Machine Learning Práctico

Pablo Zivic
@ideasrapidas

Entra a las slides

https://bit.ly/ml-practico

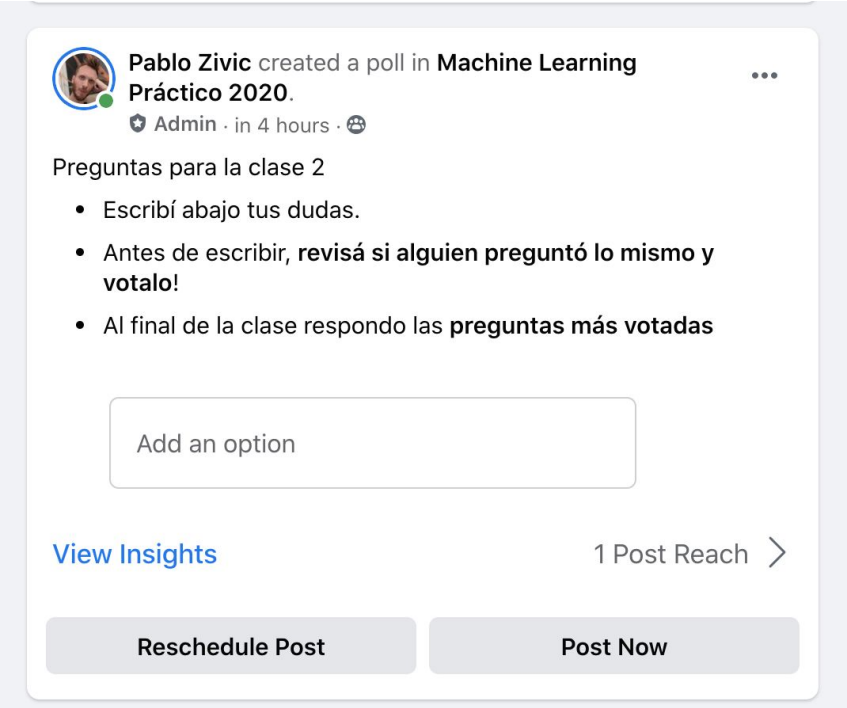
Anuncios

- Last call para grupos de estudio
  - https://forms.gle/cxmHEnF6Hq7fHk2Z7
- Para poder chatear entre todes: sumate al discord
  - https://discord.com/invite/6kUAfRvZ?fbclid=IwAR0S7QimIb_qftudua6APZjyGYjmpf3aGbqBJ-LJk4dpF-slHQDKIeh8hdg
- In class multiple choices
  - https://itempool.com/elsonidoq/live

Primera pregunta in class

https://itempool.com/elsonidoq/live/

Tenés dudas? Cargalas en el grupo de facebook

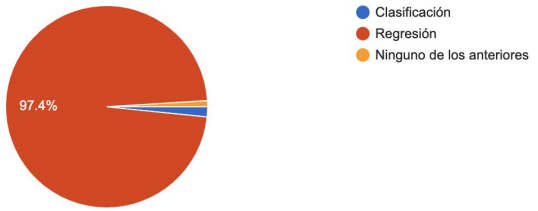


Evaluación de la clase anterior

Evaluación de la clase anterior

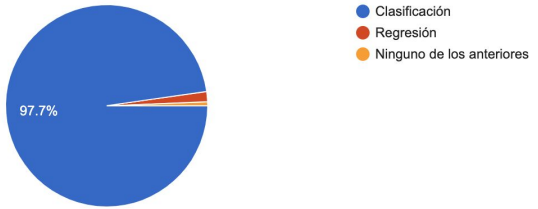
Quiero hacer un modelo que prediga cuánto tiempo falta para que llegue el colectivo en función de la frecuencia de la linea y la cantidad de gente esperando en la cola. Para eso voy a usar un modelo de

308 responses

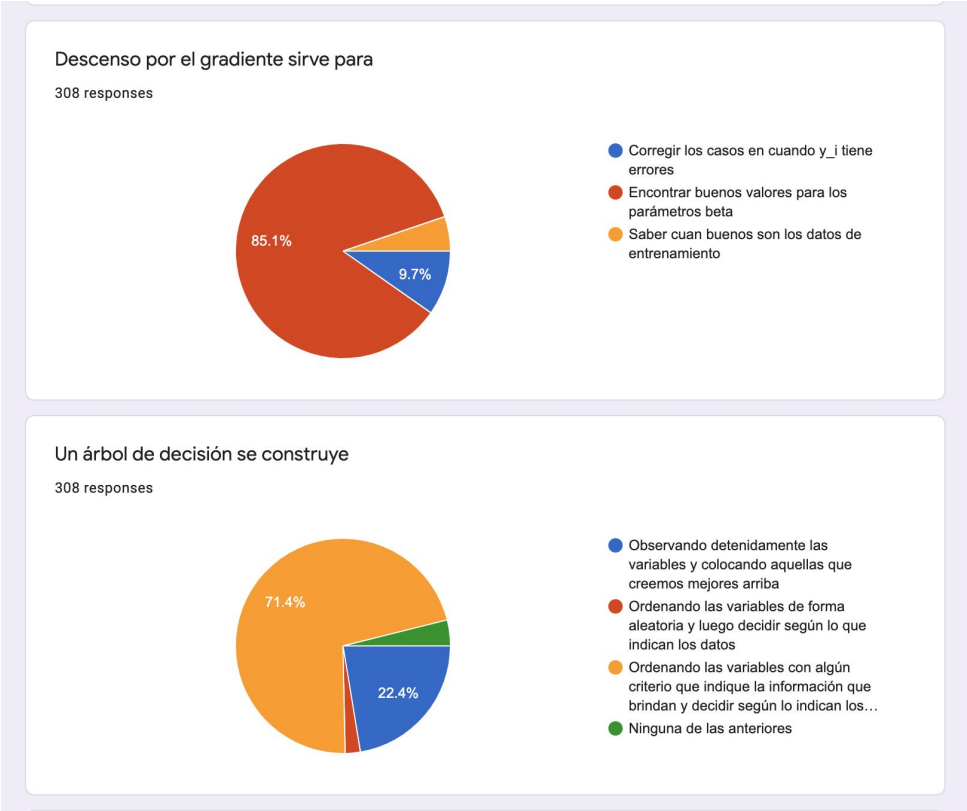


Quiero hacer un modelo que prediga si un aparato está defectuoso en función de una descripción provista por el cliente. Para eso voy a usar un modelo de

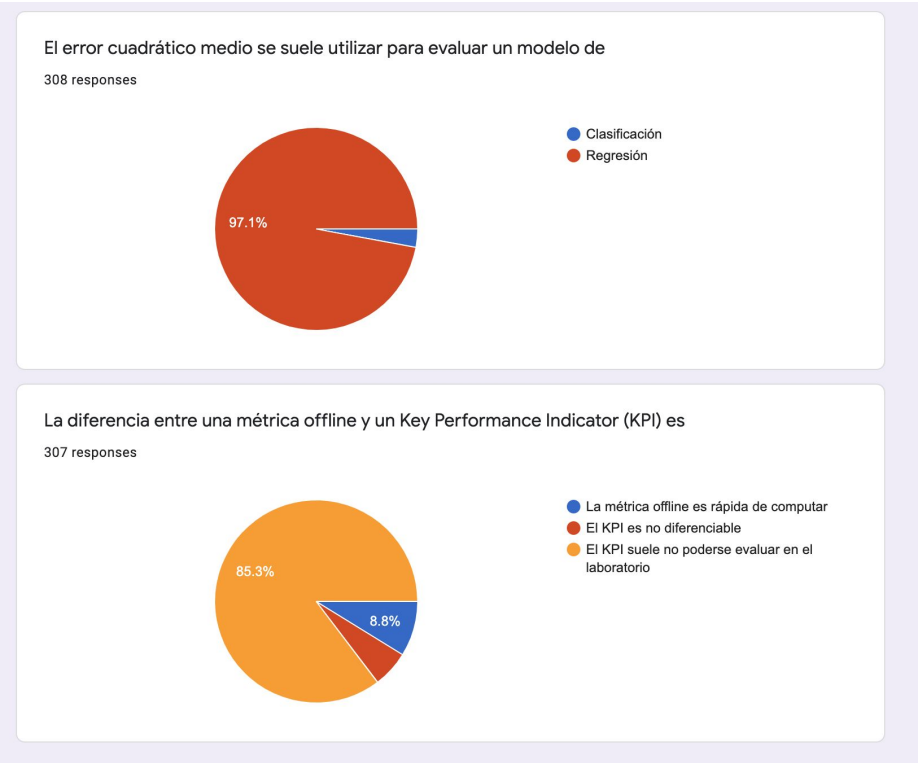
309 responses



Evaluación de la clase anterior



Evaluación de la clase anterior



Ahora si!

Arrancamos la clase 2

Empalmando con la clase anterior

- Modelos
  - Dos grandes tipos clasificación, regresión
  - Muchas familias de modelos:
    - regresión lineal / logistica, árboles, redes neuronales, etc
  - Metricas offline: precision, recall, f1, roc auc, R^2, MSE, etc

- Resolviendo problemas
  - Caso de negocio
  - KPIs
  - Metodología
  - Toma de decisiones

- Práctico
  - **E**xploratory **D**ata **A**nalysis

Situación

Terminaste de entrenar un modelo, miras la pantalla y decís

- “Uh, este modelo esta haciendo overfitting”

Te pregunto

- Cómo te diste cuenta?
- Y qué vas a hacer?

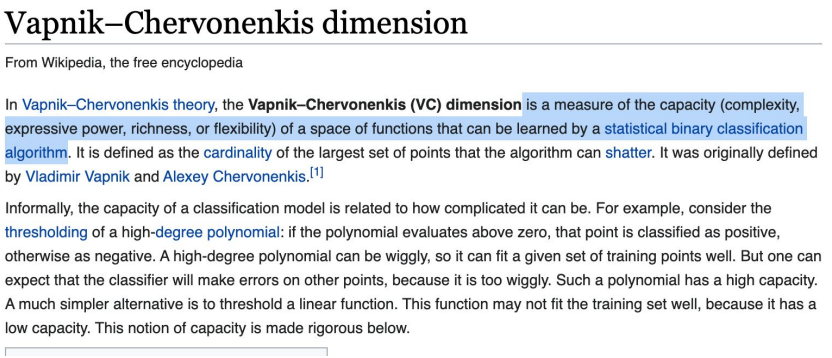
Agenda

- Vamos a ver: Herramientas para decidir cómo iterar un modelo
  - Conceptos teóricos
  - Aplicaciones prácticas
- Lectura recomendada: Capitulo de ML del libro de Ian Goodfellow de DL

¿Con qué criterio decido con qué modelo me quedo?

Model complexity

- Idea: cuanta capacidad tiene el modelo
- Formalismo: VC-Dimension



Vapnik–Chervonenkis dimension

From Wikipedia, the free encyclopedia

In Vapnik–Chervonenkis theory, the **Vapnik–Chervonenkis (VC) dimension** is a measure of the capacity (complexity, expressive power, richness, or flexibility) of a space of functions that can be learned by a statistical binary classification algorithm. It is defined as the cardinality of the largest set of points that the algorithm can shatter. It was originally defined by Vladimir Vapnik and Alexey Chervonenkis.[1]

Informally, the capacity of a classification model is related to how complicated it can be. For example, consider the thresholding of a high-degree polynomial: if the polynomial evaluates above zero, that point is classified as positive, otherwise as negative. A high-degree polynomial can be wiggly, so it can fit a given set of training points well. But one can expect that the classifier will make errors on other points, because it is too wiggly. Such a polynomial has a high capacity. A much simpler alternative is to threshold a linear function. This function may not fit the training set well, because it has a low capacity. This notion of capacity is made rigorous below.

VC-Dimension: modelo lineal

- Clasificador lineal de vectores de R^d es d
- Intuicion: la cantidad de parámetros de un modelo lineal es d

VC-Dimensión de un árbol de decisión

¿?

VC-Dimension de un arbol de decision

- No sabemos calcularlo,
  ¡¡pero no es necesario!!

- Veamos los
  hyper-parámetros
  (documentacion)

VC-Dimensión de un árbol de decisión

- No sabemos calcularlo, ¡¡pero no es necesario!!
- Veamos los hyper-parametros (documentacion)

Parameters:

criterion : *{"gini", "entropy"}, default="gini"*
The function to measure the quality of a split. Supported criteria are "gini" for the Gini impurity and "entropy" for the information gain.

splitter : *{"best", "random"}, default="best"*
The strategy used to choose the split at each node. Supported strategies are "best" to choose the best split and "random" to choose the best random split.

max_depth : *int, default=None*
The maximum depth of the tree. If None, then nodes are expanded until all leaves are pure or until all leaves contain less than min_samples_split samples.

min_samples_split : *int or float, default=2*
The minimum number of samples required to split an internal node:

- If int, then consider `min_samples_split` as the minimum number.
- If float, then `min_samples_split` is a fraction and `ceil(min_samples_split * n_samples)` are the minimum number of samples for each split.

Changed in version 0.18: Added float values for fractions.

min_samples_leaf : *int or float, default=1*
The minimum number of samples required to be at a leaf node. A split point at any depth will only be considered if it leaves at least `min_samples_leaf` training samples in each of the left and right branches. This may have the effect of smoothing the model, especially in regression.

- If int, then consider `min_samples_leaf` as the minimum number.
- If float, then `min_samples_leaf` is a fraction and `ceil(min_samples_leaf * n_samples)` are the minimum number of samples for each node.

Changed in version 0.18: Added float values for fractions.

min_weight_fraction_leaf : *float, default=0.0*
The minimum weighted fraction of the sum total of weights (of all the input samples) required to be at a leaf node. Samples have equal weight when sample_weight is not provided.

max_features : *int, float or {"auto", "sqrt", "log2"}, default=None*
The number of features to consider when looking for the best split:

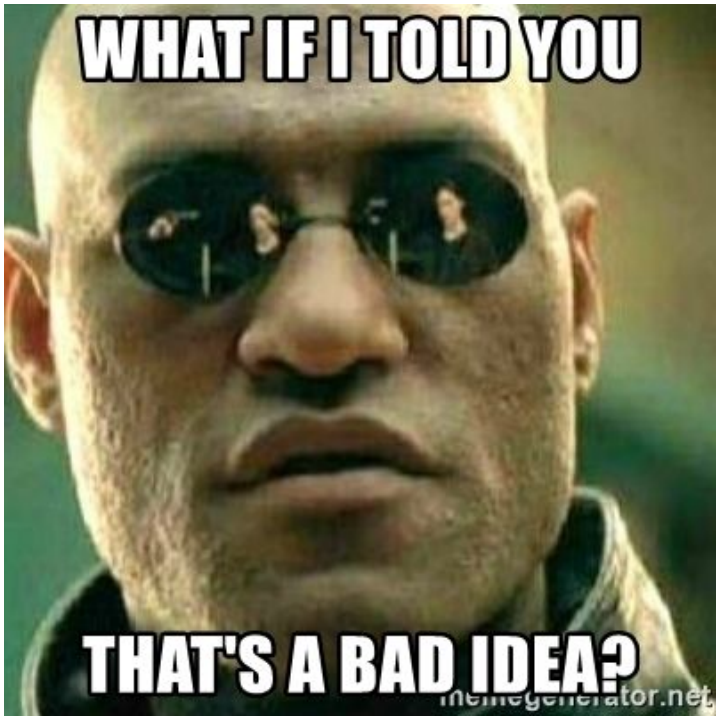
- If int, then consider `max_features` features at each split.
- If float, then `max_features` is a fraction and `int(max_features * n_features)` features are considered at each split.

Preguntas in class

https://itempool.com/elsonidoq/live/

Entonces ponemos

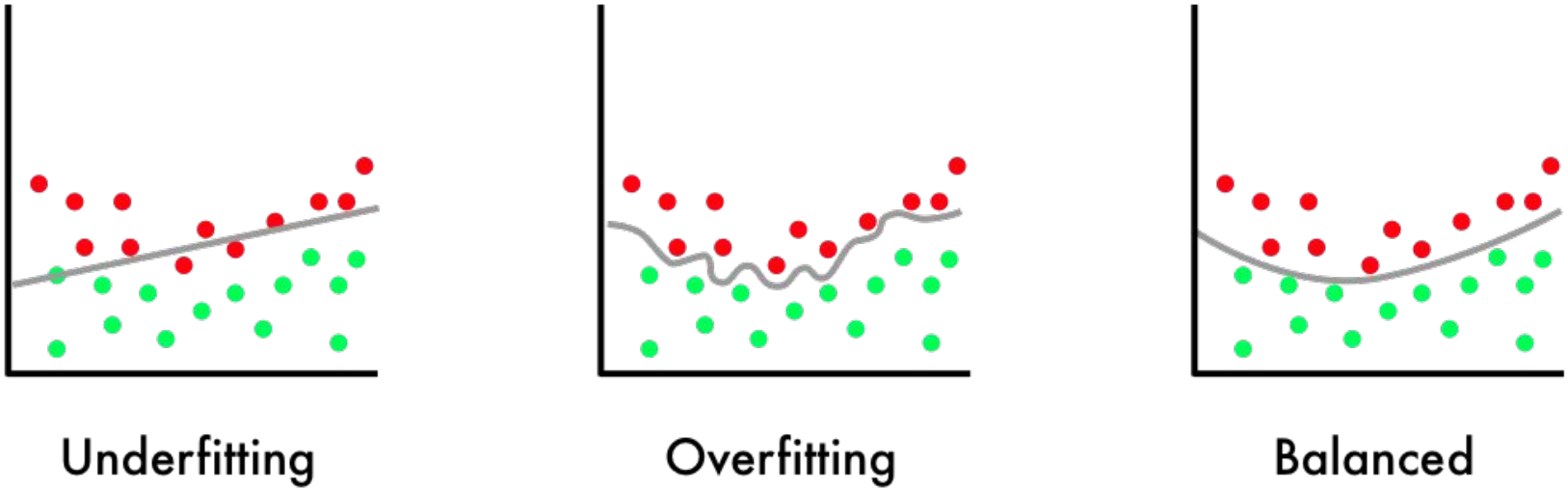
- max_depth = 100000000
- min_samples_split = 1
- max_features = todas


WHAT IF I TOLD YOU
THAT'S A BAD IDEA?

Overfitting

Also known as fitting the noise

Overfitting



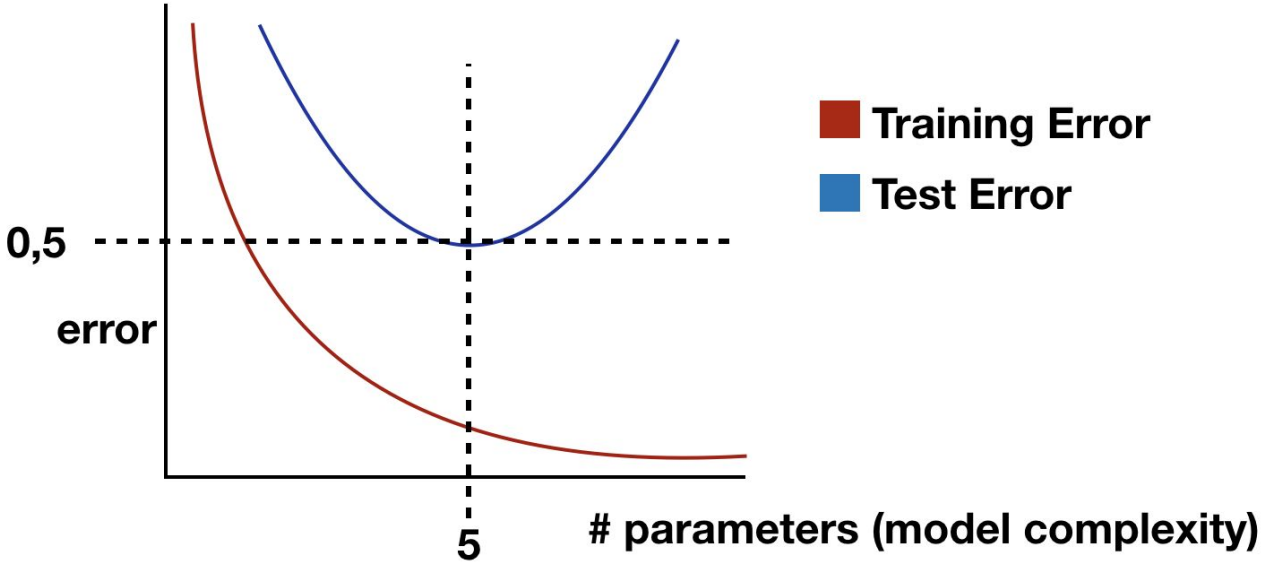
Cómo me doy cuenta que estoy overfitting?

- Train/dev/test split
  - Los 3 deben venir de la misma distribución
  - **Train** se usa para construir el modelo
  - **Dev** se usa para evaluar el modelo
  - **Test** para checkear si estoy haciendo overfitting a Dev

Cómo me doy cuenta que estoy overfitting?

- Train/dev/test split
  - Los 3 deben venir de la misma distribución
  - **Train** se usa para construir el modelo
  - **Dev** se usa para evaluar el modelo
  - **Test** para checkear si estoy haciendo overfitting a Dev

- Si el modelo hizo overfitting
  - Quiere decir que ajustó al ruido ⇒ Debería degradarse en un sample nuevo de datos
  - Eso quiere decir que **va a funcionar mucho peor en dev/test que en train**


Training Error
Test Error
0,5
error
5
parameters (model complexity)

Y cómo controlo el model complexity?

- Árbol:
  - Profundidad
  - Número de datos en las hojas
  - Cantidad de bins para variables continuas

- Regresión logistica / lineal
  - Cantidad de features
  - Hay mas, volvemos a esto en un ratito ^J^

Preguntas in class

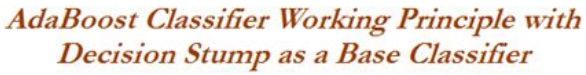
https://itempool.com/elsonidoq/live/

Y cómo controlo el model complexity?

- Ensembles de arboles
  - xgboost, lightgbm, catboost
  - gradient boosting

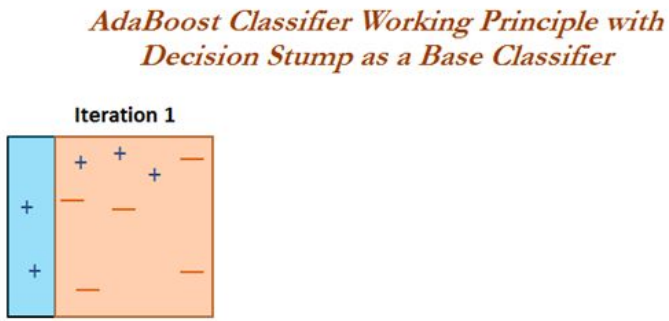
Y cómo controlo el model complexity?

- Ensembles de arboles
  - xgboost, lightgbm, catboost
  - gradient boosting



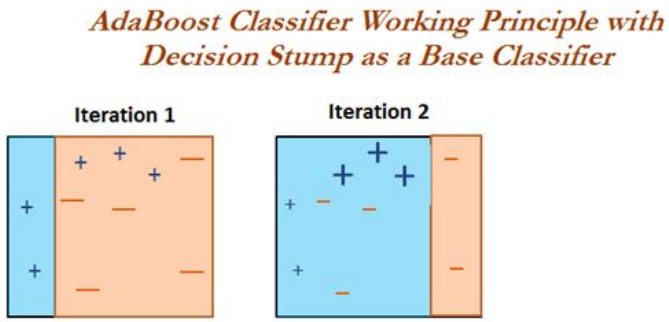
Y cómo controlo el model complexity?

- Ensembles de arboles
  - xgboost, lightgbm, catboost
  - gradient boosting



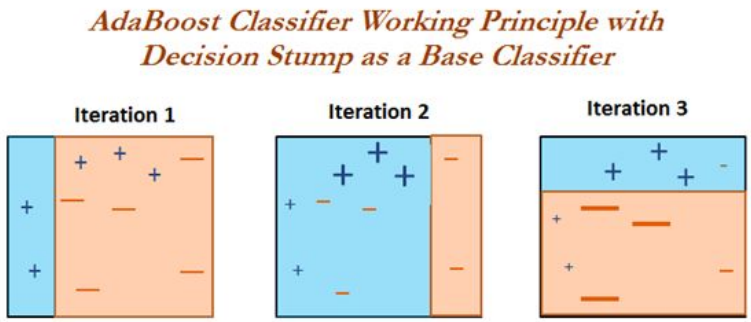
Y cómo controlo el model complexity?

- Ensembles de arboles
  - xgboost, lightgbm, catboost
  - gradient boosting



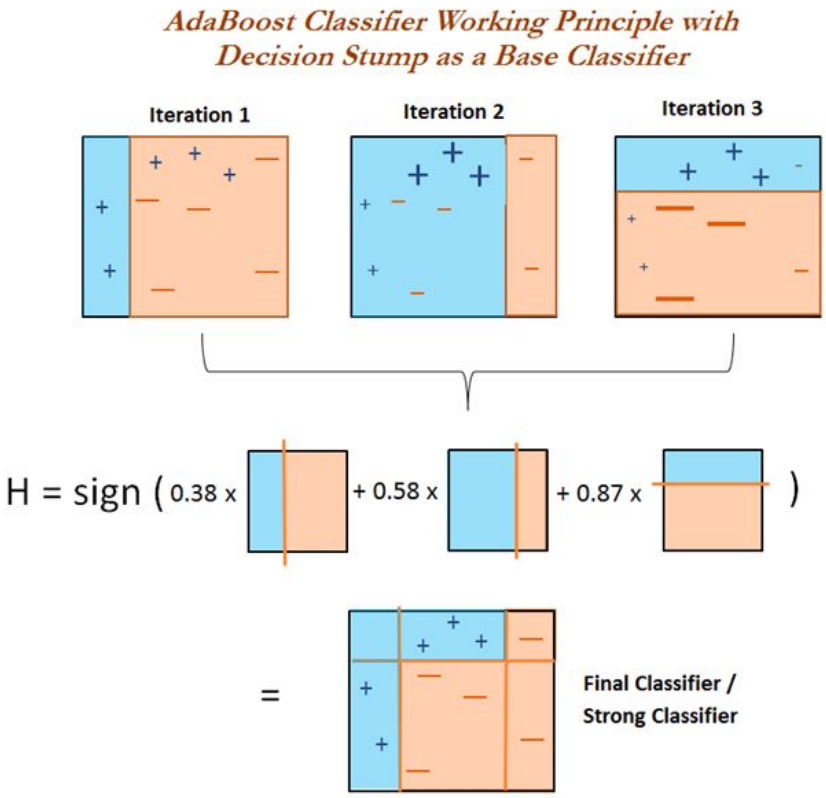
Y cómo controlo el model complexity?

- Ensembles de arboles
  - xgboost, lightgbm, catboost
  - gradient boosting



Y cómo controlo el model complexity?

- Ensembles de arboles
  - xgboost, lightgbm, catboost
  - gradient boosting



Y cómo controlo el model complexity?

- Ensembles de arboles
  - xgboost, lightgbm, catboost
  - gradient boosting

- Docu de [lightgbm]

For Better Accuracy

- Use large `max_bin` (may be slower)
- Use small `learning_rate` with large `num_iterations`
- Use large `num_leaves` (may cause over-fitting)
- Use bigger training data
- Try `dart`

Y cómo controlo el model complexity?

- Ensembles de arboles
  - xgboost, lightgbm, catboost
  - gradient boosting

- Docu de [lightgbm]

For Better Accuracy

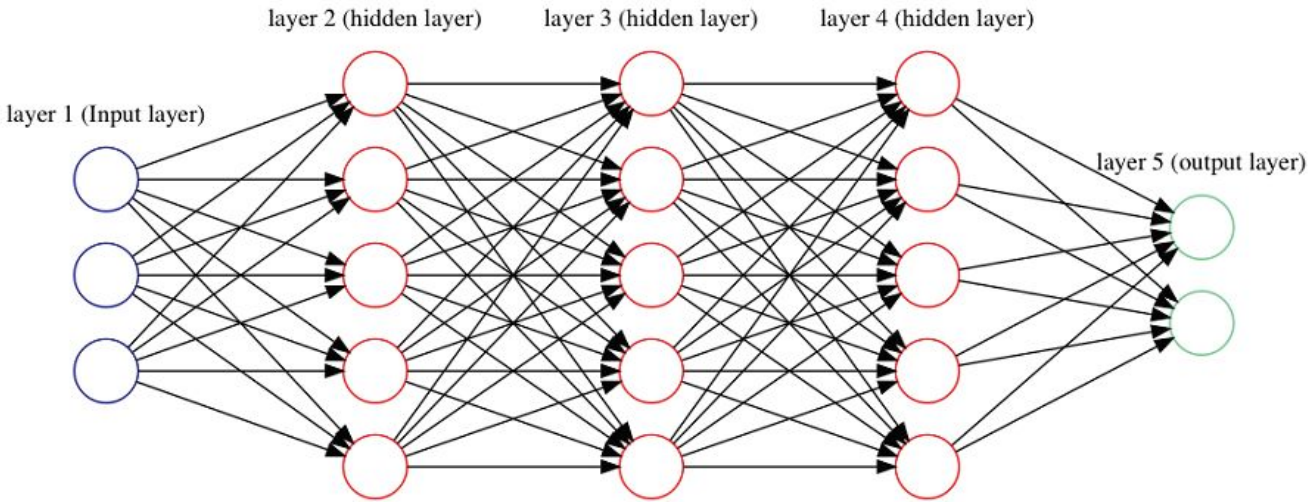
- Use large `max_bin` (may be slower)
- Use small `learning_rate` with large `num_iterations`
- Use large `num_leaves` (may cause over-fitting)
- Use bigger training data
- Try `dart`

Deal with Over-fitting

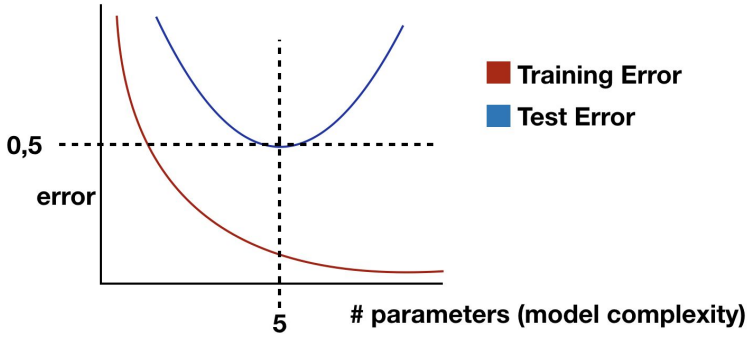
- Use small `max_bin`
- Use small `num_leaves`
- Use `min_data_in_leaf` and `min_sum_hessian_in_leaf`
- Use bagging by set `bagging_fraction` and `bagging_freq`
- Use feature sub-sampling by set `feature_fraction`
- Use bigger training data
- Try `lambda_l1`, `lambda_l2` and `min_gain_to_split` for regularization
- Try `max_depth` to avoid growing deep tree
- Try `extra_trees`
- Try increasing `path_smooth`

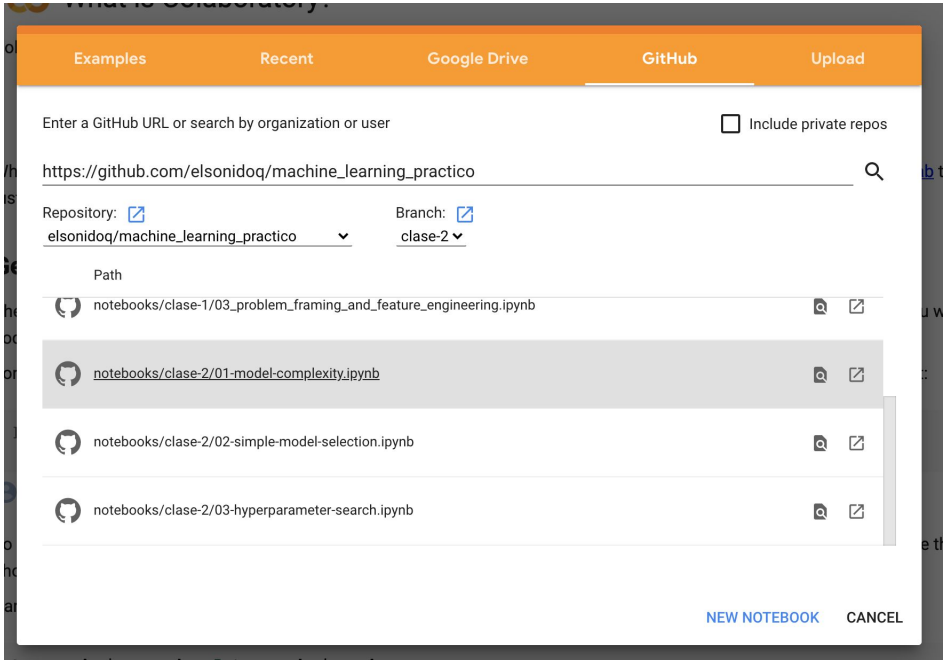
Y cómo controlo el model complexity?

Deep Learning!



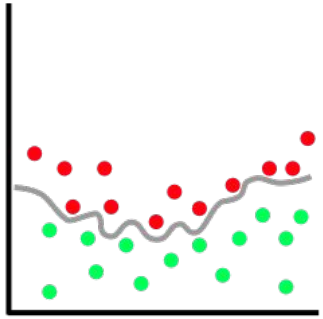
Model complexity: hands on!





Apartado: cantidad de datos

Agregar datos es una forma de reducir el overfitting



Overfitting

Y cómo controlo el model complexity?

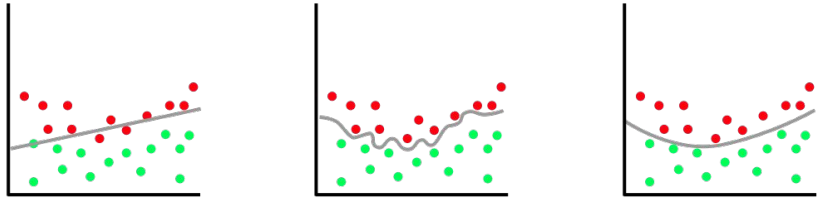
- Penalizando soluciones poco probables que puedan estar ajustando el ruido
- Por ejemplo, en un modelo lineal, **penalizar coeficientes grandes**
- Esto se conoce como **regularización**

Penalizar coeficientes grandes: intuición

- Modelo: $y_i = f_\beta(x_i)$
- Hipótesis
  - si x_1 está “cerca” de x_2
  - entonces y_1 debería estar cerca de y_2
  - ergo, quisieramos que $f_\beta(x_1)$ este cerca de $f_\beta(x_2)$
- Tener coeficientes “grandes” atenta contra este comportamiento

Penalizar coeficientes grandes: intuición

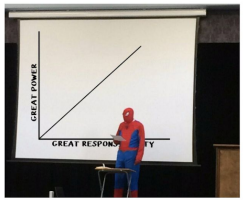
- Modelo: $y_i = f_\beta(x_i)$
- Hipótesis
  - si x_1 está “cerca” de x_2
  - entonces y_1 debería estar cerca de y_2
  - ergo, quisieramos que $f_\beta(x_1)$ este cerca de $f_\beta(x_2)$
- Tener coeficientes “grandes” atenta contra este comportamiento



Penalizar coeficientes grandes: intuición

- Modelo: $y_i = f_\beta(x_i)$
- Hipótesis
  - si x_1 está “cerca” de x_2
  - entonces y_1 debería estar cerca de y_2
  - ergo, quisieramos que $f_\beta(x_1)$ este cerca
- Tener coeficientes “grandes” atenta

Una **gran complejidad** conlleva una **gran regularización**



Ejemplos de regularizaciones: L1 y L2

L1 Regularization

$$\text{Cost} = \sum_{i=0}^{N} (y_i - \sum_{j=0}^{M} x_{ij} W_j)^2 + \lambda \sum_{j=0}^{M} |W_j|$$

L2 Regularization

$$\text{Cost} = \underbrace{\sum_{i=0}^{N} (y_i - \sum_{j=0}^{M} x_{ij} W_j)^2}_{\text{Loss function}} + \underbrace{\lambda \sum_{j=0}^{M} W_j^2}_{\text{Regularization Term}}$$

Multiplicadores de Lagrange

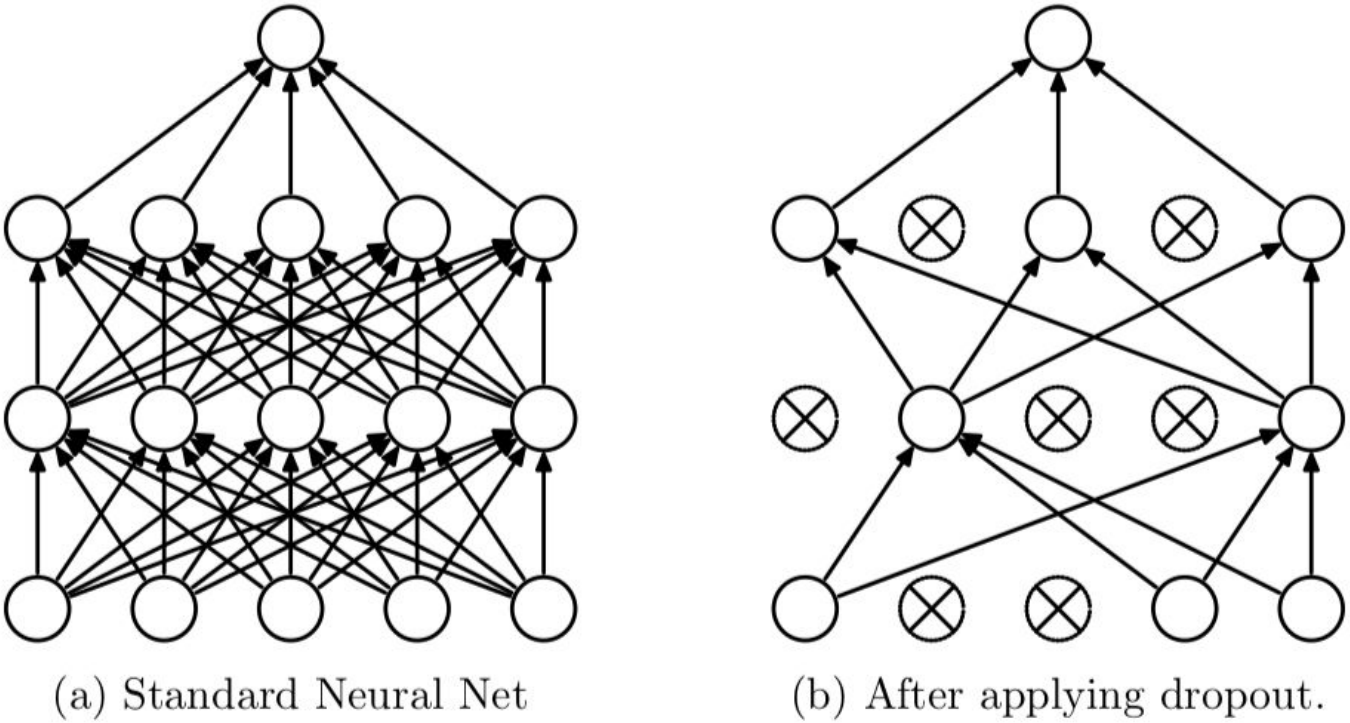
$$\text{maximize } f(x, y)$$
$$\text{subject to: } g(x, y) \leq 0$$

$$\mathcal{L}(x, y, \lambda) = f(x, y) - \lambda g(x, y),$$

Regularization Term	Method
L2 norm: $\|\boldsymbol{\beta}\|_2$	ridge regression
L1 norm: $\|\boldsymbol{\beta}\|_1$	LASSO
Lq norm: $\|\boldsymbol{\beta}\|_q$	bridge regression
weighted L1 norm: $\sum_{j=1}^{p} w_j \lvert\beta_j\rvert$	adaptive LASSO
$c_1\|\boldsymbol{\beta}\|_1 + c_2\|\boldsymbol{\beta}\|_2^2$	elastic net
$\sum_{g=1}^{G} \sqrt{p_g}\|\beta_g\|_2$	group LASSO

Hastie pagina 61
Bishop pagina 144

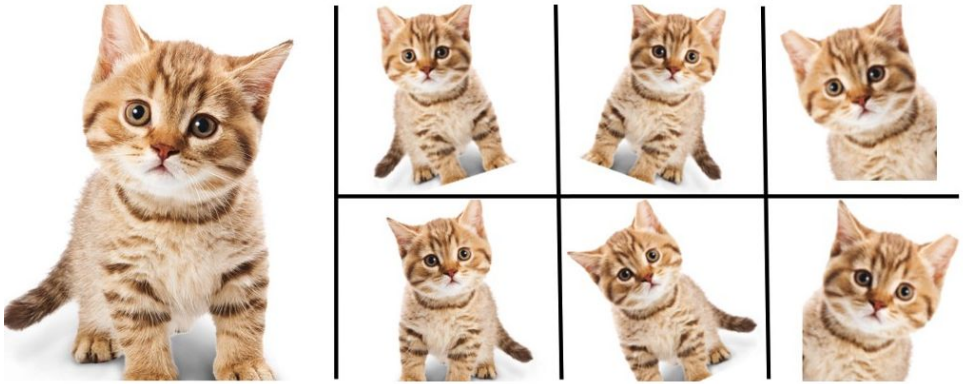
Ejemplos de regularizaciones: Dropout



(a) Standard Neural Net

(b) After applying dropout.

Data augmentation



Preguntas in class

https://itempool.com/elsonidoq/live/

Bias / Variance trade off

Bias / variance trade off

Una versión resumida de este post.
Muy muy recomendados!!!

Bias / variance trade off [estimadores]

drawn. Formally, *estimator* is a function on a sample S:

$$\hat{\theta}_S = g(S), S = (x(1), \ldots, x(m)),$$

where $x(i)$ is a random variable drawn from a distribution D, i.e. $x(i) \sim D$.

Bias / variance trade off [estimadores]

Estimator properties

Now we would like to know how good our *estimators* are. There are two properties we can consider: ***Estimator Bias*** and ***Estimator Variance***.

Estimator Bias measures how good our *estimator* is in *estimating* the real value. It is a simple difference:

$$Bias(\hat{\theta}_S, \theta) = \mathbb{E}_{S \sim D^m}[\hat{\theta}_S] - \theta$$

Estimator Variance measures how "jumpy" our *estimator* is to sampling, e.g. if we observe the stock price every 100ms instead of every 10ms would the estimator change a lot?

$$Var(\hat{\theta}_S) = Var_{S \sim D^m}[\hat{\theta}_S]$$

Bias / variance trade off [estimadores]

$$MSE = \mathbb{E}\left[(\hat{\theta}_S - \theta)^2\right] = Bias^2(\hat{\theta}_S, \theta) + Var(\hat{\theta}_S)$$

where the expectations are taken with respect to S random variable.

Bias / variance trade off [Machine Learning]

- Asumimos que existe una función *desconocida* que vincula **x** con **y**

$$y = f(x) + \epsilon$$

- Entrenamos un modelo $\hat{f}$ y evaluamos cómo le va en datos no vistos

$$MSE = \mathbb{E}_{(x,y)\sim D, S\sim D^m, \epsilon\sim E}\left[(y - \hat{f}_S(x))^2\right]$$

Bias / variance trade off [Machine Learning]

- Se puede demostrar (y esta en el medium) que

$$MSE = \mathbb{E}_{(x,y)\sim D, S\sim D^m, \epsilon\sim E}\left[(y - \hat{f}_S(x))^2\right]$$

$$\begin{aligned} MSE =& Var(f(x) - \hat{f}_S(x)) \\ &+ Var(\epsilon) \\ &+ \left(\mathbb{E}[f(x)] - \mathbb{E}[\hat{f}_S(x)]\right)^2 \end{aligned}$$

Bias / variance trade off [Machine Learning]

- Se puede demostrar (esta en el post de medium) que

$$MSE = \mathbb{E}_{(x,y)\sim D, S\sim D^m, \epsilon\sim E}\left[(y - \hat{f}_S(x))^2\right]$$

$$\begin{aligned} MSE =& Var(f(x) - \hat{f}_S(x)) \\ &+ Var(\epsilon) \\ &+ \left(\mathbb{E}[f(x)] - \mathbb{E}[\hat{f}_S(x)]\right)^2 \end{aligned}$$

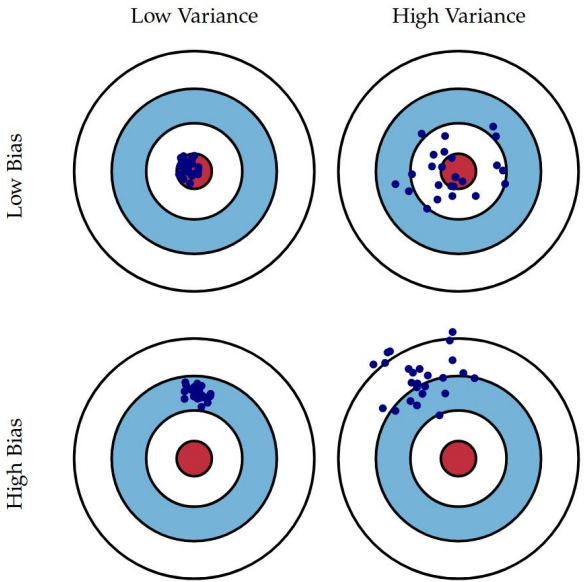


Fig. 1 Graphical illustration of bias and variance.

De qué me sirve?

Entreno un modelo.

- Si tengo high bias ⇒ Increase model complexity
- Si tengo high variance ⇒ menos model complexity o regularize o más data

Y cómo me doy cuenta

- High Bias
  - Error alto en training y validation sets
- High variance
  - Error bajo en training set y alto en validation set
- Asumiendo que no overfiteamos al validation set

Preguntas in class

https://itempool.com/elsonidoq/live/

Model selection

El arte de navegar por el universo de soluciones

Parámetros vs hyper parámetros

- Un parámetro es algo que se estima en el método de aprendizaje
  - e.g. es el β en una regresión
- Un hyper parámetro determina el método de aprendizaje
  - e.g. si vamos a usar o no regularización

Model selection es
buscar los mejores hyper parámetros para nuestro problema

Hyper parámetros

- Todo lo que no se estima con los datos puede pensarse como hyper parámetro
- Normalizamos los datos?
- Vamos a utilizar X característica?
- Usamos arbol o red neuronal?
- Qué altura máxima le permito tener al árbol?

Model selection: Diferentes approaches

- Pruebas manuales
- Grid search
- Random search
- Secuential model based optimization
- Auto ML

Pruebas manuales

- Sistematicidad es fundamental
- Permite un análisis mucho más detallado entre prueba y prueba
- Muy útil con modelos que son caros de entrenar

Grid search


YOU COMPLETED A GRID SEARCH?
PLEASE TELL ME MORE ABOUT YOUR LONG, SAD LIFE
memegenerator.net

Grid search

Pros:

- Exhaustivo
- Facil de programar

Cons

Grid search

Pros:

- Exhaustivo
- Facil de programar

Cons

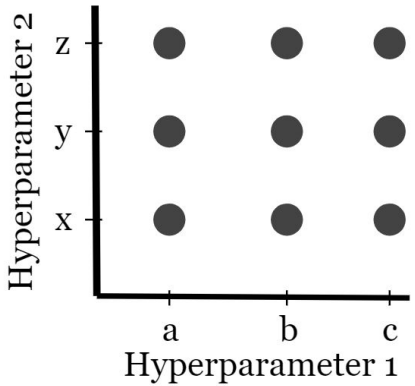
- Exhaustivo 

Random search

Es como Grid Search pero no se queda clavado probando todas las posibilidades

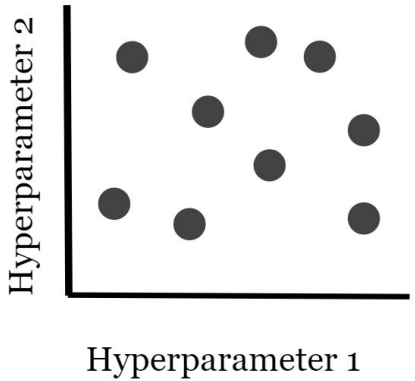
Grid Search

```
Pseudocode
Hyperparameter_One = [a, b, c]
Hyperparameter_Two = [x, y, z]
```



Random Search

```
Pseudocode
Hyperparameter_One = random.num(range)
Hyperparameter_Two = random.num(range)
```



Random search

Pros:

- Explora el espacio sin tener que detenerse en cada posible combinación
- Facil de programar

Cons:

- No hay relación entre los experimentos viejos y los nuevos
  - i.e. no aprende del pasado

Secuential **M**odel **B**ased **O**ptimization

- Entrenar un modelo es una operación cara
- Model selection puede pensarse como un problema de optimización
- Queremos optimizar *f(hyper_parameters) → metrica*
- Por ejemplo
  - Familia de modelos: regresion lineal
  - hyper_parameters: coeficiente λ de regularización
  - f(λ)
    - Entrena el modelo utilizando λ
    - Devuelve el error cuadrático en el development set

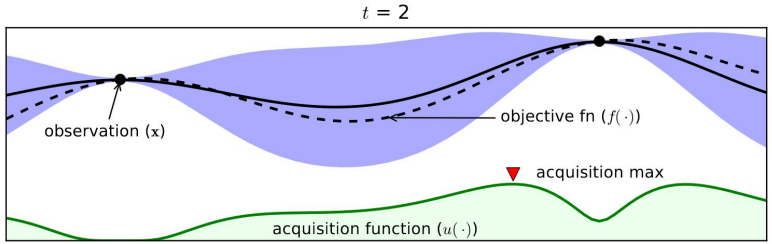
Secuential **M**odel **B**ased **O**ptimization

- La idea es utilizar un modelo para guiar la búsqueda en el espacio de hyper parámetros

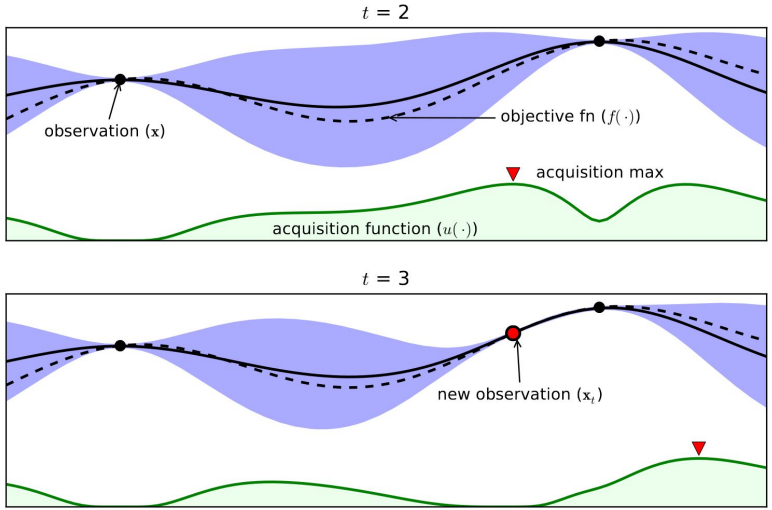
- Llevándolo a concreto en nuestro ejemplo
  - $x_i \rightarrow \lambda$
  - $y_i \rightarrow$ error cuadratico en development set

- Hay muuuuchas implementaciones

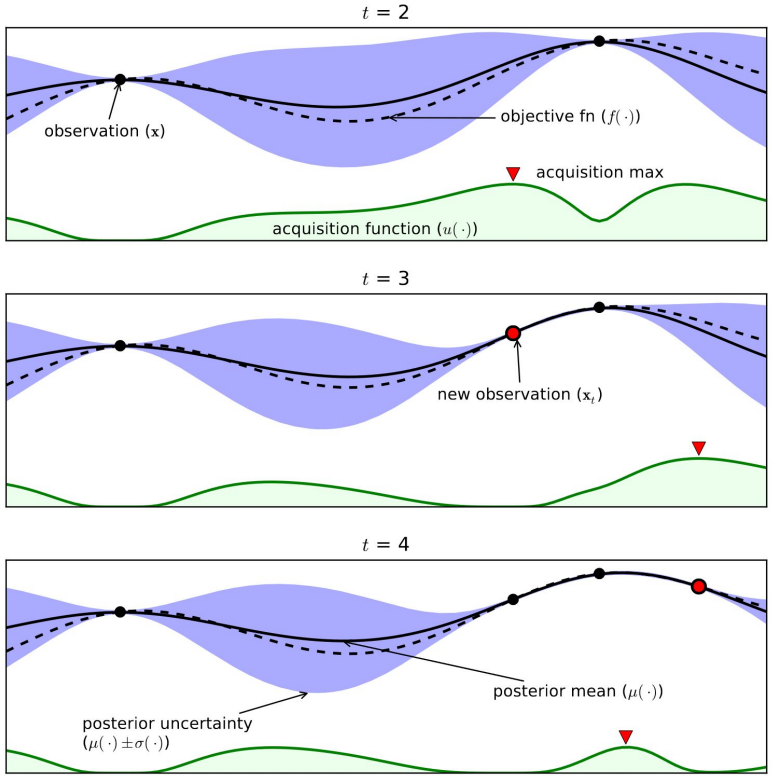
Ejemplo: SMBO bayesiano



Ejemplo: SMBO bayesiano



Ejemplo: SMBO bayesiano



Exemplo SMBO: hyperopt

```
from sklearn import datasets
iris = datasets.load_iris()
X = iris.data
y = iris.target

def hyperopt_train_test(params):
    clf = KNeighborsClassifier(**params)
    return cross_val_score(clf, X, y).mean()
```

Exemplo SMBO: hyperopt

```
from sklearn import datasets
iris = datasets.load_iris()
X = iris.data
y = iris.target

def hyperopt_train_test(params):
    clf = KNeighborsClassifier(**params)
    return cross_val_score(clf, X, y).mean()

space4knn = {
    'n_neighbors': hp.choice('n_neighbors', range(1,100))
}
```

Exemplo SMBO: hyperopt

```
from sklearn import datasets
iris = datasets.load_iris()
X = iris.data
y = iris.target

def hyperopt_train_test(params):
    clf = KNeighborsClassifier(**params)
    return cross_val_score(clf, X, y).mean()

space4knn = {
    'n_neighbors': hp.choice('n_neighbors', range(1,100))
}

def f(params):
    acc = hyperopt_train_test(params)
    return {'loss': -acc, 'status': STATUS_OK}

trials = Trials()
best = fmin(f, space4knn, algo=tpe.suggest, max_evals=100, trials=trials)
```

Auto ML

- Esencialmente SMBO
- Posiblemente con Deep Learning
- Abstrae los espacios de búsqueda
  - ⇒ No tenes que saber cómo funciona cada hyper parametro

Ejemplo: Auto sklearn

Example

```
>>> import autosklearn.classification
>>> import sklearn.model_selection
>>> import sklearn.datasets
>>> import sklearn.metrics
>>> X, y = sklearn.datasets.load_digits(return_X_y=True)
>>> X_train, X_test, y_train, y_test = \
        sklearn.model_selection.train_test_split(X, y, random_state=1)
>>> automl = autosklearn.classification.AutoSklearnClassifier()
>>> automl.fit(X_train, y_train)
>>> y_hat = automl.predict(X_test)
>>> print("Accuracy score", sklearn.metrics.accuracy_score(y_test, y_hat))
```

No free launch: todas las metodologías son útiles

- Nada es gratis
  - e.g. AutoML nos da algo a cambio de computo y tiempo

- Pensar un problema genera saltos cualitativos de performance
  - Framing del problema, arquitectura específica del modelo / red neuronal

Antes de terminar!

Situación

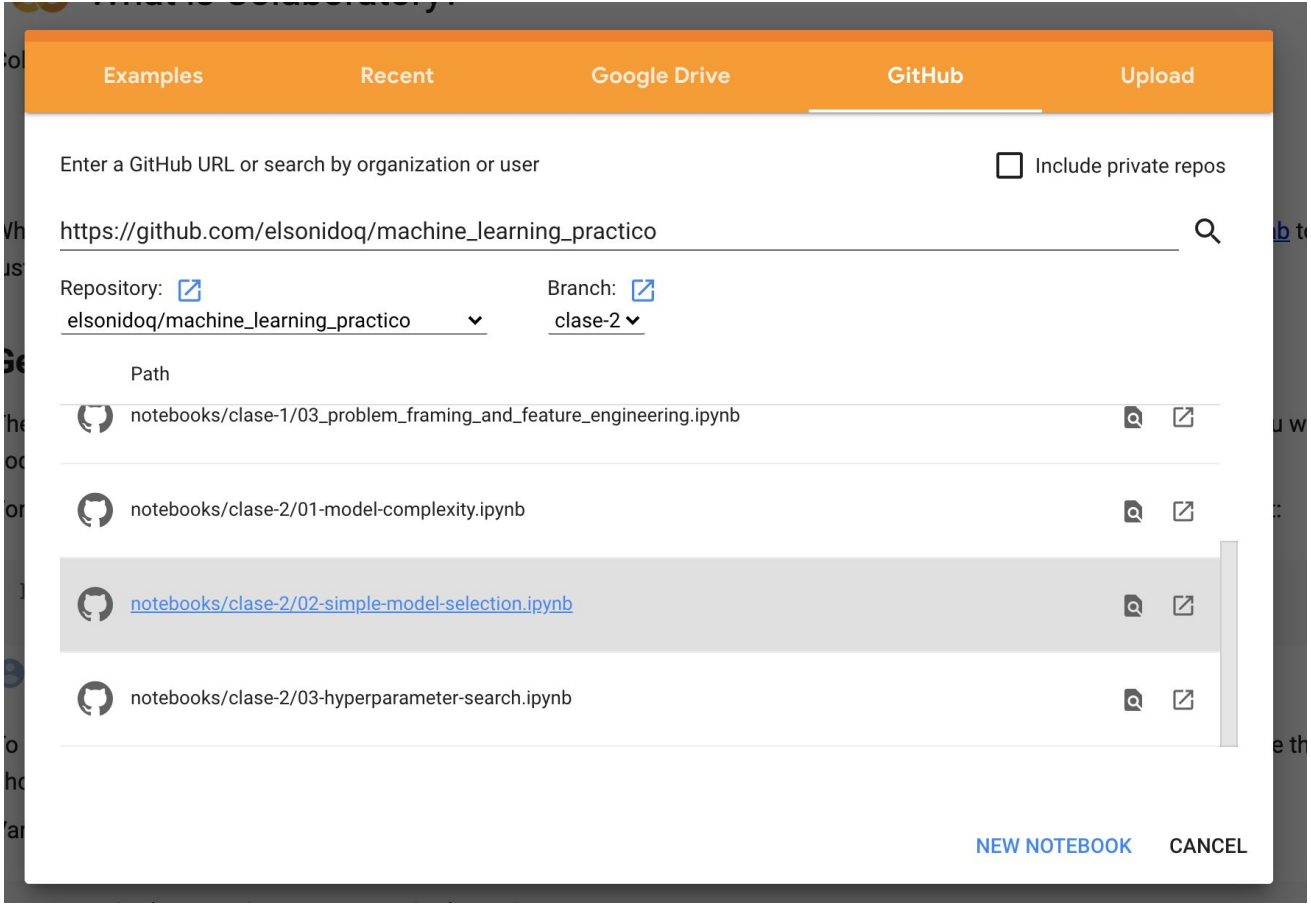
Terminaste de entrenar un modelo, miras la pantalla y decís

- “Uh, este modelo esta haciendo overfitting”

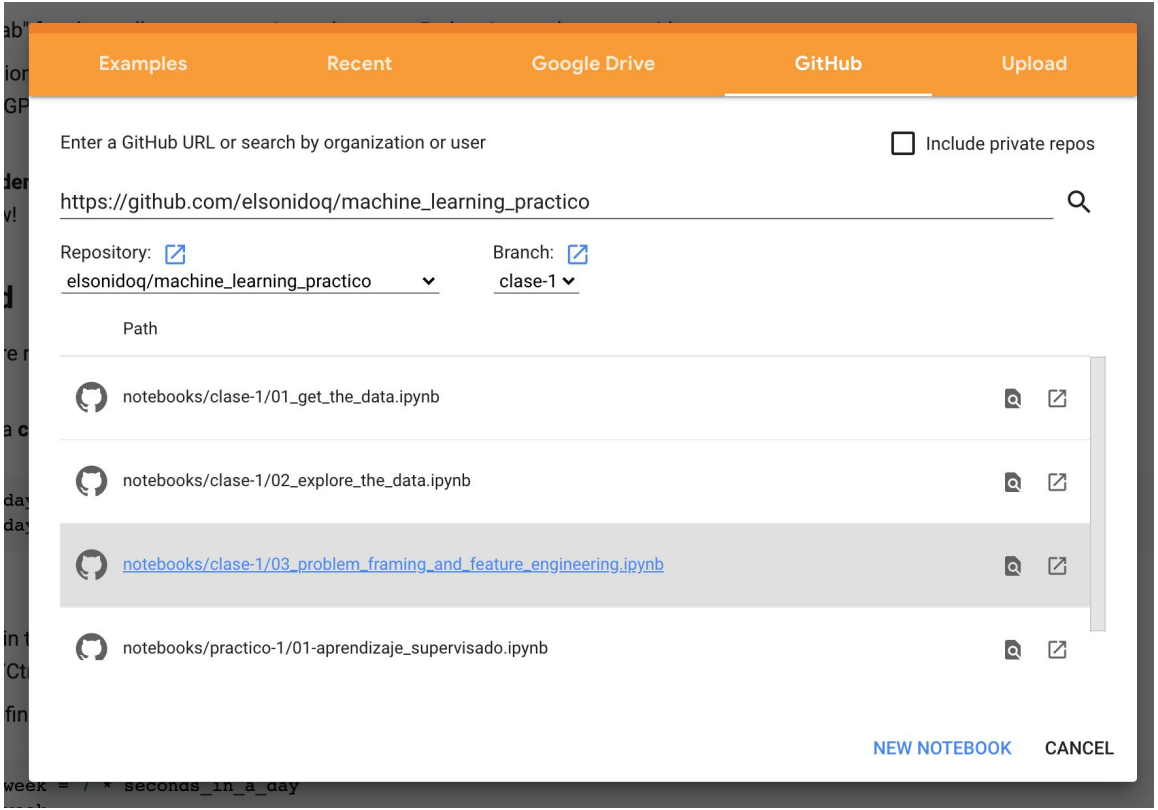
Te pregunto

- Cómo te diste cuenta?
- Y qué vas a hacer?

Hands on con toy examples!


Examples
Recent
Google Drive
GitHub
Upload
Enter a GitHub URL or search by organization or user
Include private repos
https://github.com/elsonidoq/machine_learning_practico
Repository:
elsonidoq/machine_learning_practico
Branch:
clase-2
Path
notebooks/clase-1/03_problem_framing_and_feature_engineering.ipynb
notebooks/clase-2/01-model-complexity.ipynb
notebooks/clase-2/02-simple-model-selection.ipynb
notebooks/clase-2/03-hyperparameter-search.ipynb
NEW NOTEBOOK
CANCEL

Entrenemos un baseline


Examples
Recent
Google Drive
GitHub
Upload
Enter a GitHub URL or search by organization or user
Include private repos
https://github.com/elsonidoq/machine_learning_practico
Repository:
elsonidoq/machine_learning_practico
Branch:
clase-1
Path
notebooks/clase-1/01_get_the_data.ipynb
notebooks/clase-1/02_explore_the_data.ipynb
notebooks/clase-1/03_problem_framing_and_feature_engineering.ipynb
notebooks/practico-1/01-aprendizaje_supervisado.ipynb
NEW NOTEBOOK
CANCEL

Metodología

Extrayendo las cosas que funcionan bien

Repeat code: a great way to have bugs

Load the data

Con lo que aprendimos del notebook anterior

In [2]:
```
import pandas as pd

title_basics = pd.read_csv(PATH / 'title.basics.tsv', sep='\t')

movie_gross = pd.read_csv(PATH / 'movie_gross.csv')
# Calculamos el id de pelicula
movie_gross['tconst'] = movie_gross.movie_imdb_link.apply(lambda x: x.split('/')[4])
# Deduplicamos los registros
movie_gross = movie_gross.groupby('tconst').gross.max().reset_index()

title_ratings = pd.read_csv(PATH / 'title.ratings.tsv', sep='\t')
```

```
/Users/przivic/anaconda3/lib/python3.8/site-packages/IPython/core/interactiveshell.py:3071: DtypeWarning: Columns (5)
have mixed types.Specify dtype option on import or set low_memory=False.
  has_raised = await self.run_ast_nodes(code_ast.body, cell_name,
```

In [3]:
```
from itertools import chain
from collections import Counter

def parse_genres(genres):
    if isinstance(genres, float) or genres == r'\N': return ['no-genre']
    else: return genres.split(',')


# Convertimos runtimeMinutes a float. No se puede tener una columna de tipo int con NaN
title_basics.runtimeMinutes = (
    title_basics.runtimeMinutes.apply(lambda x: np.nan if not x.isdigit() else x).astype(float)
)

title_basics['genres'] = title_basics.genres.apply(parse_genres)
```

In [179]:
```
title_basics = title_basics[
    # Dejamos tvSpecial, video y tvMovie por ahora, vamos a ver de que se tratan
    ~title_basics.titleType.isin(['tvEpisode', 'tvSeries', 'tvMiniSeries', 'videoGame', 'tvShort', 'short'])
    # Que tengan valor de runtimeMinutes
    & ~title_basics.runtimeMinutes.isna()
    # Menos de 3 horas y media para no descartar a titanic
    & (title_basics.runtimeMinutes <= 3.5 * 60)
    # Descartamos los shorts
    & title_basics.genres.apply(lambda x: 'Short' not in x)

]
```

Repeat code: a great way to have bugs

In [199]:
```
from datetime import datetime
from sklearn.base import BaseEstimator, TransformerMixin

class YearsAgo(BaseEstimator, TransformerMixin):
    def __init__(self):
        self.now = datetime.now().year

    def fit(self, X, y): return self

    def transform(self, X):
        res = []
        for e in X:
            res.append({'years_ago': self.now - int(e['startYear'])})
        return res
```

Repeat code: a great way to have bugs

```python
from collections import Counter

class DirectorFeatures(BaseEstimator, TransformerMixin):
    def __init__(self, min_cnt_movies = 2):
        self.min_cnt_movies = min_cnt_movies

    def fit(self, X, y):
        # Esto no es la forma mas elegante, pero es mas comodo y a esta altura priorizo la comodidad
        # Llevamos las cosas de nuevo a un DataFrame y calculamos features por director
        directors_stats = (
            pd.DataFrame(X)
              .groupby('director')
              .agg({
                  'tconst': 'count',
                  'averageRating': ['mean', 'max', 'min'],
                  'numVotes': ['mean', 'min', 'max']}
              )
        )

        # Para hacer flattening de las columnas
        # https://stackoverflow.com/questions/14507794/pandas-how-to-flatten-a-hierarchical-index-in-columns
        directors_stats.columns = [
            '_'.join(i)
            for i in zip(directors_stats.columns.get_level_values(1), directors_stats.columns.get_level_values(0))
        ]

        # Guardamos las estadisticas
        self.directors_stats_ = directors_stats

        # Diccionario con los datos para los directores comunes
        self.directors_stats_lk_ = (
            directors_stats[directors_stats.count_tconst >= self.min_cnt_movies].to_dict(orient='index')
        )

        # Valor default para los que consideramos que tenemos demasiado poca data
        self.default_ = directors_stats[directors_stats.count_tconst < self.min_cnt_movies].mean(0).to_dict()
```

Metodología

- Prototipar en notebook
- Extract a un archivo .py (o .R)
  - Cuando validamos que funciona
  - Cuando necesitamos ese código desde otra notebook
  - Cuando es demasiado código y hace la notebook muy larga

Vamos al PyCharm

Guia de ejercicios