

Smart_Hours.

Documentación

Integrantes del grupo: Pau Albiol, David Álvarez, Guillermo Creus.

Mesa: 4.

Fecha entrega: 30/05/2018.

Índice.

1. Objetivos.
2. Alcance.
3. Análisis del problema y descripción de la solución.
4. Herramientas e implementación.
5. Planificación y costes.
6. Resultados y conclusión.
7. Bibliografía.

1. Objetivos.

Creación de un horario de estudio que se adapte a los objetivos académicos del cliente.

2. Alcance.

Funcionalidades

Smart_Hours genera un horario y ofrece cuatro tablas y un gráfico que contienen datos de interés enfocados al cliente. Las horas del horario están distribuidas de manera adecuada para que cliente pueda cumplir con sus objetivos académicos. Para crear el horario, el cliente debe interactuar con la web rellenando tres formularios.

1. El primer formulario, consiste en indicar qué horas de la semana (de lunes a viernes, de 9:00 h a 23:00 h) tiene ocupadas el cliente para que no se asignen horas de estudio en dichas horas no disponibles.
2. En el segundo, éste debe elegir su rendimiento e introducir las asignaturas que quiere en su horario. El parámetro del rendimiento indica "el objetivo". Hay cuatro opciones a elegir:
 - **Mínimo:** Aprobar con 5 las asignaturas elegidas.
 - **Medio:** Mantener la nota media del cliente o aprobar con 5 en caso de que esta media sea de suspenso.
 - **Alto:** Aumentar la nota media en un 10% o aprobar con 5.5 en caso de que esta media sea de

suspense.

- **Manual:** El cliente puede introducir qué notas quiere para cada asignatura.

3. En el tercer formulario se pide la nota de selectividad y las diez asignaturas de primero (es el Historial académico del cliente). Este parámetro permite predecir las notas que va a sacar el cliente y calcular cuán lejos está de sus objetivos.

Finalmente se generará el horario y se mostrarán las distintas tablas. Estas son:

- **Fin de semana:** Tabla que indica las horas a dedicar durante el fin de semana si las horas de lunes a viernes son insuficientes para que el cliente cumpla sus objetivos.
- **Notas:** Tabla donde de cada asignatura elegida, se da información sobre la nota predicha, la nota deseada (es decir el objetivo) y la media global.
- **Distribución de horas de estudio:** Gráfico tipo *3d-Pie* que representa en porcentajes, el tiempo de estudio de cada asignatura (respecto al total de horas de estudio).
- **Estudiantes con notas más cercanas:** Tabla que compara las notas del Historial académico del cliente con el de los doce estudiantes más cercanos a él.
- **Notas de las asignaturas matriculadas de los estudiantes más cercanos:** Tabla que compara las notas predichas del cliente frente a las notas de los doce estudiantes más cercanos a él.

Otras funcionalidades (más técnicas)

El horario no muestra todas las horas si estas están vacías. Es decir, si el cliente estudia a partir de las 18:00 h, el horario no mostrará las horas vacías de 9:00 h a 18:00 h. Por comodidad, para rellenar los checkboxes en el apartado de selección de Horas ocupadas, es posible clicar y arrastrar con el ratón haciendo menos tedioso este formulario.

Condiciones y limitaciones

Como mínimo el cliente debe introducir una nota en el parámetro de Historial académico. Sólo se aceptan caracteres numéricos. En caso de decimales se debe indicar con un punto. (Por ejemplo: 7.5).

La tabla del horario abarca de lunes a viernes, de 9:00 h a 23:00 h. Si el número de horas de estudio de lunes a viernes es insuficiente, las horas restantes se asignan en el fin de semana.

3. Análisis del problema y descripción de la solución.

Descripción de los problemas

1. Problema del filtraje:

Se ha partido de tres bases de datos que contenían resultados académicos de estudiantes del ETSEIB. La primera de ellas contenía las notas de selectividad, la segunda las notas de la fase inicial y la tercera las notas de la fase no inicial (hasta Q1 17/18). Debido que no siempre hay los mismos estudiantes, hay más datos de los necesarios y puede que un estudiante tenga múltiples notas de la misma asignatura (puede ser repetidor) se ha procedido a un filtraje de datos.

2. Problema de predicción:

Dado que un factor para calcular las horas de estudio es la nota predicha del cliente, obviamente se necesitan realizar predicciones (de una variable numérica continua). Para ello se ha utilizado el método de K-Vecinos más cercanos.

3. Problema de asignación de horas:

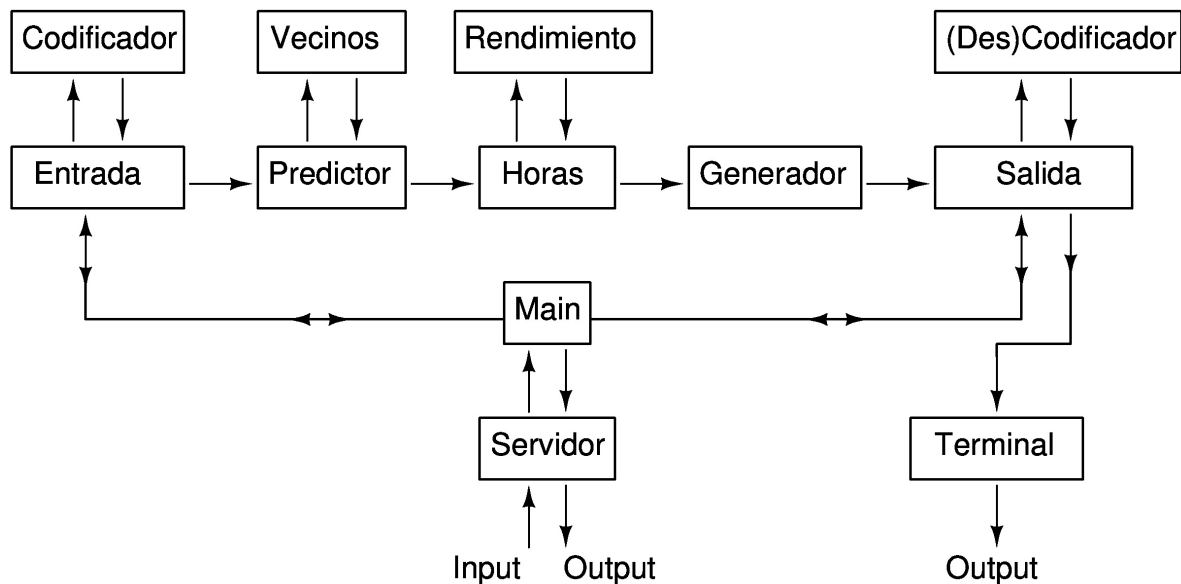
Una vez predichas las notas, se deben comparar con los objetivos del cliente y traducir a “horas de estudio”.

4. Problema de ordenación de horas:

Obtenidas las horas de estudio para cada asignatura, estas se deben distribuir en un horario de forma ordenada y lógica (evitar problemas del tipo: en un día hay que estudiar 4 horas seguidas de la misma asignatura).

Solución propuesta (Planteamiento global de funcionamiento)

El funcionamiento general del código, desde que el usuario introduce el input hasta que el output se genera en pantalla está resumido en el siguiente diagrama.



Main

Función principal que controla el programa. El funcionamiento es el siguiente:

1. Convertimos el input que el usuario entra a un input utilizable por el programa (con el script de entrada).
2. Predecimos las notas de las asignaturas que el usuario se ha matriculado (con el script predictor).
3. Obtenemos las notas deseadas de acuerdo con el rendimiento introducido por el usuario (con el script rendimiento).
4. Con los datos obtenidos calculamos las horas de estudio (con el script horas).
5. Generamos el horario de estudio (con el script generador).
6. Imprimimos resultados por la terminal (con el script terminal).
7. Generamos output utilizable por el servidor (con el script salida).

Entrada y Salida

Distintas funciones para dar formato al input y al output.

Codificador

Consiste en un codificador y en un decodificador que permiten codificar el nombre de una asignatura con su código (CODAS) y al revés. Los datos se encuentran en 'databases/asignaturas.csv'.

Vecinos

Encuentra los alumnos que están más cercanos al usuario (los datos están en 'databases/data.csv'). Para ello, buscamos a los alumnos que tienen una distancia más pequeña al usuario (entendiendo la distancia como las diferencias de las notas elevadas al cuadrado).

Predictor

Usando el script de **vecinos** hace una predicción de las notas que el usuario obtendrá en las asignaturas que se ha matriculado. Para ello (para cada asignatura) hacemos una media de las notas que han obtenido los alumnos más cercanos al usuario. Además, para mejorar la fiabilidad, calculamos

Rendimiento

Calcula notas deseadas de acuerdo con el rendimiento introducido por el usuario:

- **Mínimo.** Aprobar con 5 todas las asignaturas.
- **Medio.** Mantener nota media del usuario o aprobar con 5 en caso de que su media sea de suspenso.
- **Alto.** Aumentar nota media del usuario en un 10% o aprobar con 5.5 en caso de que su media sea de suspenso.
- **Manual.** El usuario ya habrá introducido manualmente las notas que desea.

Horas

Dadas las notas esperadas (predictor) y las notas deseadas (rendimiento), calcula el número de horas que el usuario debería estudiar por asignatura a la semana. Para ello, usamos como base que cada crédito ECTS (*European Credit Transfer and Accumulation System*) corresponde a unas 20-25 horas de trabajo por parte del estudiante. Nosotros, descontando el número de horas de trabajo en el aula, hacemos la correspondencia 1 ECTS - 0,8 horas estudio semanales (para una persona media).

Ahora bien, comparando la nota deseada con la esperada (predicha por nuestro programa), ajustamos este número de horas a uno más realista y personalizado para el usuario.

Generador

Con las restricciones de horario introducidas por el usuario y con las horas de estudio semanales por asignatura (dadas por script horas), genera un horario de estudio. Distribuyendo las horas entre la semana y también en el fin de semana.

4. Herramientas e implementación.

Herramientas de soft usadas: Para el filtrado de datos se ha usado pandas, una librería de Python destinada al análisis de datos que proporciona unas estructuras de datos flexibles y que permiten trabajar con ellos de forma eficiente. Más concretamente se ha trabajado con DataFrame (los datos se estructuran en forma de tablas).

El programa se ha desarrollado en Python...?

Para crear la aplicación web se ha usado Flask. Flask es un framework minimalista escrito en Python que permite crear aplicaciones web con un mínimo número de líneas de código (basado en el motor de templates Jinja2).

La página web funciona con archivos html y javascript... Para trabajar en equipo de eficiente se creó un [repositorio](https://github.com/David-98/Horario) público en GitHub (<https://github.com/David-98/Horario>). Github es una plataforma de desarrollo colaborativo para alojar proyectos utilizando el controlador de versiones Git.

Descripción de la solución concreta adaptada:

5. Planificación y costes.

Para realizar el proyecto de Smart_Hours, una de las primeras cosas que se creó fue un plan de ruta cuya planificación abarcaba cerca de dos meses de desarrollo. Las horas dedicadas en Smart_Hours han sido para cada integrante:

- 5 horas de clase de teoría dónde se aprendió sobre el uso de pandas y flask.
- Un promedio de 55 horas de autoaprendizaje.

Es decir, en total unas 180 horas.

El único coste del proyecto ha sido el consumo energético para alimentar los ordenadores. Suponiendo que un ordenador usa 100W por 1 hora diaria a 0,12 € el kWh, calculando para un integrante el coste mensual es de 0,43 € al mes. Dado que el proyecto ha durado 2 meses y Smart_Hours ha sido desarrollado por 3 integrantes, el coste energético estimado es de 2,6 €.

6. Resultados y conclusiones.

Para realizar este proyecto, una buena comunicación y coordinación entre el equipo ha sido imprescindible. La estrategia de haber dividido y planificado el proyecto en pequeños problemas y asignar roles a cada integrante ha resultado muy eficiente. Ha simplificado las cosas pues cada integrante se focalizaba en un problema y en caso de no saber continuar, entre todos se ha buscado una solución o propuesto alternativas para solucionar (o en algún caso evitar) el problema.

Un gran impacto de este proyecto ha sido el de autoaprendizaje pues siempre que aparecía algún inconveniente, en internet se hallaba la solución.

También se han adquirido competencias genéricas sobre la comunicación oral y escrita. Ha habido una clara evolución en cómo se han realizado las presentaciones orales y el estilo de los PDF's que se proyectaban.

7. Bibliografía.

<https://stackoverflow.com/>

<https://www.w3schools.com/>

<https://pandas.pydata.org/pandas-docs/stable/10min.html>