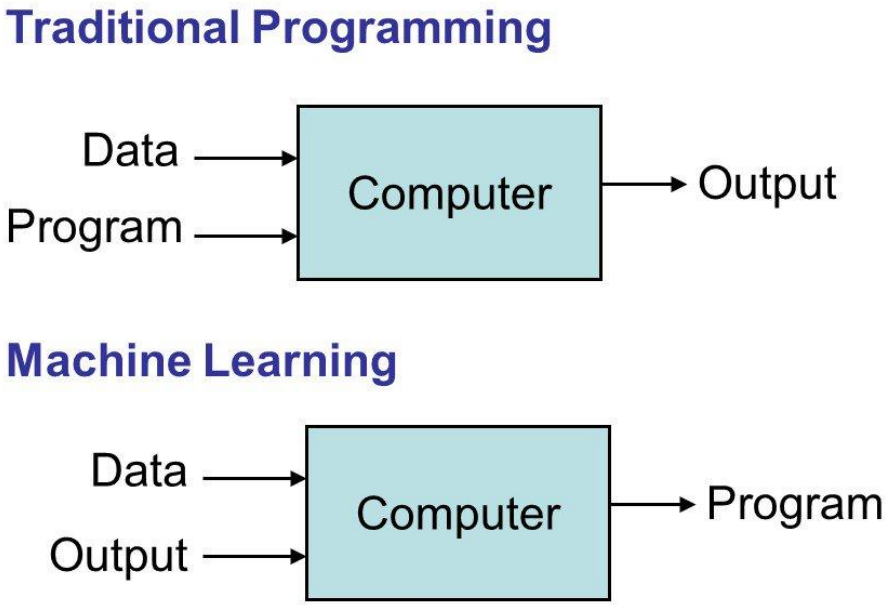
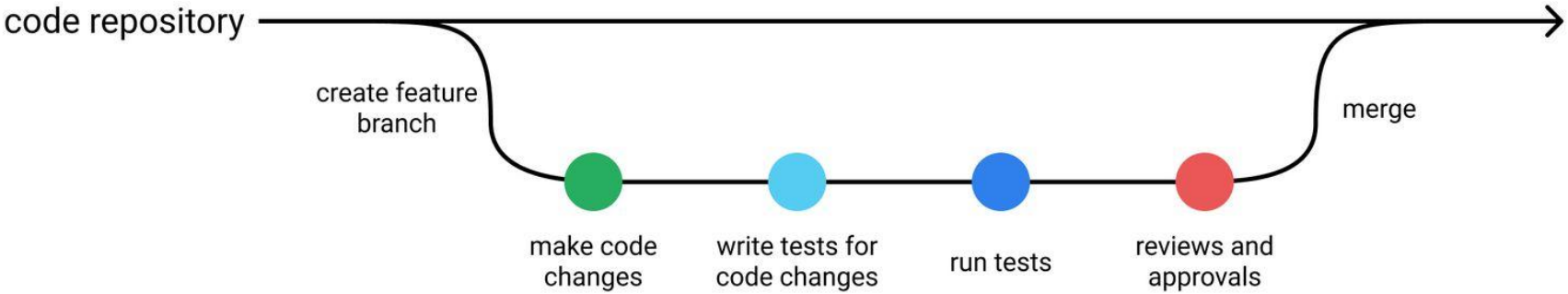


Image by Pedro Domingos

Workflow de desarrollo de software tradicional

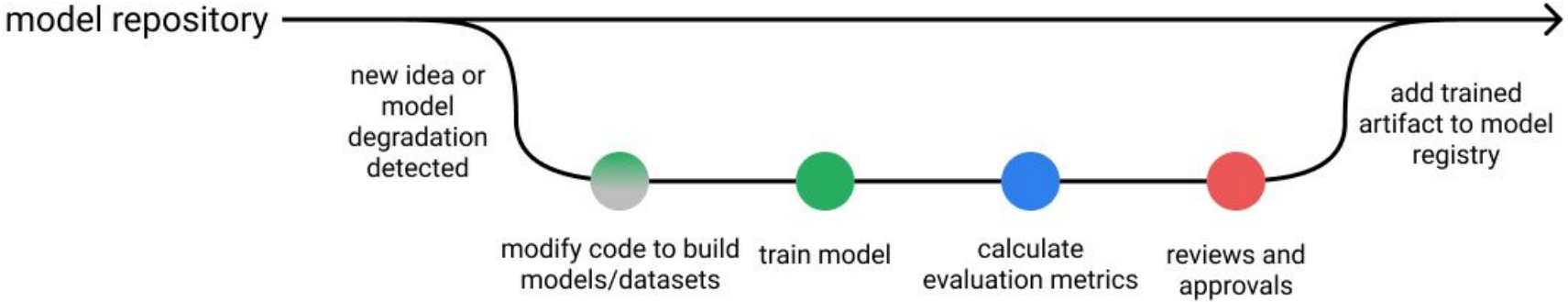


Ejecución de tests:

- unit tests
- regression tests
- integration tests
- ...

Image by Jeremy Jordan

Workflow de desarrollo de machine learning



(MLFlow)

- Metrics
- Plots
- Stats
- Hyperparams used

Image by Jeremy Jordan

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**

Model Testing: tests de pre-train

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**

Model Testing: tests de post-train

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

Model Testing: tests de post-train

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

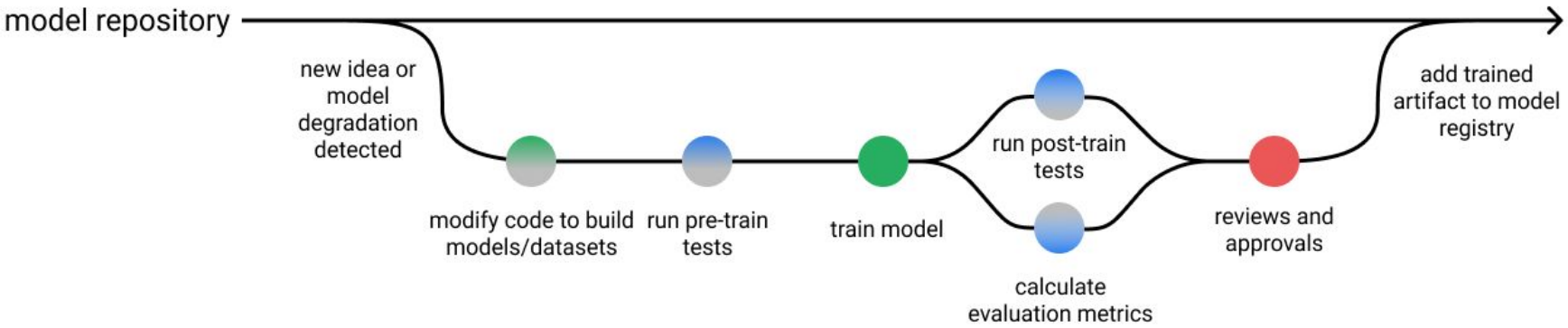


Image by Jeremy Jordan

FIN :)