# Clasificación de textos

Edgar Andrés Santamaría y Maitane Ruiz Monroy

EHU / UPV

## Objetivos

- Objetivo: Habiéndonos dado unas autopsias verbales, preprocesarlas y predecir el diagnostico (causa de la muerte).
- El texto como rasgo predictivo caracterizado por:
  - TF-IDF
  - PCA
- Clasificador usado para el Baseline y para realizar clasificadores combinados en paralelo:
  - Random Forest Classifier

## Tarea y Datos

- Descripción de la tarea:
  - La tarea consiste en combinar varios clasificadores para mejorar las conclusiones finales de predicción en una tarea de *multilabeling*.
- Datos a utilizar en la tarea:
  - Obtenidos de la página web de OSF Home `https://osf.io/xuk5q/`
- Clase a predecir *(pág 253)*
  - gs-text (diagnósticos de las autopsias, 48 clases distintas)

| Parámetros | Baseline | Combinados |
|---|---|---|
| learning_rate | 1e-06 | 1e-06 |
| AB: n_estimators | 2 | 10 |
| RF: n_estimators | 2 | 32 |
| n_components | 96 | 96 |
| norm | L1 | L1 |
| sublinear_tf | true | true |
| max_features | 32 | 32 |
| max_depth | 32 | 32 |
| n_splits | 3 | 3 |

## Librerías

- *mathplotlib* (generar gráficos)
- *numpy* (generar estructuras de datos)
- *sklearn* (proporciona algoritmos)

### Bibliografía

*Scikit-learn: Machine Learning in Python, Pedregosa et al., JMLR 12, pp. 2825-2830, 2011.*

## TF-IDF

TF-IDF es la división de la frecuencia de término y frecuencia inversa de documento.
Para evitar errores, se han convertido los vectores a *non-sparse*. Con *sublinear_tf* creamos un conjunto de vectores con distribución normal y con *norm* limitamos el espacio numérico de los vectores. *(pág 344, pág 1662)*

## PCA

PCA es una técnica utilizada reducir linealmente la dimensión.
Limitamos con *n_components* los atributos por vector a 96. *(pág 1572)*

## AdaBoosting

Es una técnica que utiliza una secuencia de clasificadores y los entrena repetidamente con datos de errores anteriores. *(pág 264, pág 1606)* Hemos alterado los siguientes parámetros:

- learning_rate = 1e-06 (tasa de aprendizaje)
- n_estimators = 10 (número de estimadores)

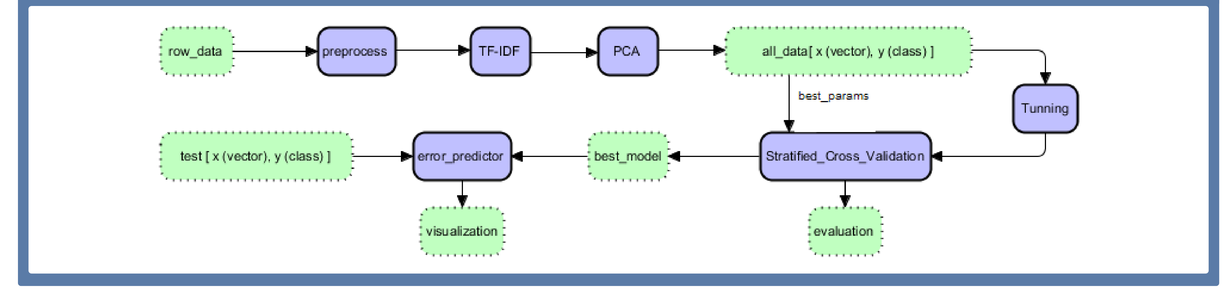
## Random Forest Classifier

Cada árbol del conjunto del bosque, se genera de los ejemplos del train (con remplazamiento).
Además, se aplican podas a cada nodo, en función de las características del conjunto. Se han utilizado 32 estimadores ajustando el parámetro *n_estimators* y también *bootstrap* poniendo el parámetro a *true*. *(pág 463)*

## Evaluación

- Cross_val_predict: Método que aplica la estrategia de cross-validation y devuelve predicciones de los datos de test en la fase de entrenamiento. *(pág 403, pág 2002)*
- StratifiedKFold: Método que proporciona un subconjunto de entrenamiento, con el mismo porcentaje de ejemplo para cada clase a predecir (muestreo estratificado). Con n_splits = 3, determinamos el número de entrenamientos por validación. *(pág 1974)*

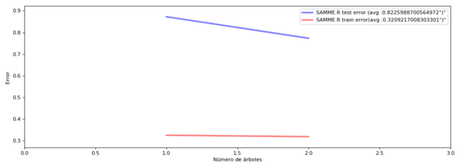
## Esquema del sistema




## Resultados experimentales

Los resultados que se muestran a continuación son los de Baseline y los de los clasificadores Random Forest combinados mediante la técnica Boosting y adecuadamente ajustados sus parámetros.
Para mostrar su mejora se han utilizado diferentes métricas: *Log_loss*, *accuracy*, *precision*, *f-score* con la ponderación de las clases minoritarias y *f-score* con la ponderación de las clases mayoritarias. *(pág 1877)*

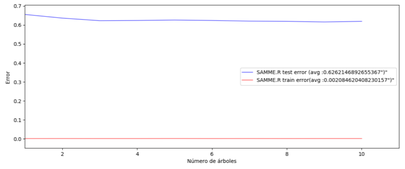
### Baseline




- Log_loss: 15.9363
- Accuracy: 0.1633
- F-score (micro): 0.1633
- F-score (macro): 0.1210
- Precision: 0.1633

El algoritmo inicial es malo (*log_loss*), y su tasa de acierto es baja (*accuracy*, *precision*), por último cabe destacar que la clasificación en aquellas clases menos entrenadas (*f-score* micro) no muestra notable diferencia con la clasificación en aquellas clases mas ejemplificadas (*f-score* macro).

### Clasificadores combinados




- Log_loss: 7.1737
- Accuracy: 0.3025
- F-score (micro): 0.3025
- F-score (macro): 0.2065
- Precision: 0.3025

Tras el ajuste del algoritmo logramos una mejora sustancial en el sistema de manera general (*log_loss*), y se mejora notablemente la tasa de acierto del sistema (*accuracy*, *precision*), por último cabe destacar la mejora de clasificación en aquellas clases menos ejemplificadas (*f-score* micro).

## Conclusiones

- Objetivos: Dada la complejidad de la tarea de *multilabeling* y el poco tiempo disponible para realizarla, creemos que predecir 4 de cada 10 diagnósticos de manera acertada es aceptable.
- Debilidades: el sistema requiere altas prestaciones para ajustarse de manera correcta.
- Fortalezas: El sistema es altamente eficiente.
- Mejoras: Aplicar distintos espacios de representación (*word embbedings*).