**Informe del progreso de TFG, fase 1 (2017.10.04-2017.10-26)**

1. Trabajo preparatorio:
   * Herramientas de programación:
     + Anaconda Python 3.6
     + Eclipse-Pydev
   * Mirar tutoriales de “Machine Learning”
   * Lectura del código del trabajo anterior (sobretodo el “grade\_prediction.ipynb”)
     + Compresión de la estructura y flujo de ejecución
     + Compresión de las funciones y variables
     + Pasar el código des del “ipynb” al “.py” del Pydev (modularización y coherencia) para poder debuggear el código a profundidad.
   * Reutilización y adaptación del código del trabajo anterior
     + Se ha reutilizado la mayor parte del código que se corresponde a la carga y limpieza de datos. También se aplica un reemplazo al dataframe resultante, de valores NaN por 5.0 de float.
     + Se ha modificado el código reutilizado para que sea más legible y comprensible: estructuración, cambio de nombre de las variables, el añadido de comentario.
     + Se utiliza el editor Pydev de Eclipse para desarrollar el programa.
2. Implementación de funcionalidades:

Para esta vez, hay que implementar dos funcionalidades: un ranking de asignaturas por número ordinal y un clasificador binario de suspensos. Para empezar, vamos a tratar el caso de un sólo grado, pues se cargan los datos correspondientes a las calificaciones de los alumnos de “mates”.

* + Dimensión de datos después de la limpieza:

Se leen los datos des del fichero “qualifications\_mates\_info.csv”, por el nombre se puede deducir que los datos del grado de mates e informática están en un fichero, posteriormente se hará una separación.

|  |  |  |
| --- | --- | --- |
| **Variable** | **Descripción** | **Dimensión** |
| qual | Carga inicial | 19621 x 5 |
| qual2process | Eliminación de “id\_assig”==NaN | 18383 x 5 |
| filtered\_qual | Elección del grado a tratar | 9227 x 5 |
| pivot\_year1  pivot\_year2 | Ordenación de la tabla por nota(value), id\_alumne(index), id\_assig(column) | 516 x 10 |
| pivot conc | Concatenación de pivot\_year1 y pivot\_year2; eliminación de filas thresh=11 de NaNs | 231 x 20 |
| pivot\_conc | Eliminación de filas cuyas sumas de las notas del segundo año = 0 | 221 x 20 |
| df\_year1 | pivot\_conc[: , :10] | 221 x 10 |
| df\_year2 | pivot\_conc[: , 10:20] | 221 x 10 |

* + Las dos aproximaciones de predicción para ambas funcionalidades
    - Ranking mediante predicción de notas: notas (input) -> **predicción** -> notas (predichas) -> conversión -> ranking (resultado final)
    - Predicción de ranking: notas (input) -> conversión -> ranking (inicial) -> **clasificación** -> ranking (resultado final)
    - Aprobado/Suspenso mediante predicción de notas: notas (input) -> **predicción** -> notas (predichas) -> conversión -> estado binario (resultado)
    - Clasificación binaria de aprobado/suspenso: notas (input) -> conversión -> estado binario (inicial) -> **clasificación** -> estado binario (resultado)
  + Predictores empleados:

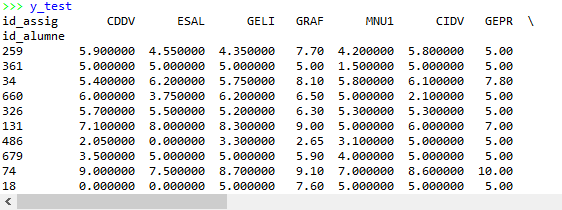
En total tenemos tres casos:

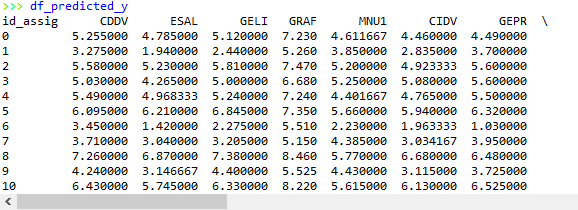
|  |  |  |
| --- | --- | --- |
| **Casos** | **Tipo de datos** | **Predictor** |
| Predicción de notas | Float64 | RandomForestRegressor |
| Clasificación de ranking | Int | RandomForestClassifier |
| Clasificación de suspensos | boolean | RandomForestClassifier |

Ambos predictores son importados des del paquete “sklearn.ensemble”.

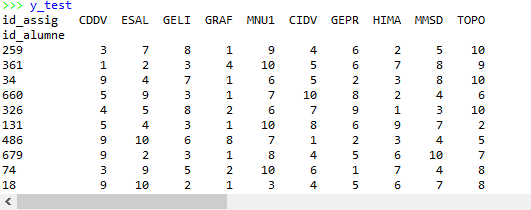
* + Los módulos (ficheros .py) del programa:
    - main.py: carga los datos y lanza en menú.
    - loadData.py: carga los datos y los procesa para que estén en condición para la predicción.
    - randomForest.py: implementa la predicción RandomForest de notas y clasificación de ranking; también tiene algunas funciones de soporte: pasar de notas al ranking, cálculo de puntuaciones.
    - binClassifier.py: implementa la predicción RandomForest de notas y clasificación de suspensos; también tiene algunas funciones de soporte: pasar de notas al estado binario, cálculo de puntuaciones.
  + El output de predicciones y comparación de performances:
    - El output de predicciones:

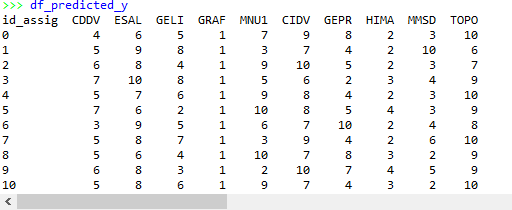
Qp\_t vs Qp\_p



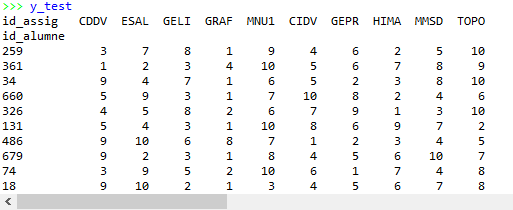


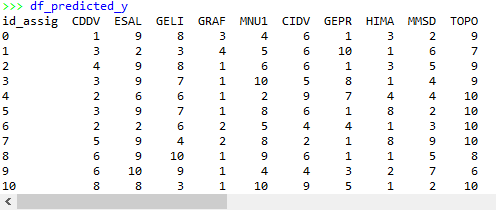
Qp\_rt vs Qp\_rp



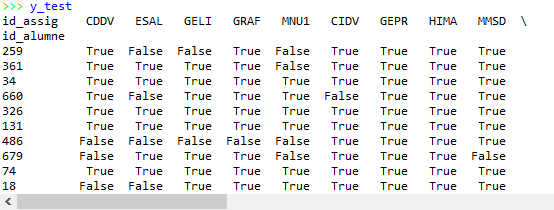


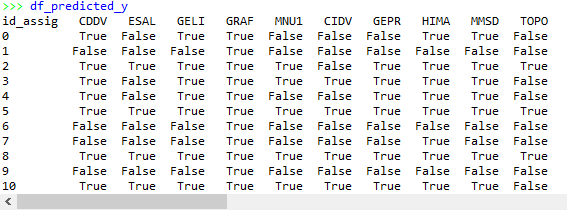
Rp\_t vs Rp\_p



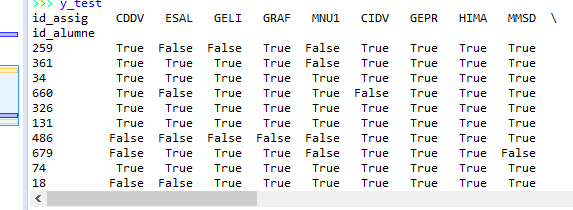


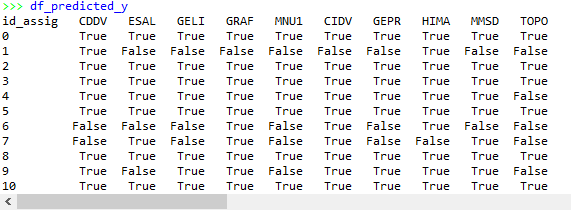
Qp\_bt vs Qp\_bp





Bp\_t vs Bp\_p





* + - Comparación de performances:

|  |  |  |
| --- | --- | --- |
| **Tipo de predicción** | **Medida de Validación** | **Puntuación** |
| Predicción de nota | Clf.score | 0.3055 |
| Predicción de nota | cross\_validation.cross\_val\_score | 0.1759 |
| Predicción de nota->ranking | metrics.accuracy\_score | 0.1375 |
| Predicción de ranking | metrics.accuracy\_score | 0.1393 |
| Predicción de notas->binario | metrics.accuracy\_score | 0.6804 |
| Clasificación binaria | metrics.accuracy\_score | 0.7196 |

1. Tareas para la próxima cita:
   * Apuntes en la cita
   * Procesar datos de los grados “Informática” y “Derecho”
   * Mejorar las precisiones de predictores (GridSearch), usar diferentes métricas.
   * Etc
2. Dudas:
   * ranking prediction: repeated class
   * similarity/score computing type
   * which approach is selected?
   * guardar cada link?