NORMALIZACIÓN, SELECCIÓN DE CARACTERÍSTICAS Y REDUCCIÓN DE LA DIMENSIONALIDAD

VALIDACIÓN CRUZADA

ALGORITMOS DE APRENDIZAJE SUPERVISADO

- **K-NN**
- **SVM**
- **NAIVE BAYES**
- **RANDOM FORESTS**

ENSEMBLE

Vamos a trabajar con un conjunto de datos formado por numerosas partidas de ajedrez recogidas del portal lichess.org, y que tiene como atributos:

- ID: identificador de la partida
- White: ajedrecista con las piezas blancas
- Black: ajedrecista con las piezas negras
- Result: resultado de la partida
- WhiteElo: elo (puntuación que indica la fuerza; cuanto mayor es, mejor es) del bando blanco. Se puede asumir que "?" es un elo de 0 puntos.
- BlackElo: elo del bando negro
- TimeControl: ritmo de juego (por ejemplo, "x+y" equivale a un ritmo de juego de "x" segundos para toda la partida con un incremento de "y" segundos por jugada)
- ECO: código estandarizado para cualquier apertura determinada (lista completa en https://www.365chess.com/eco.php)
- Opening: nombre de la apertura
- Termination: motivo por el que ha terminado la partida
- MovesCount: número de jugadas

El conjunto de datos se encuentra en el siguiente enlace:
https://drive.google.com/file/d/1QsFdlwhe4BQpSK-Dg5z9K-RN5PWuIB-t/view?usp=sharing

Se pide desarrollar un código (archivo setup.ipynb) que realice los siguientes pasos:

- Cargar el conjunto de datos y visualizarlo. Hacer todas las operaciones de preprocesamiento que se consideren necesarias (el campo TimeControl no hay que modificarlo) y visualizar nuevamente el conjunto de datos. Llamaremos a este conjunto de datos *conjunto de datos original*.
- Añadir los siguientes atributos al conjunto de datos original:
  - GameElo: elo de la partida. GameElo = CEIL((WhiteElo + BlackElo) / 2)
  - EloDiff: diferencia de elo entre los bandos. EloDiff = WhiteElo – BlackElo
- Estandarizar el conjunto de datos original a escala unitaria (media = 0 y varianza = 1) y visualizarlo. Llamaremos a este conjunto de datos *conjunto de datos estandarizado.*
- Normalizar el conjunto de datos original a un rango fijo [0, 1] y visualizarlo. Llamaremos a este conjunto de datos *conjunto de datos normalizado.*

- Aplicar un PCA de 0.95 al conjunto de datos original y visualizarlo. Llamaremos a este conjunto de datos *conjunto de datos originalPCA95.*
- Aplicar un PCA de 0.80 al conjunto de datos original y visualizarlo. Llamaremos a este conjunto de datos *conjunto de datos originalPCA80.*
- Aplicar un PCA de 0.95 al conjunto de datos estandarizado y visualizarlo. Llamaremos a este conjunto de datos *conjunto de datos estandarizado PCA95.*
- Aplicar un PCA de 0.80 al conjunto de datos original y visualizarlo. Llamaremos a este conjunto de datos *conjunto de datos estandarizado PCA80.*
- Aplicar un PCA de 0.95 al conjunto de datos normalizado y visualizarlo. Llamaremos a este conjunto de datos *conjunto de datos normalizado PCA95.*
- Aplicar un PCA de 0.80 al conjunto de datos original y visualizarlo. Llamaremos a este conjunto de datos *conjunto de datos normalizado PCA80.*

- Configurar una validación cruzada de k=5 iteraciones para cada uno de los conjuntos de datos considerados: original, estandarizado, normalizado, y las distintas versiones de PCA.

Se pide desarrollar un código (archivo train.ipynb) que genere los modelos entrenados para cada iteración de la validación cruzada de cada uno de los algoritmos de aprendizaje supervisado considerados: K-NN, SVM, Naive Bayes y Random forests. El atributo *Result* se utilizará como clase.

Se pide desarrollar un código (archivo eval.ipynb) que, para cada modelo, realice las predicciones de los tests en cada iteración de la validación cruzada y genere su rendimiento. A la hora de hacer las predicciones de cada muestra de test se tienen que obtener las probabilidades estimadas de pertenencia a cada clase y almacenarlas. Implementar los ensembles Votación, Media y Mediana.

Se pide desarrollar un código (archivo results.ipynb) que cargue el rendimiento de cada modelo en cada iteración de la validación cruzada de cada conjunto de datos considerado (original, estandarizado, normalizado, y las distintas versiones de PCA (originalPCA95, originalPCA80, estandarizadoPCA95, estandarizadoPCA80, normalizadoPCA95 y normalizadoPCA80)). Tras esto, el código deberá crear tablas y figuras que permitan hacer una comparativa del rendimiento de los métodos.

Finalmente, se tiene que redactar un informe con los resultados obtenidos y las conclusiones que se pueden extraer. El informe tiene que estar escrito en Latex usando la plantilla LNCS.

MÉTRICAS

Implementa todas las siguientes métricas de rendimiento:

Métrica	**Definición**
F1-score (Fm)	$Fm = 2 \times (PR \times RC) / (PR + RC)$
Sensibilidad (S)	$S = TP / (TP + FN)$
Exactitud (Acc)	$Acc = (TP + TN) / (TP + FP + FN + TN)$
Especificidad (SP)	$SP = TN / (FP + TN)$
Recall (RC)	$RC = TP / (TP + FN)$
Precisión (PR)	$PR = TP / (TP + FP)$

Tasa de falsos negativos (FNR)	FNR = FN / (TP + FN)
Tasa de falsos positivos (FPR)	FPR = FP / (FP + TN)

Obtén también la curva ROC y el AUC (área bajo la curva):
https://es.wikipedia.org/wiki/Curva_ROC

Se tiene que entregar un único archivo comprimido que contenga los siguientes archivos:

- El informe en formato pdf generado por el proyecto Latex.
- Archivo comprimido (.rar o .zip) con todos los archivos del proyecto Latex.
- El código (en formato ipynb) que se ha desarrollado.
- Un video (en formato mp4) con la ejecución del código.
- Declaración explícita en la que se asuma la originalidad del trabajo entregado.