Treball de Fi de Grau

## Grau en Enginyeria en Tecnologies Industrials

# Aplicació web pel càlcul del rendiment acadèmic en la Fase Inicial de grau segons procedència geogràfica

**MEMÒRIA**


**Autor:** David Mirabet Manjón
**Director:** Lluis Solano Albajes

**Convocatòria:** Juny 2017


Escola Tècnica Superior
d'Enginyeria Industrial de Barcelona

# Resum

Aquest Treball de Fi de Grau tracta sobre la creació d'una aplicació web per a la consulta d'informació estadística del rendiment acadèmic durant la fase inicial del Grau en Enginyeria en Tecnologies Industrials de l'Escola Tècnica Superior d'Enginyeria Industrial de Barcelona, de la Universitat Politècnica de Catalunya.

L'aplicació permet consultar la previsió de la probabilitat d'aprovar una assignatura i la seva nota durant el primer quadrimestre, així com les convocatòries necessàries per superar-la, basant-se amb la nota d'accés a la universitat i la procedència geogràfica de l'alumne. A més, l'aplicació ofereix estadístiques sobre la quantitat d'alumnes provinents de cada comarca catalana i la quantitat d'alumnes que abandonen el grau durant el primer any.

S'empra l'agrupació per codis postals de Catalunya i la classificació amb arbres de decisió a més d'estadística descriptiva. El processament de dades es duu a terme amb el paquet Pandas de Python amb el que s'implementen els corresponents algoritmes necessaris per a filtrar les dades necessàries i obtenir-ne de noves que en són d'interès.

Amb els llenguatges informàtics HTML, CSS i JavaScript es modela una interfície gràfica interactiva i visualment atractiva on l'usuari pot navegar consultant els estudis realitzats.

Finalment, s'exposen la planificació temporal, els costos associats al projecte i s'avalua el seu impacte social i ambiental.

# Sumari

# Llistat de figures i taules

# *1.* GLOSSARI

## Paràmetres

$Y$ Variable categòrica resposta d'un arbre de classificació.

$H$ Entropia d'un arbre de classificació.

$A,B,C$ Nodes als que se'ls hi aplica una pregunta nodal.

$Pr(\cdot)$ Màxima probabilitat.

## Sigles i acrònims

| | |
|---|---|
| ETSEIB | *Escola Tècnica Superior d'Enginyeria Industrial de Barcelona* |
| CSV | *comma-separated values* |
| ECTS | *European Credit Transfer and Accumulation System* |
| HTML | *Hypertext Markup Language* |
| CSS | *Cascading Style Sheets* |
| UPC | *Universitat Politècnica de Catalunya* |
| Idescat | *Institut d'Estadística de Catalunya* |
| *PAU* | *Proves d'Accés a la Universitat* |
| *GETI* | *Grau en Enginyeria en Tecnologies Industrials* |
| SGV | *Scalable Vector Graphics* |

# *2.* PREFACI

## 2.1. Origen del projecte i motivació

L'interès per assolir un nivell superior d'aprenentatge en el llenguatge de programació Python i conèixer-ne de nous tot obtenint una visió més amplia del món de la programació i les oportunitats que ofereix són l'origen del projecte i la principal motivació per realitzar-lo.

A més, la possibilitat de posar en practica els coneixements adquirits en les assignatures d'Informàtica i Estadística realitzades durant el grau complementen aquesta motivació que resulta en un multidisciplinari projecte atractiu i enriquidor a nivell personal.

## 2.2. Requeriments previs

Els requeriments previs necessaris són els coneixements adquirits en les assignatures d'Informàtica i Estadística així com una base sòlida en el llenguatge de programació web.

# 3. INTRODUCCIÓ

## 3.1. Objectius del projecte

L'objectiu principal del projecte és la creació d'una aplicació web per al tractament estadístic de resultats acadèmics. L'aplicació esta enfocada als estudiants catalans que volen matricular-se per estudiar el Grau en Enginyeria en Tecnologies Industrials. Interactuant amb l'aplicació els nous estudiants poden consultar la dificultat que tindran en aprovar les assignatures del primer quadrimestre de grau la qual està basada en els antecedents corresponents a la regió geogràfica d'on prové el propi estudiant.

## 3.2. Estructura del projecte

A fi d'assolir els objectius, s'enfoca el treball des de dos dominis interconnectats entre si: la programació i l'estadística. Es partirà de tres fitxers en format .csv, els dos primers contenen totes les dades acadèmiques necessàries proporcionades per Prisma, el servei de l'Àrea de Docència de l'UPC, i el tercer conté la divisió postal de Catalunya proporcionada per l'Institut d'Estadística de Catalunya (Idescat).

En primer lloc, es tria la llibreria Pandas de l'intèrpret de programació Python amb la qual es realitzaran tots els algoritmes de processament i filtratge de dades per tal d'eliminar aquelles que no resultin necessàries i obtenir-ne de noves que si siguin d'interès. També es farà servir per aplicar tècniques de mineria de dades com els arbres de decisió.

Finalment, es trien els llenguatges (HTML, CSS, JavaScript) per a dissenyar i crear la interfície gràfica interactiva amb la que l'usuari pot interactuar i consultar. També s'usarà la llibreria "Flask" de Python per interconnectar l'aplicació amb l'estudi estadístic.

Seguidament es mostra el diagrama de blocs del projecte:

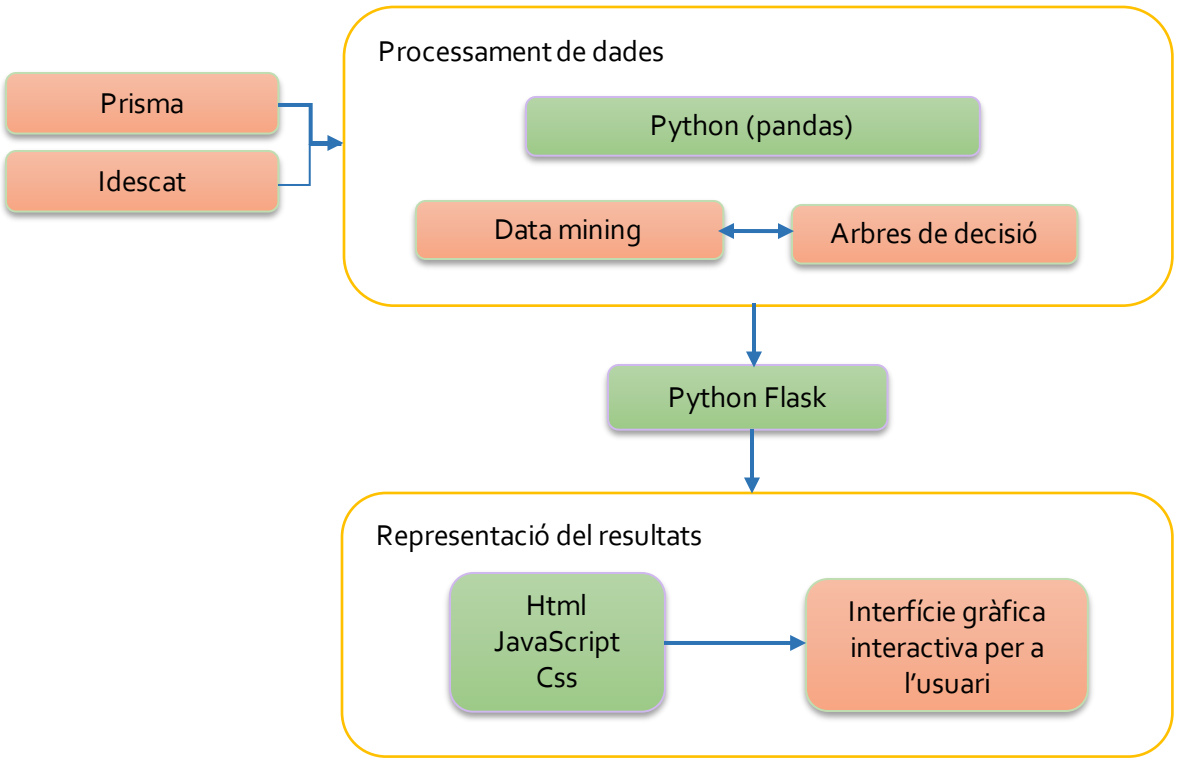



*Figura 3.1 Diagrama de blocs del projecte.*

## 3.3. Abast del projecte

Aquest treball recull la idea del que podria ser un treball més ampli i extens. L'estudi estadístic es restringeix a l'àrea geogràfica de Catalunya i a estudiants del GETI a l'ETSEIB durant el primer any de grau que han entrat mitjançant les Proves d'Accés a la Universitat per a estudiants provinents de Batxillerat. Així doncs, no es tractaran dades referents a estudiants d'altres cursos, d'altres graus, d'altres facultats i provinents de fora de Catalunya. A més, l'estudi està limitat per la quantitat d'informació coneguda i no confidencial sobre l'estudiant abans de cursar el grau.

# 4. PUNT DE PARTIDA

Es disposa inicialment de quatre fitxers en format *csv,* dos amb les dades acadèmiques necessàries per a l'estudi proposat i un més amb la divisió postal de Catalunya. El format *csv* (de l'anglès comma-separated values) és una tipologia de documents emprada per a la representació de les dades en forma de matriu o taula.

Tanmateix, el primer fitxer de dades acadèmiques (QualifINDfaseini16.csv) està compost per 33161 files i 10 columnes. Cada fila conté la informació d'una convocatòria d'una assignatura d'un alumne de l'ETSEIB. L'arxiu recull totes les convocatòries, assignatures i estudiants fins el 2015 inclòs.

La Primera columna conté el codi de l'expedient de l'alumne, la segona el codi de l'assignatura i la tercera el nombre de crèdits ECTS corresponents a l'assignatura. La quarta i la cinquena columna indiquen respectivament l'any i el quadrimestre de la convocatòria. La sisena columna informa de si l'estudiant ha superat o no l'assignatura, i la setena, la vuitena i la novena columna mostren diferents criteris d'avaluació. Finalment, la desena columna indica del grup classe de l'estudiant.

| 1 | CODEX | CODI_UPC_UD | CR | CURS | Q | APR | NP | NC | NF | GR |
|---|---|---|---|---|---|---|---|---|---|---|
| 2 | 227332 | 240025 | 7.5 | 2010 | 2 | S | 7.7 | 7.7 | 7.7 | 51 |
| 3 | 230686 | 240025 | 7.5 | 2010 | 2 | S | 8.8 | 8.8 | 8.8 | 51 |
| 4 | 232356 | 240025 | 7.5 | 2010 | 2 | S | 5.7 | 5.7 | 5.7 | 51 |
| 5 | 227742 | 240025 | 7.5 | 2010 | 2 | N | 3.7 | 3.7 | 3.7 | 42 |
| 6 | 228884 | 240025 | 7.5 | 2010 | 2 | S | 5.4 | 5.4 | 5.4 | 42 |
| 7 | 227522 | 240025 | 7.5 | 2010 | 2 | N | 0.4 | 0.4 | 0.4 | 42 |
| 33154 | 289296 | 240024 | 4.5 | 2015 | 2 | S | 5.5 | 5.5 | 5.5 | 80 |
| 33155 | 322193 | 240023 | 6 | 2016 | 1 | S | 6.1 | 6.1 | 6.1 | 20 |
| 33156 | 322074 | 240013 | 6 | 2016 | 1 | S | 5.2 | 5.2 | 5.2 | 20 |
| 33157 | 322027 | 240021 | 6 | 2016 | 1 | S | 5.7 | 5.7 | 5.7 | 20 |
| 33158 | 308056 | 240015 | 6 | 2015 | 2 | S | 5.8 | 5.8 | 5.8 | 21 |
| 33159 | 304010 | 240015 | 6 | 2015 | 2 | S | 6.3 | 6.3 | 6.3 | 12 |
| 33160 | 304035 | 240015 | 6 | 2015 | 2 | S | 5 | 5 | 5 | 12 |
| 33161 | 308061 | 240015 | 6 | 2015 | 2 | S | 7.3 | 7.3 | 7.3 | 12 |

*Figura 4.1 Dades del fitxer QualifINDfaseini16.csv*

D'altra banda, el segon fitxer de dades acadèmiques (DadespersIND16.csv) conté dades personals i acadèmiques preuniversitàries dels estudiants del Grau en Enginyeria en Tecnologies Industrials de totes les convocatòries fins 2015 (inclòs). Està compost per a 3353 files, on cada fila inclou la informació respectiva a un estudiant, i 6 columnes.

La primera, segona i tercera columna contenen respectivament el codi de l'estudiant, el sexe i el codi postal familiar. La quarta columna mostra l'any d'accés i la cinquena la nota de les PAU. Finalment, la darrera columna conté el codi postal del centre escolar.

| 1 | CODEX | SEX | CPF | ANY | SELE | CPE |
|---|---|---|---|---|---|---|
| 2 | 230661 | D | 8328 | 2010 | 10.05 | 8034 |
| 3 | 230310 | H | 8390 | 2010 | 12.336 | 8390 |
| 4 | 229174 | H | 7013 | 2010 | 10.386 | |
| 5 | 228108 | H | 17310 | 2010 | 10.294 | 17310 |
| 6 | 226494 | H | 8005 | 2010 | 11.028 | 8005 |
| 3349 | 308055 | H | 12002 | 2015 | 10.36 | |
| 3350 | 308056 | H | 8021 | 2015 | 10.638 | 8034 |
| 3351 | 308058 | H | 8906 | 2015 | 11.25 | 8950 |
| 3352 | 308060 | H | 8800 | 2015 | 11.078 | 8800 |
| 3353 | 308064 | H | 8172 | 2015 | 10.93 | 8022 |

*Figura 4.2 Dades del fitxer DadespersIND16.csv*

El darrer dels fitxers correspon a la divisió postal per municipis i comarques de Catalunya (Corresp_cques_CAT_codpost.csv) proporcionat per Idescat (1). Està compost per 1360 files, cadascuna de les quals correspon a un municipi de Catalunya, i 5 columnes.

La primera columna correspon al codi del municipi i la segona al seu nom. La tercera indica el codi de comarca mentre que la quarta és el nom de la comarca. Finalment, la cinquena corresponent al codi postal corresponent a cada municipi.

| 1 | CODMUN | NOMMUN | CODCCA | NOMCCA | CODPOST |
|---|---|---|---|---|---|
| 2 | 430017 | Aiguamúrcia | 01 | Alt Camp | 43714 |
| 3 | 430017 | Aiguamúrcia | 01 | Alt Camp | 43815 |
| 4 | 430056 | Alcover | 01 | Alt Camp | 43460 |
| 5 | 430108 | Alió | 01 | Alt Camp | 43813 |
| 6 | 430347 | Bràfim | 01 | Alt Camp | 43812 |
| 1355 | 082592 | Santa Maria de Palautordera | 41 | Vallès Oriental | 08460 |
| 1356 | 082763 | Tagamanent | 41 | Vallès Oriental | 08593 |
| 1357 | 082943 | Vallgorguina | 41 | Vallès Oriental | 08470 |
| 1358 | 082969 | Vallromanes | 41 | Vallès Oriental | 08188 |
| 1359 | 083067 | Vilalba Sasserra | 41 | Vallès Oriental | 08455 |
| 1360 | 089024 | Vilanova del Vallès | 41 | Vallès Oriental | 08410 |

*Figura 4.3 Dades del fitxer Corresp_cques_CAT_codpost.csv*

Després d'un primer anàlisi de les dades s'observa que la majoria dels estudiants provenen de la comarca del Barcelonès motiu pel qual es crea un nou fitxer corresponent a la divisió postal per districtes de Barcelona (*districtes.csv*). A més, s'afegeixen les poblacions de Santa Coloma de Gramanet, Badalona i L'Hospitalet del Llobregat per cobrir tota la regió postal del Barcelonès.

| 1 | DIS | CP | DIS | CP | DIS | CP | DIS | CP |
|---|---|---|---|---|---|---|---|---|
| 2 | Badalona | 8911 | Eixample Dreta | 8037 | L'Hospitalet | 8904 | Sant Gervasi | 8022 |
| 3 | Badalona | 8912 | Eixample Esquerra | 8015 | L'Hospitalet | 8905 | Sant Gervasi | 8034 |
| 4 | Badalona | 8913 | Eixample Esquerra | 8011 | L'Hospitalet | 8906 | Sant Marti | 8005 |
| 5 | Badalona | 8914 | Eixample Esquerra | 8036 | L'Hospitalet | 8907 | Sant Marti | 8018 |
| 6 | Badalona | 8915 | Gracia | 8012 | L'Hospitalet | 8908 | Sant Marti | 8019 |
| 7 | Badalona | 8916 | Gracia | 8023 | Nou Barris | 8016 | Sant Marti | 8020 |
| 8 | Ciutat Vella | 8001 | Gracia | 8024 | Nou Barris | 8031 | Sant Marti | 8026 |
| 9 | Ciutat Vella | 8002 | Horta-Guinardo | 8032 | Nou Barris | 8033 | Santa Coloma | 8921 |
| 10 | Ciutat Vella | 8003 | Horta-Guinardo | 8035 | Nou Barris | 8042 | Santa Coloma | 8922 |
| 11 | Eixample Dreta | 8007 | Horta-Guinardo | 8041 | Sant Adria | 8930 | Santa Coloma | 8923 |
| 12 | Eixample Dreta | 8008 | Les Corts | 8028 | Sant Andreu | 8027 | Santa Coloma | 8924 |
| 13 | Eixample Dreta | 8009 | Les Corts | 8029 | Sant Andreu | 8030 | Sants-Montjuic | 8004 |
| 14 | Eixample Dreta | 8010 | L'Hospitalet | 8901 | Sant Gervasi | 8006 | Sants-Montjuic | 8014 |
| 15 | Eixample Dreta | 8013 | L'Hospitalet | 8902 | Sant Gervasi | 8017 | Sants-Montjuic | 8038 |
| 16 | Eixample Dreta | 8025 | L'Hospitalet | 8903 | Sant Gervasi | 8021 | Sants-Montjuic | 8039 |

*Figura 4.4 Dades del fitxer districtes.csv*

## 4.1. Disseny de la interfície d'usuari de l'aplicació

La funció de l'aplicació és mostrar un conjunt d'estadístiques sobre el rendiment de l'alumnat durant la Fase Inicial del grau universitari donada una comarca de procedència i la nota d'accés a la universitat en una configuració visualment atractiva i interactiva.

Així doncs, en entrar a l'aplicació, l'usuari trobarà una primera interfície a mode d'índex en la que se li demanarà quines estadístiques o predicció vol consultar. Es disposen quatre tipus de consultes: la primera de caire purament descriptiva i la resta de caire probabilístic. Seguidament es mostra un petit esbós de l'índex:

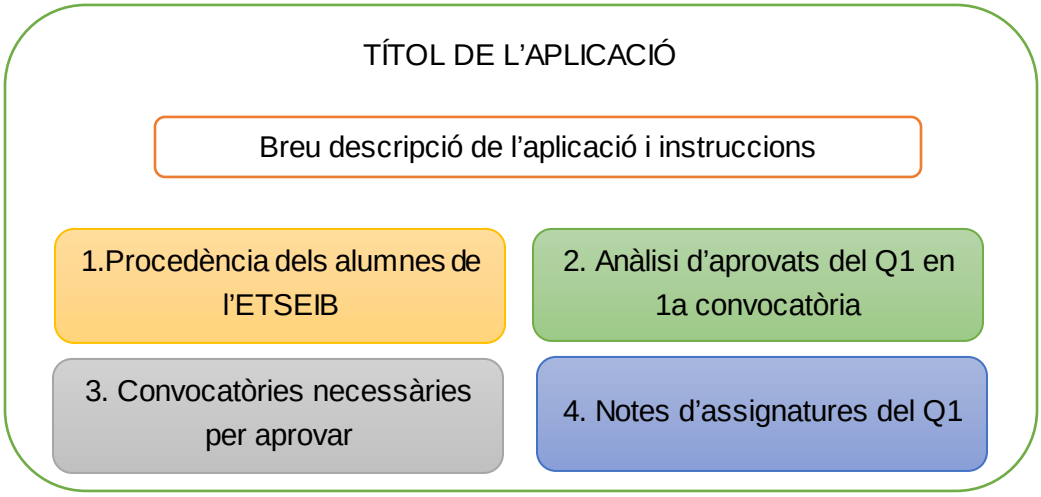



*Figura 4.5 Esbós de l'índex de l'aplicació web*

Les diferents opcions corresponen als diferents estudis que es realitzen en aquest projecte i es descriuen a continuació respectant la numeració de l'esbós:

1. Clicant la primera opció l'usuari serà redirigit a una pàgina on se li mostrarà els estudiants que han entrat al GETI segons l'any i la seva comarca o districte de Barcelona en un gràfic de barres apilades.

D'altra banda, clicant en una de les altres tres opcions l'usuari serà redirigit a una pàgina on trobarà un mapa de Catalunya interactiu.

2. Clicant sobre una comarca es mostra un gràfic amb la probabilitat d'aprovar cada assignatura del Q1 del GETI i en 1a convocatòria segons un rang de selectivitat. En el cas de la comarca del Barcelonès s'obre un desplegable per tal de seleccionar el districte.
3. Clicant sobre una de les comarques i donada una nota de selectivitat retorna gràfics de pastís per cada assignatura amb la probabilitat del nombre de convocatòries necessàries per aprovar les assignatures del Q1.

4. Similar al tercer però en aquest cas es mostra la probabilitat de la nota final en format categòric per a cada assignatura en 1a convocatòria.

## 4.2. La programació estadística i la programació web

### 4.2.1. La programació amb Python

Python és un llenguatge de programació interpretat i multiparadigma de llicència de codi obert amb la filosofia d'afavorir un codi llegible. Python està disponible per sistemes operatius Unix, Linux, Windows i MacOs i disposa d'una gran varietat de llibreries instal·lables les quals estan destinades a una funcionalitat concreta. En aquest projecte es treballa amb les llibreries Pandas i Flask. La primera és un software programat en python per la manipulació de i l'anàlisi de dades amb el qual es farà tota la mineria de dades, filtrat de dades i estudi estadístic. La segona és un entorn de treball minimalista escrit en python que permet crear aplicacions web com la desitjada en aquest projecte. (2)(3)

### 4.2.2. La programació web (HTML, CSS, JavaScript)

Per duu a terme la creació de l'aplicació web és necessari un servidor web, és a dir, un programa informàtic que processi l'aplicació creada en el servidor local i que en funció de les sol·licitacions de l'usuari respongui en conseqüència. D'aquesta manera l'aplicació es programa mitjançant un llenguatge de programació web.

Per a l'aplicació desitjada s'usaran tres llenguatges de programació web amb diferents funcionalitats (4):

HTML (Hypertext Markup Language): emprat per a descriure pagines web.

CSS (Cascading Style Sheets): utilitzat per a descriure l'estètica dels elements HTML.

JavaScript: codi de programació web i de HTML per a crear pagines interactives.

# 5. MINERIA DE DADES

Aquest projecte està basat en la mineria de dades, el camp de l'estadística i les ciències de la computació que refereix als processos que intenten descobrir patrons de grans volums de dades utilitzant tècniques estadístiques, sistema de base de dades i aprenentatge automàtic. A partir de l'aplicació d'aquestes tècniques es poden construir models predictius, de classificació o de segmentació. En aquest cas el model a construir que encaixa millor amb els objectius dels estudis a realitzar és el de classificació i la tècnica utilitzada serà els arbres de decisió. (5)

## 5.1. Els arbres de decisió

Un arbre de decisió és un model de predicció i classificació que permet representar de manera gràfica o analítica tots els successos que poden sorgir a partir d'una decisió presa en un cert moment. La idea bàsica és simple, es vol predir una resposta o classe $Y$ en funció d'unes variables entrada $X1, X2, ...Xp$. A cada node de l'arbre se li aplica un test corresponent a una de les entrades $Xi$. Depenent del resultat del test es pren una ramificació nodal determinada. De vegades s'arriba a un node final, on es duu a terme una predicció la qual té en compte tots els resultats dels test que s'han realitzat fins arribar a aquest node.

Els models de predicció com les regressions lineals o polinòmiques deriven en models globals on una sola formula predictiva abasta tot l'espai de dades. Tot i això, quan les dades interactuen de manera no lineal trobar un model adequat pot ser molt complex. Alternativament, els arbres de decisió aproximen aquestes regressions no lineals subdividint l'espai de dades en regions petites on les interaccions són més abordables. Les subdivisions es poden partir de manera recursiva formant una agrupació jeràrquica. Aleshores, s'obté un model amb dos parts, una partició recursiva i un model simple per cada partició. Es consideren dos tipus d'arbres de decisió els arbres de regressió i de classificació, aquest projecte treballa amb el segon.

### 5.1.1. Terminologia

- Node de decisió: node en el qual és necessari prendre una decisió. Està representat per un quadrat.
- Node de probabilitat: indica que en aquest punt succeeix un esdeveniment aleatori. Està representat per un cercle.
- Node final: indica el final d'una seqüencia recursiva, és a dir, la predicció final que conté la informació i probabilitats dels nodes de la seqüencia. Està representat per un triangle.

- Ramificació: mostra els diferents camins possibles en prendre una decisió o en ocorre un esdeveniment aleatori. Es representa amb una fletxa.
- Nivell: correspon a l'ordre creixent de la seqüencia recursiva, per exemple al node arrel li correspon el nivell 0, als nodes on arriben les seves ramificacions el nivell 1 i així successivament.

Per entendre el concepte i estructura es disposa del següent exemple gràfic basat en un projecte de reciclatge de metalls contaminants. Com s'observa l'arbre comença amb un node de decisió (node arrel) al qual li correspon el nivell 0 i la primera variable d'entrada $X1$, aquesta és binaria i val 1 si es prepara una proposta o 0 si no es prepara. Prendre una decisió significa directament passar al nivell 1, si es prepara la proposta s'arriba un node de probabilitat regit per una variable binaria $X2$ amb probabilitat del 50%, si no es prepara la proposta s'arriba a un node final amb la seva variable resposta corresponent $Y=0\$$. Si no s'arriba a un node final l'arbre segueix ramificant-se i incrementant els seus nivells fins arribar-hi. També es pot observar seguint els camins que porten als nodes finals que les respostes $Y$ conserven les informacions que han aportat totes les variables d'entrada $X$.

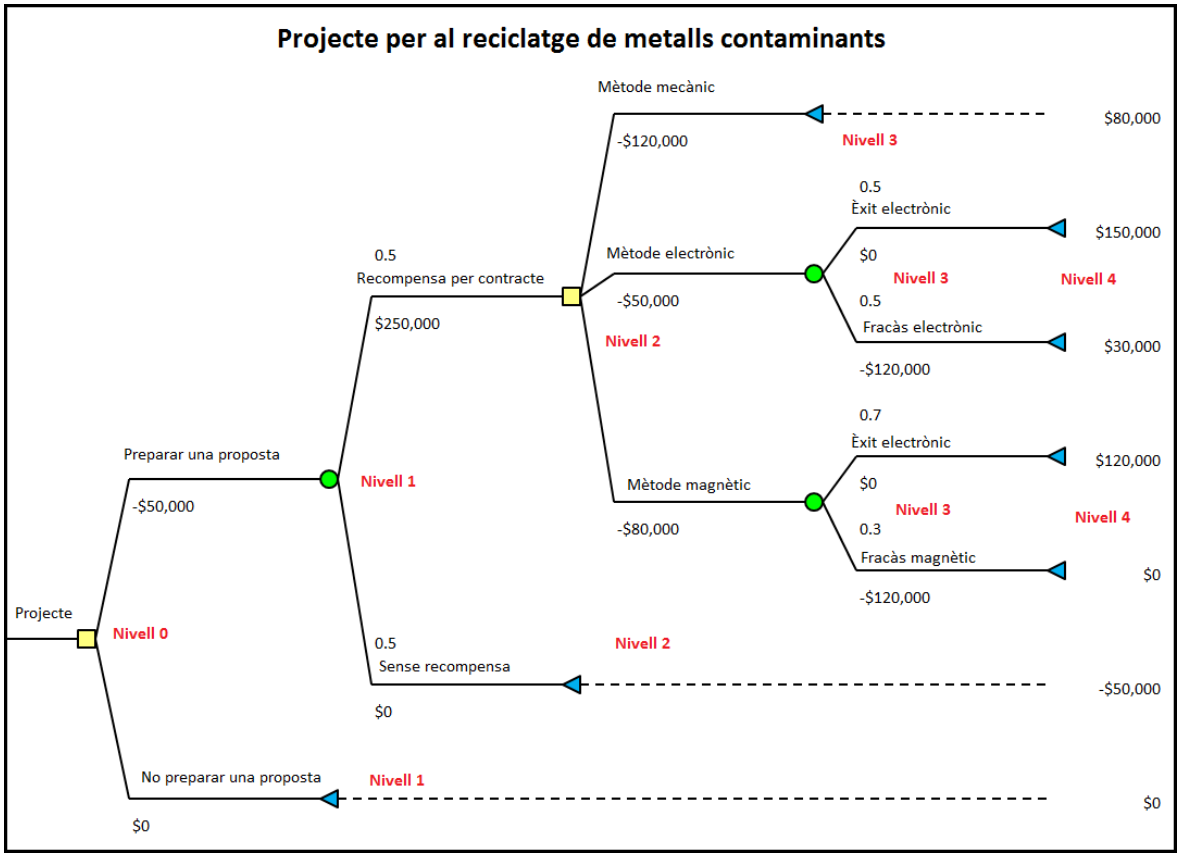



*Figura 5.1 Exemple d'un arbre de decisió.*

## 5.2. L'arbre de classificació

Els arbres de regressió i els de classificació treballen de la mateixa manera, l'únic que el primer prediu un valor numèric i el segon una classe o categoria. Les variables d'entrada poden ser numèriques o categòriques per ambdós arbres.

### 5.2.1. Mesura de la informació

Sent la variable $Y$ categòrica, es defineix $I[C; Y]$ com la informació que la subdivisió o clúster ens dóna sobre la variable $Y$ i $C$ és la variable que ens diu a quin node final correspon la informació. La intenció és maximitzar $I[C; Y]$ , per fer-ho es comença per un node anomenat arrel i es busca aquella ramificació que maximitza la resposta, aquest procés es repeteix recursivament pels nodes subseqüents.

Tot algoritme recursiu té el seu criteri d'aturada, en aquest cas es considera l'aturada quan les ramificacions d'un node porten a un altre node que conté menys de 5 dades o quan la nova subdivisió aporta poca nova informació. També existeix un cost de classificació errònia de qualsevol resposta $Y$, independentment de la seva classe real i sense importar a quina classe es va classificar erròniament, és uniforme i s'anomena entropia $H[Y]$. Es defineix també l'entropia condicional $H\,[\,Y\,[\,A = a\,]$. Aquest concepte d'entropia està extensament explicat en un article de David Feldman referenciat a la bibliografia. (6)

$$I[Y; A] = \sum_a \Pr(A = a)\ I[Y; A = a] \qquad \text{(Eq 5.1)}$$

$$I[Y; A = a] = H[Y] - H[Y|A = a] \qquad \text{(Eq 5.2)}$$

$I[Y;A=a]$ correspon a com la incertesa sobre $Y$ disminueix a mesura que $A = a$, $I[Y ; A]$ correspon a com la incertesa sobre $Y$ disminueix en promig a partir de saber el valor de $A$. $A$ acostuma a ser la resposta binaria a una pregunta sobre les variables $X$, $A = 1A(X)$. Per tant, al principi de l'algoritme, en el node arrel, s'escull una determinada $A$ a fi de maximitzar $I[Y ; A]$ la qual es calcula fent servir l'equació 5.1 que utilitza les freqüències relatives de les dades per obtenir les probabilitats.

Per cada node subsegüent es busquen preguntes que tenen en compte la informació adquirida en els passos anteriors. Per aquest node es computen les probabilitats i informacions utilitzant els casos del propi node en comptes dels del conjunt global de dades (cada subdivisió és un sub-problema del problema original). Matemàticament, significa que si s'arriba al node quan $A$=$a$ i $B$=$b$ es busca la pregunta $C$ que maximitza $I[Y; C \mid A = a, B = b]$, la qual correspon a la informació condicional en $A$=$a$, $B$=$b$. Algebraicament:

$$I[Y; C]|A = a, B = b] = H[Y|A = a, B = b] - H[Y|A = a, B = b, C] \quad \text{(Eq 5.3)}$$

Computacionalment, en lloc de mirar tots els casos del conjunt de dades, es pot mirar tan sols mirar aquells en què $A = a$ i $B = b$, i considerar-los com si corresponguessin a tots els casos. A més, s'observa que el primer terme del costat dret, $H\,[I \mid A = a, B = b]$, no depèn de la següent pregunta $C$. Així doncs, en comptes de maximitzar $I\,[I; C \mid A = a, B = b]$, es pot simplement minimitzar l'entropia $H\,[I \mid A = a, B = B, C]$.

### 5.2.2. Prediccions

Existeixen dos tipus de prediccions possibles pels arbres de classificació: la **predicció puntual**, una suposició única com una classe o categoria (per exemple és un aprovat o és un suspès), i la **predicció distribucional** la qual dona una probabilitat per cada classe o categoria i és la que es farà servir en aquest projecte. Per a la predicció distribucional, cada node final dona una distribució de les categories. Si el node correspon a una seqüencia de respostes $A = a, B = b, \ldots Q = q$, implica que portaran associades les corresponents probabilitats $Pr\,(Y = y \mid A = a, B = b, \ldots Q = q)$ per cada valor "$y$" possible de la resposta. Un exemple senzill seria un cas de 18 alumnes 6 dels quals han aprovat i la resta han suspès, llavors, el node final prediria un suspès amb una probabilitat de 2/3 i un aprovat amb una probabilitat de 1/3. Aquesta és l'estimació de **màxima probabilitat** de la veritable distribució de probabilitat, i s'escriu com $Pr\,(\cdot)$.

Es relativament fàcil errar en una predicció distribucional quan el número de classes és bastant més gran que la mida de la mostra. Aquest problema apareixerà durant el projecte ja que hi ha comarques de les quals provenen molt pocs alumnes, això farà que la predicció desafortunadament no sigui vàlida però es mostrarà igualment de manera informativa.

### 5.2.3. Models d'arbres de classificació de l'aplicació

Dels quatre estudis presentats en la introducció, només l'estudi de procedència geogràfica no fa servir l'arbre de classificació per processar les dades. Seguidament es mostra l'estructura gràfica de l'arbre de classificació de l'estudi corresponent a les notes categòriques del Q1 el qual segueix la mateixa terminologia explicada en el punt 5.1.1. El node arrel de l'arbre és un l'alumne predeterminat que consulta l'aplicació. Les decisions dels nodes de decisió seran preses per l'usuari quan interactuï amb l'aplicació. Els nivells són comuns en els tres estudis excepte el nivell 5 el qual pot ser la nota categòrica, el número de convocatòria o simplement un percentatge. Els nivells 4 (assignatures) i 5 (resposta) són les parts de l'arbre que es podran veure gràficament gràcies als diferents gràfics de Highcharts mentre que els altres nivells queden emmascarats per el propi sistema d'interaccions de l'aplicació amb l'usuari. Cal remarcar que a partir del nivell 3 les ramificacions de l'arbre segueixen una estructura simètrica.

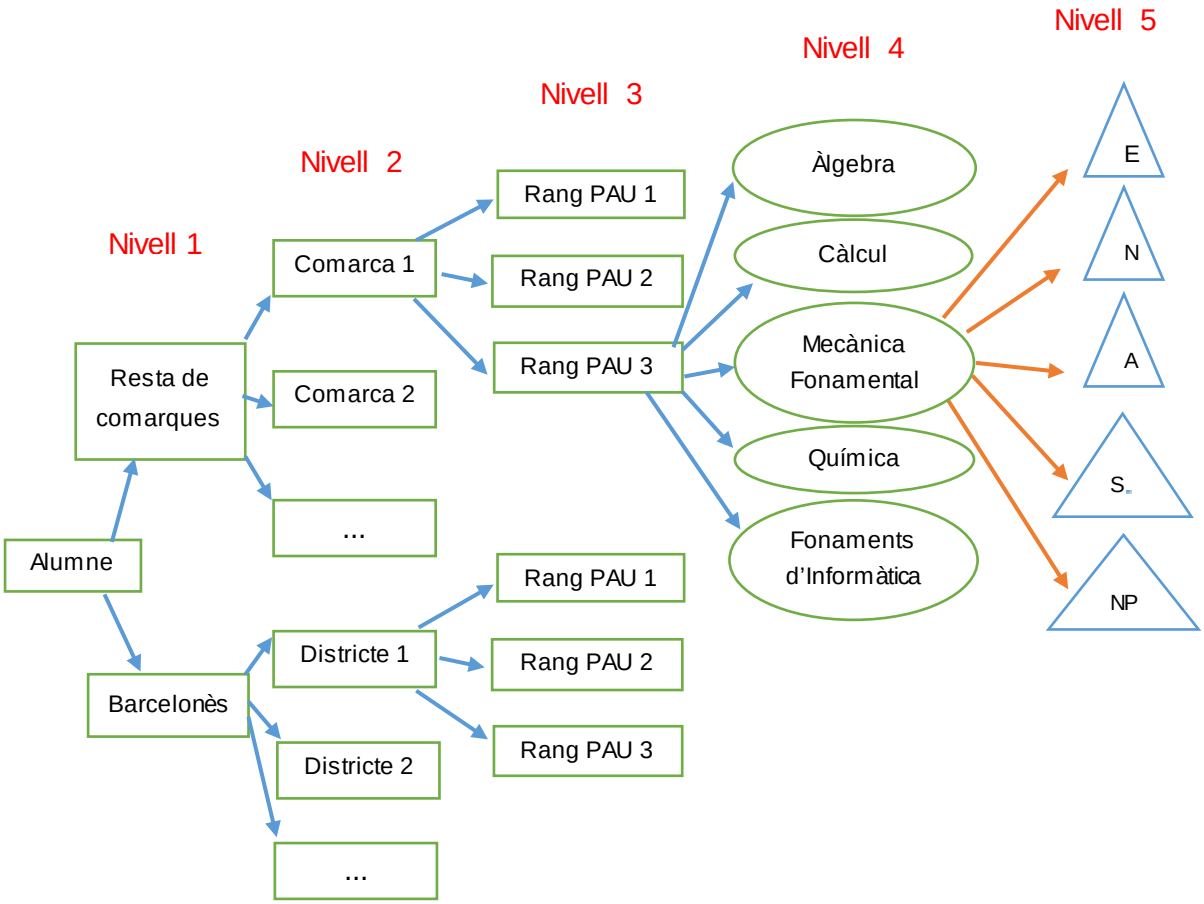



*Figura 5.2 Estructura de l'arbre de decisió que s'ha utilitzat en el processament de dades.*

Seguidament es mostra un gràfic de punts corresponent a l'arbre anterior on es poden veure les subdivisions que es creen en els nivells 3, 4 i 5 per tal de realitzar una predicció. Aquest no és l'estil de gràfic que es mostrarà a l'aplicació els quals seran més concrets i afins als interessos de consulta de l'usuari.

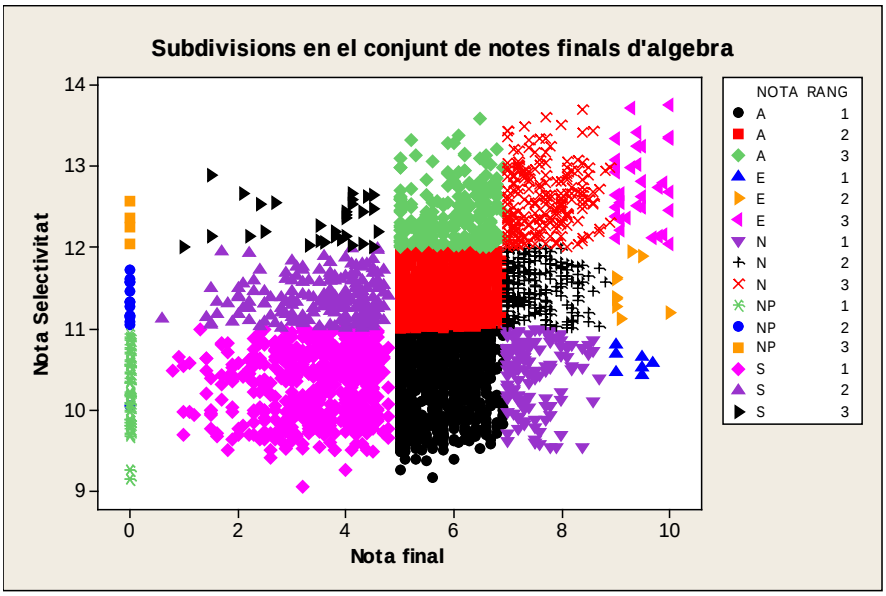



*Figura 5.3 Gràfic de les subdivisions dels nivells 3,4 i 5 pel Barcelonès..*

# 6. IMPLEMENTACIÓ I PROCESSAT DE LES DADES

Per a tal de dur a terme l'anàlisi estadístic amb python, és imprescindible tractar les dades, eliminant aquelles que no són d'interès, així com reordenant les dades d'interès en un format més òptim per a l'ús de l'arbre de decisió descrit en el punt anterior, evitar errors en l'execució dels algoritmes de python i condicionar-les per a la seva posterior representació a l'aplicació web. En total s'han utilitzat 18 funcions de python estructurades en diferents mòduls. També s'ha utilitzat un total de 18 fitxers *csv:* 4 amb dades inicials, 3 condicionats per treballar de manera òptima, 3 plantilles amb estructura d'arbre de classificació i 8 fitxers d'actualització continua dedicats a cada gràfic de Highcharts diferent.

La taula següent mostra les etapes seguides en la implementació i els diferents mòduls de python utilitzats juntament amb el número de funcions i fitxers *csv*:

| | Etapa | Mòdul de Python | Fitxers csv |
|---|---|---|---|
| **1** | Condicionament dels fitxers inicials | Condicionament inicial (2 funcions) | Inicials:<br>Corresp_cques_CAT_codpost.csv<br>DadespersIND16.csv<br>QualifINDfaseini16.csv<br>districtes.csv |
| **2** | Filtrat de dades d'interès | Filtrat de dades (3 funcions) | juntat.csv, indi+com.csv |
| **3** | Creació de noves variables d'interès | Creació de variables (4 funcions) | juntat.csv, indi+com.csv, convo.csv |
| | | Arbre de classificació (4 funcions) | Plantilles d'arbre:<br>com+asig.csv, com+convo.csv, nota_categorica.csv |
| **4** | Condicionament final en funció del gràfic de Highcharts | Condicionament final per a Highcharts (5 funcions) | Gràfic de barres: taula.csv<br>Barres apilades: cat2.csv, bcn2.csv<br>Sectors: algebra.csv, calcul.csv, mecanica.csv, quimica.csv, fonaments.csv |

*Taula 6.1 Taula del procediment seguit i els fitxers utilitzats per al processat de dades.*

## 6.1. Condicionament dels fitxers inicials

En primer lloc s'han de condicionar els fitxers inicials per poder treballar amb ells, a partir dels 4 fitxers de dades inicials es crearan dos fitxers base. El primer d'ells anomenat *juntat.csv* que és la fusió dels fitxers *QualifINDfaseini16.csv* i *DadespersIND16.csv* i conté les dades personals dels estudiants com les seves notes. El segon anomenat *indi+com.csv* és la fusió dels fitxers *DadespersIND16.csv, Corresp_cques_CAT_codpost.csv* i *districtes.csv* i conté les dades personals dels estudiants juntament amb el seu districte de Barcelona o comarca de procedència.

En el primer cas s'importa el paquet de Pandas com l'abreviació "pd" i s'utilitza la funció "merge" que permet fusionar els dos fitxers desitjats fent servir com a referència el codi de l'estudiant el qual contenen ambdós fitxers. La variable "df" que apareixerà moltes vegades en el codi és un marc de dades (DataFrame). El resultat són les notes dels alumnes durant la fase inicial però ara s'hi ha afegeixen les columnes de sexe, codi postal de l'escola, codi postal familiar, any i nota de selectivitat provinents del arxiu de dades personals.

```
def juntar():
    df1 = pd.read_csv(open('DadespersIND16.csv','r'))
    df2 = pd.read_csv(open('QualifINDfaseini16.csv','r'))
    data = pd.merge(df1,df2)
    data.to_csv('juntat.csv')
```

*Figura 6.1 Funció que fusiona els fitxers DadespersIND16.csv i QualifINDfaseini16.csv*

El segon fitxer s'obté recorrent els codis postal escolar del fitxer de dades individuals, si aquest codi correspon a un codi postal de comarca catalana s'afegeix en una nova columna el nom de la comarca d'on prové l'estudiant. El mateix procediment es repeteix pels estudiants del Barcelonès fent servir el fitxer *districtes.csv*. La raó perquè s'escull el codi postal escolar i no el familiar s'explica en el punt següent.

```
def comarca():
    df = pd.read_csv(open('DadespersIND16.csv','r'))
    dis = pd.read_csv(open('Corresp_cques_CAT_codpost.csv','r'))
    df['COM'] = pd.Series(' ', index=df.index)
    dis.set_index('CPE',inplace = True)
    i = 0
    for cpe in list(df.CPE):
        if cpe in dis.index:
            comarca = dis.loc[cpe].COM
            df.set_value(i, 'COM', comarca)
            df.set_value(i, 'DIS', dis.loc[cpe].DIS)
        i = 1 + i

    df.to_csv('indi+com.csv')
```

*Figura 6.2 Funció que afegeix la columna comarca al fitxer de dades individuals.*

## 6.2. Filtrat de les dades d'interès

### 6.2.1. Dades d'interès

L'aplicació web permetrà consultar 4 tipus d'estudis diferents sobre el rendiment dels alumnes: la procedència anual dels nous estudiants, l'èxit en superar les assignatures del Q1 la primera vegada que les cursen i el nombre de convocatòries que necessiten per superar cada assignatura. Per cadascun d'aquest estudis són necessàries unes dades que han d'estar ben ordenades i ben definides per no cometre errors en l'execució dels algorismes. Seguidament es mostra una llista de les consideracions més importants que s'han tingut a l'hora de filtrar les dades:

- Codi postal escolar (CPE): es fa servir aquest codi i es prescindeix del familiar a l'hora de classificar els alumnes ja que ens interessa on han fet els seus estudis de Batxillerat, aquest codi ens dóna més informació com per exemple si han de venir a viure a Barcelona perquè provenen d'una comarca llunyana.
- Assignatura cursada per primera vegada: es considera quan correspon al Q1 i l'any que l'alumne es presenta a les PAU i l'any que la cursa coincideixen. Tot i això les dades d'aquests anys no tenen perquè coincidir, això ens porta a la següent dada d'interès.
- Any d'accés a l'Etseib: ve definida per l'any que l'alumne cursa la primera assignatura.
- Comarca o districte: només es considera la regió de la comunitat autònoma de Catalunya per a la divisió geogràfica, els codis postal que no hi pertanyen queden exclosos de l'estudi.
- Nota de les PAU: només es consideren alumnes amb nota entre 9 i 14, queden exclosos aquells que hagin entrat a l'ETSEIB mitjançant altres vies.
- GETI: només es consideren alumnes que cursen el GETI.

### 6.2.2. L'algorisme del filtrat de dades

El primer pas consisteix en depurar els fitxers base, és a dir, excloure aquelles dades que es consideren incompletes per l'estudi i eliminar duplicats. L'algoritme reemplaça cel·les buides per espais per posteriorment eliminar-les i elimina els duplicats. Amés, l'algoritme filtra aquells alumnes que provenen de Catalunya dels que no ja que en la creació dels fitxers base han quedat cel·les buides pels que tenien un codi postal de fora de Catalunya.

```
def depurar(df):
    df = df.drop.duplicates()
    df = df.replace(np.nan,' ', regex=True)
    for column in df:
        a=0
        for valor in df[column]:
            if valor == ' ':
                df = df.drop(a)
            a=a+1
    df = df.reset_index(drop = True)
```

*Figura 6.3 Funció per a l'eliminació de duplicats i files amb dades incompletes.*

El següent pas seria filtrar per alumnes que cursen el GETI i amb nota de les PAU entre 9 i 14 però els fitxers originals ja estaven filtrats. Tot i això, es faria amb les següents comandes de python: `df = df[(df.CODI_PROGRAMA == 752) & (9 <= df.SELE <= 14)]`

Seguidament, donat que l'any que l'alumne fa la selectivitat no és d'interès, es decideix redefinir la columna "ANY" com a l'any que l'alumne cursa la primera assignatura del grau. El programa recorre cada alumne retornant el seu DataFrame particular, llavors, es busca el menor any en el què ha cursat una assignatura (curs) i es compara amb el valor de la columna "ANY", si coincideixen no fa res i si difereix la columna "ANY" obté el valor de la variable (curs).

```
def arreglaany():
    df = pd.read_csv(open('juntat.csv','r'))
    df.set_index('CODEX', inplace=True)
    for i in list(df.index):
        dflocal = df.loc[i]
        if type(dflocal.CURS) is np.int64:
            curs = dflocal.CURS
        else:
            curs = min(dflocal.CURS)
        if type(dflocal.ANY) is np.int64:
            if curs != dflocal.ANY:
                df.set_value(i,'ANY',curs)
        else:
            if curs not in  dflocal.ANY:
                df.set_value(i,'ANY',curs)
    df.to_csv('juntat.csv')
```

Figura 6.4 Funció que arregla l'any que un alumne entra a *l'ETSEIB.*

## 6.3. Creació de noves variables d'interès

### 6.3.1. Variables d'interès

Com s'ha explicat a la introducció es volen fer 4 estudis diferents però inicialment no es disposa de la informació suficient per treballar-hi, per això és necessari crear noves variables a partir de variables preexistents. Les noves variables a crear són les següents:

- Nota de la 1a convocatòria en assignatures del Q1.
- Abandona: variable lògica que és certa quan l'alumne no supera mai alguna de les assignatures del Q1 i falsa en cas contrari.
- Número d'alumnes per comarca o districte de Barcelona.
- Rang de selectivitat: és part de l'arbre de decisió i agrupa els alumnes segons la seva puntuació a les PAU.
- Número de convocatòria: és el nombre de vegades que un alumne cursa una assignatura, també serà part de l'arbre de decisió.

### 6.3.2. Algoritmes de creació

a) *Rang de selectivitat*

Per definir la variable rang de selectivitat s'ha utilitzat el programa estadístic Minitab per veure gràficament la distribució de les notes de selectivitat de tots els alumnes. Es vol dividir el conjunt de les notes en 3 rangs per tal de tenir una certa varietat d'opcions però no en excés ja que si la classificació és massa ramificada podrien obtenir-se resultats de poca fiabilitat degut a la falta d'una mostra suficient de dades que els avalin.

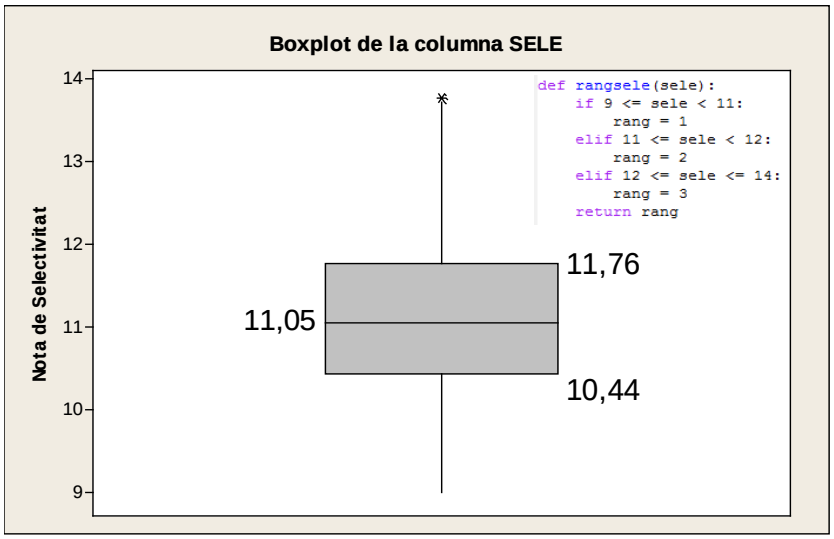



*Figura 6.5 Boxplot del conjunt de notes de selectivitat i funció classificadora.*

Com s'observa en el gràfic aproximadament el 50% de les notes està per sota d'11 i és decideix que sigui el primer rang, definit per alumnes amb nota suficient per accedir al grau aquell any, el segon serà el aproximadament el 30% que compren les notes entre 11 i 12, definit per alumnes amb una nota notable, i finalment el 20% que correspon a notes superiors a 12, definit per aquells alumnes excel·lents abans de cursar el grau.

*b) Convocatòria*

Definir la variable convocatòria és bastant complex ja que hi ha alumnes que no han cursat alguna assignatura i altres que han abandonat. L'algoritme utilitzat fa servir els dos fitxers base i emmagatzemarà al final les dades de les convocatòries en el fitxer *convo.csv* on es crea una columna buida per assignatura. Per cada fitxer es defineix el codi de l'estudiant com a índex i comença un bucle que recorre tots els estudiants. Per cada alumne s'analitza cada assignatura amb un altre bucle que obté com a variable un DataFrame amb el Q1 d'aquell alumne, si l'alumne ha cursat l'assignatura analitzada se substitueix el DataFrame pel de l'assignatura en concret i s'obté una variable llista provinent de la columna "APR" (superat 'S' o no 'N') que pot contenir alguna 'N' però només una 'S', si conté una 'S' la cel·la buida corresponent a l'assignatura i l'alumne obté el valor de la longitud de la llista, és a dir, les vegades cursades fins superar l'assignatura, si no conté 'S' vol dir que mai ha superat l'assignatura i per tant ha abandonat el grau, la cel·la obté el valor 100 per indicar-ho. Si l'alumne no cursa l'assignatura la cel·la obté el valor 0.

```
def convocatoria():
    df = pd.read_csv(open('juntat.csv','r'))
    df.set_index('CODEX', inplace=True)
    df2 = pd.read_csv(open('indi+com.csv','r'))
    df2.set_index('CODEX', inplace=True)
    #el fitxer juntat.csv conté el nom de les assignatures i no els codis
    q1 = ['Algebra','Calcul','Mecanica','Quimica','Fonaments Informatica']
    for assignatura in q1:
        df2[assignatura] = pd.Series(' ', index=df.index)

    for alumne in list(df2.index):
        for assignatura in q1:
            q1alumne = df.loc[alumne] #DataFrame
            if assignatura in list(q1alumne['CODI_UPC_UD']):
                q1alumne = q1alumne[q1alumne['CODI_UPC_UD'] == assignatura]
                convocatories = list(q1alumne.APR)
                if not 'S' in convocatories:
                    df2.set_value(alumne, assignatura, 100)
                else:
                    df2.set_value(alumne, assignatura, len(convocatories))
            else:
                df2.set_value(alumne, assignatura, 0)

    df2.to_csv('convo.csv')
```

*Figura 6.6 Funció per crear la variable convocatoria.*

Una vegada creades les noves columnes es busca el valor màxim de convocatòries al que un alumne s'ha presentat, en aquest cas és 6, per tant, la variable convocatòria prendrà valors de 0 a 6 i 100 en el cas que hagi abandonat.

c) *Abandona*
Es crea una nova columna anomenada "Abandona" en el fitxer *convo.csv* que val True si l'alumne té un 100 en alguna columna de les assignatures i False en cas contrari.

d) *Nota de la 1a convocatòria en assignatures del Q1.*
Aquesta variable es crea d'una manera molt semblant a la variable convocatòria però s'emmagatzema en el propi fitxer base *indi+com.csv.* En aquest cas s'itera la columna "CODI_UPC_UD" que conté els codis d'assignatura del document *juntat.csv* i es mira si pertanyen al Q1 i si les cursen per primera vegada o no comparant any i curs, llavors, omple la cel·la buida amb la nota final de l'alumne en l'assignatura corresponent.

```
def notalaconvocatoria():
    df = pd.read_csv(open('indi+com.csv','r'))
    df2 = pd.read_csv(open('juntat.csv','r'))
    df.set_index('CODEX', inplace=True)
    q1 = ['Algebra','Calcul','Mecanica','Quimica','Fonaments Informatica']
    for assignatura in q1:
        df[assignatura] = pd.Series(' ', index=df.index)

    i = 0 #número de la fila
    df2 = df2[df2.Q == 1]
    df2 = df2.reset_index(drop = True)
    for assignatura in df2.CODI_UPC_UD:
        alumne = df2.CODEX[i]
        if alumne in df.index:
            if (df2.ANY[i] == df2.CURS[i]):
                if assignatura in q1:
                    df.set_value(alumne, assignatura, df2.NF[i])
        i = i + 1
    df.to_csv('indi+com.csv')
```

*Figura 6.7 Funció que guarda la nota final en primera convocatòria d'assignatures del Q1*

### 6.3.3. L'arbre de classificació

Per dur a terme la classificació explicada en el punt 5 és necessari organitzar les variables de classificació per tal de crear un document amb estructura d'arbre que emmagatzemi les dades que després seran utilitzades per a la representació de gràfics a la pàgina web.

a) *Estructura pel càlcul del percentatge d'aprovats en primera convocatòria*
Es crea un nou fitxer anomenat *com+asig.csv* on s'elabora l'estructura d'arbre composta per les comarques i el rang de selectivitat, a més, s'afegeixen columnes comptador, les que tenen com a nom la inicial de l'assignatura s'ompliran amb el nombre de persones que aproven l'assignatura i les restants amb el total d'alumnes que la cursen. El document té aquesta estructura per facilitar després el càlcul del percentatge desitjat.

| 1 | COM | RANG | A | C | M | Q | F | TA | TC | TM | TQ | TF |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 2 | Alt Camp | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 3 | Alt Camp | 2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 4 | Alt Camp | 3 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 5 | Alt Emporda | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 6 | Alt Emporda | 2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 7 | Alt Emporda | 3 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 8 | Alt Penedes | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 9 | Alt Penedes | 2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 10 | Alt Penedes | 3 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

*Figura 6.8 Fitxer com+asig.csv*

L'algoritme que omple les columnes itera les files (alumne) del fitxer *indi+com.csv* en forma de tuples i per cada assignatura obté la nota final, si aquesta nota existeix i és igual o superior a 5 suma un aprovat i una persona a la columna corresponent del fitxer arbre, si és inferior a 5 suma només una persona i si no existeix no fa res.

```
def percentatge_aprovats():
    df = pd.read_csv(open('indi+com.csv','r'))
    dc = pd.read_csv(open('com+asig.csv','r'))
    dc.set_index(['COM','RANG'], inplace=True)

    l = ['A','C','M','Q','F'] #columnes amb el numero d'aprovats
    t = ['TA','TC','TM','TQ','TF'] #columnes del total d'alumnes
    ll = [13,14,15,16,17] #posició de les columnes assignatura en les tuples

    for row in df.itertuples():
        com = row.COM
        rang = rangsele(row.SELE)
        c = 0  #variable per recorre les llistes l i t
        for assignatura in ll:
            nota = row[assignatura]
            aprovats = dc.loc[(com,rang)][l[c]]
            total = dc.loc[(com,rang)][t[c]]
            if nota == ' ':
                pass
            else:
                dc.set_value((com,rang), l[c], (total + 1))
                if float(nota) >= 5:
                    dc.set_value((com,rang), t[c], (aprovats + 1))
            c = c+1
            if c > 4:
                c = 0
    dc.to_csv('com+asig.csv')
```

*Figura 6.9 Funció que omple les columnes del fitxer com+asig.csv*

*b) Estructura per calcular les convocatòries necessàries per aprovar*

Es crea un nou fitxer anomenat *com+convo.csv* que és igual que el del punt anterior però se li afegeix la variable convocatòria com a nova ramificació de l'arbre. La variable convocatòria com s'ha vist abans va de 0 a 6, en aquest cas no té sentit considerar no haver fet cap convocatòria però sí el cas que l'alumne mai hagi aprovat aquella assignatura, per tant, s'afegeix la convocatòria 100 que simbolitza haver-la abandonat. Les columnes funcionen com a comptadors d'alumnes.

| | COM | RANG | CONVO | A | C | M | Q | F |
|---|---|---|---|---|---|---|---|---|
| 2 | Alt Camp | 1 | 1 | 0 | 0 | 0 | 0 | 0 |
| 3 | Alt Camp | 1 | 2 | 0 | 0 | 0 | 0 | 0 |
| 4 | Alt Camp | 1 | 3 | 0 | 0 | 0 | 0 | 0 |
| 5 | Alt Camp | 1 | 4 | 0 | 0 | 0 | 0 | 0 |
| 6 | Alt Camp | 1 | 5 | 0 | 0 | 0 | 0 | 0 |
| 7 | Alt Camp | 1 | 6 | 0 | 0 | 0 | 0 | 0 |
| 8 | Alt Camp | 1 | 100 | 0 | 0 | 0 | 0 | 0 |
| 9 | Alt Camp | 2 | 1 | 0 | 0 | 0 | 0 | 0 |
| 10 | Alt Camp | 2 | 2 | 0 | 0 | 0 | 0 | 0 |
| 11 | Alt Camp | 2 | 3 | 0 | 0 | 0 | 0 | 0 |
| 12 | Alt Camp | 2 | 4 | 0 | 0 | 0 | 0 | 0 |
| 13 | Alt Camp | 2 | 5 | 0 | 0 | 0 | 0 | 0 |
| 14 | Alt Camp | 2 | 6 | 0 | 0 | 0 | 0 | 0 |
| 15 | Alt Camp | 2 | 100 | 0 | 0 | 0 | 0 | 0 |

*Figura 6.10 Fitxer com+convo.csv*

L'algoritme és molt semblant al de l'apartat a), l'únic canvi és que quan troba un alumne amb cap convocatòria en una assignatura no fa res.

```
def convocatories_aprobar():
    df = pd.read_csv(open('convo.csv','r'))
    dc = pd.read_csv(open('com+convo.csv','r'))
    dc.set_index(['COM','RANG','CONVO'], inplace=True)
    l = ['A','C','M','Q','F'] #columnes amb el numero d'aprovats
    ll = [13,14,15,16,17] #posicio de les columnes assignatura en les tuples

    for row in df.itertuples():
        com = row.COM
        rang = rangsele(row.SELE)
        c = 0 #variable per recorre les llistes l i t
        for assignatura in ll:
            convocatoria = row[assignatura]
            if convocatoria == 0:
                pass
            else:
                convocats = dc.loc[(com,rang,convocatoria)][l[c]]
                dc.set_value((com,rang,convocatoria), l[c], (convocats + 1))
                c = c+1
                if c > 4:
                    c = 0
    dc.to_csv('com+convo.csv')
```

*Figura 6.11 Funció que omple les columnes de com+convo.csv*

Posteriorment es fa servir un altre algoritme que calcula el percentatge de cada convocatòria per un rang i una comarca determinada. L'algoritme crea noves columnes pels percentatges de cada assignatura. Després, itera les columnes del marc de dades i obté el percentatge dividint el valor de les cel·les indexades per comarca, rang i convocatòria per la suma total dels valors de les cel·les indexades per comarca i rang. Si la suma total és 0 la divisió donaria infinit, per tant, es deixen les cel·les percentatge en blanc per una posterior eliminació de les files.

```
def percentatge_convocatories():
    dc = pd.read_csv(open('com+convo.csv','r'))
    dc.set_index(['COM','RANG','CONVO'], inplace=True)
    l = ['A','C','M','Q','F'] #columnes comptador d'alumnes segons convocatoria
    for assig in l:
        dc['P'+ assig] = pd.Series('', index=dc.index)

    for row in dc.iterrows():
        com, rang, convo = (row[0][0], row[0][1], row[0][2])
        for assig in l:
            total = sum(dc.loc[(com,rang)][assig])
            if total == 0:
                pass
            else:
                percentatge = "{0:.2f}%".format(float(row[1][assig])/total*100.0)
                dc.set_value((com,rang,convo), 'P'+assig, percentatge)
    dc.to_csv('com+convo.csv')
```

*Figura 6.12 Funció que calcula la contribució de cada convocatòria.*

c) *Estructura per calcular la nota categòrica segons assignatura*

L'estructura i l'algoritme emprats són gairebé idèntics als utilitzats pel càlcul de convocatòries, l'únic canvi és la columna convocatòria que es converteix en la columna nota categòrica i conté el rang categòric amb les opcions NP (No presentat), IN (Insuficient), S (Suficient), N (notable) i E (Excel·lent). Aquest nou arbre es guarda en el fitxer *nota_categorica.csv*.

## 6.4. Condicionament dels fitxers finals per a la representació dels resultats a l'aplicació web

Cadascun dels estudis proposats en la introducció fa servir un tipus de gràfic diferent per representar les dades estadístiques. Com s'explicarà més endavant, s'utilitzarà la biblioteca de gràfics interactius Highcharts escrita en JavaScript. Cada gràfic de Highcharts té la peculiaritat de demanar les dades organitzades d'una certa manera, per això cal condicionar els fitxers *csv*.

### 6.4.1. Gràfic de barres apilades

El gràfic de barres apilades s'utilitzarà per representar la procedència dels estudiants per anys. El gràfic obté les dades de dos fitxers *csv*, un per comarques i l'altre per districtes de Barcelona, on els anys són les files i les comarques les columnes, anomenats cat2.csv i bcn2.csv respectivament. L'algoritme crea dos marcs de dades on les columnes són els anys i les comarques o districtes l'índex. S'itera cada fila del document base *indi+com.csv* i es van omplint els marcs de dades que actuen com a comptadors. Finalment, s'elimina l'índex, es transposen les matrius de dades i es converteixen en documents *csv*.

```
def procedencia_alumnes():
    df = pd.read_csv(open('indi+com.csv','r'))
    cat = pd.DataFrame(0, index = comarques_cat, columns = anys)
    bcn = pd.DataFrame(0, index = districtes_bcn, columns = anys)

    for row in df.itertuples():
        com = row.COM
        Any = str(row.ANY)
        if com in districtes_bcn:
            bcn.set_value(com, Any, bcn.loc[com][Any]+1)
        else:
            cat.set_value(com, Any, cat.loc[com][Any]+1)

    cat.set_index('2010',inplace = True)
    bcn.set_index('2010',inplace = True)
    (cat.transpose()).to_csv('cat2.csv')
    (bcn.transpose()).to_csv('bcn2.csv')
```

| | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 2010 | 0 | 2 | 4 | 1 | 0 | 4 | 6 | 9 | 3 | 2 | 28 | 2 | 0 | 2 | 0 | 4 |
| 2 | 2011 | 1 | 6 | 7 | 2 | 0 | 4 | 1 | 7 | 4 | 1 | 16 | 1 | 0 | 0 | 1 | 3 |
| 3 | 2012 | 0 | 2 | 7 | 0 | 0 | 9 | 5 | 5 | 3 | 2 | 36 | 0 | 2 | 1 | 0 | 4 |
| 4 | 2013 | 3 | 4 | 10 | 2 | 1 | 3 | 4 | 6 | 4 | 3 | 18 | 2 | 1 | 0 | 1 | 6 |
| 5 | 2014 | 1 | 1 | 7 | 0 | 0 | 2 | 4 | 7 | 2 | 2 | 19 | 0 | 2 | 0 | 0 | 5 |
| 6 | 2015 | 2 | 3 | 6 | 0 | 1 | 6 | 2 | 5 | 1 | 0 | 34 | 2 | 0 | 0 | 0 | 4 |

*bcn2.csv*

*Figura 6.13 Funció que prepara les dades pel gràfic de barres apliades.*

### 6.4.2. Gràfic de barres

El gràfic de barres s'utilitzarà per representar el percentatge d'aprovats en les assignatures del Q1 en primera convocatòria segons nota de selectivitat i comarca. El gràfic obté les dades d'un fitxer anomenat *taula.csv* que s'actualitzarà cada vegada que l'usuari decideixi canviar de comarca a visualitzar. L'algorisme obté una comarca que definirà l'usuari i filtra el fitxer de dades *com+asig.csv* per aquella comarca. Després, recorre cada assignatura i rang per conformar llistes de percentatges a partir de les columnes comptador del fitxer. Finalment, aquestes llistes es disposen de la manera desitjada per conformar un nou marc de dades.

```
def obte_comarca(comarca):
    df = pd.read_csv(open('app/static/com+asig.csv','r'))
    df = df[df.COM == comarca]
    df.set_index('RANG',inplace=True)
    ql = ['Algebra','Calcul','Mecanica','Quimica','Fonaments Informatica']
    d = [] #llista necessaria per
    for assig  in ql:
        percentatge = []
        for rang in df.index:
            if df.loc[rang]['T'+assig[0]] == 0:
                percentatge.append(0)
            else:
                percentatge.append("{0:.2f}".format(df.loc[rang][assig[0]]/df.loc[rang]['T'+assig[0]]*100))
        d = d + [(assig,percentatge)]

    taula = pd.DataFrame.from_items(d, orient='index', columns = list(df.index))
    taula.to_csv('app/static/taula.csv')
```

| 1 | | [9;11) | [11;12) | [12;14] |
|---|---|---|---|---|
| 2 | Algebra | 50.00 | 100.00 | 100.00 |
| 3 | Calcul | 50.00 | 100.00 | 0.00 |
| 4 | Mecanica | 0.00 | 100.00 | 100.00 |
| 5 | Quimica | 100.00 | 100.00 | 100.00 |
| 6 | Fonaments Informatica | 0.00 | 100.00 | 100.00 |

*taula.csv*

*Figura 6.14 Funció que actualitza les dades per al gràfic de barres.*

### 6.4.3. Gràfic de sectors

Per cada assignatura del Q1 s'utilitzarà un gràfic de sectors per representar la distribució de convocatòries necessàries per aprovar segons comarca i nota de selectivitat. Cada gràfic obté les dades d'un fitxer *csv* amb el nom de l'assignatura ( *algebra.csv* ) que s'actualitzarà en funció de la decisió de l'usuari. L'algorisme obté una comarca i nota de selectivitat definides per l'usuari que serveixen per filtrar el fitxer *com+convo.csv*. Després, transforma el marc de dades filtrat en una llista on cada posició correspon al marc de dades d'una assignatura. La funció neteja elimina aquelles convocatòries que contribueixen un 0% en la distribució total i per tant no es mostren en el gràfic.

```
def obte_convocatoria(comarca,sele):
    df = pd.read_csv(open('app/static/quesitos/com+convo.csv','r'))
    df = df[(df.COM == comarca) & (df.RANG == rangsele(sele))]
    df = df[['CONVO','PA','PC','PM','PQ','PF']]
    df.set_index('CONVO', inplace = True)
    assig = list(df.transpose().iterrows())
    if len(asig[0][1]) == 0:
        return False  #informa al JavaScript de l'inici en blanc
    else:
        neteja(assig[0][1]).to_csv('app/static/quesitos/algebra.csv')
        neteja(assig[1][1]).to_csv('app/static/quesitos/calcul.csv')
        neteja(assig[2][1]).to_csv('app/static/quesitos/mecanica.csv')
        neteja(assig[3][1]).to_csv('app/static/quesitos/quimica.csv')
        neteja(assig[4][1]).to_csv('app/static/quesitos/fonaments.csv')
        return True
```

| 1 | 1a convocatoria | 63.16% |
|---|---|---|
| 2 | 2a convocatoria | 26.32% |
| 3 | 3a convocatoria | 10.53% |

*algebra.csv*

*Figura 6.15 Funció que actualitza les dades per al gràfic de sectors.*

# 7. L'APLICACIÓ WEB

## 7.1. Flask

Flask és un marc de treball minimalista escrit en Python que es farà servir per crear l'aplicació web. Les funcions de Flask que més s'explotaran són aquelles que processen accions que duu a terme l'usuari interactuant amb l'aplicació i retornen una resposta. L'esquema de l'entorn de treball que es crearà és el següent:

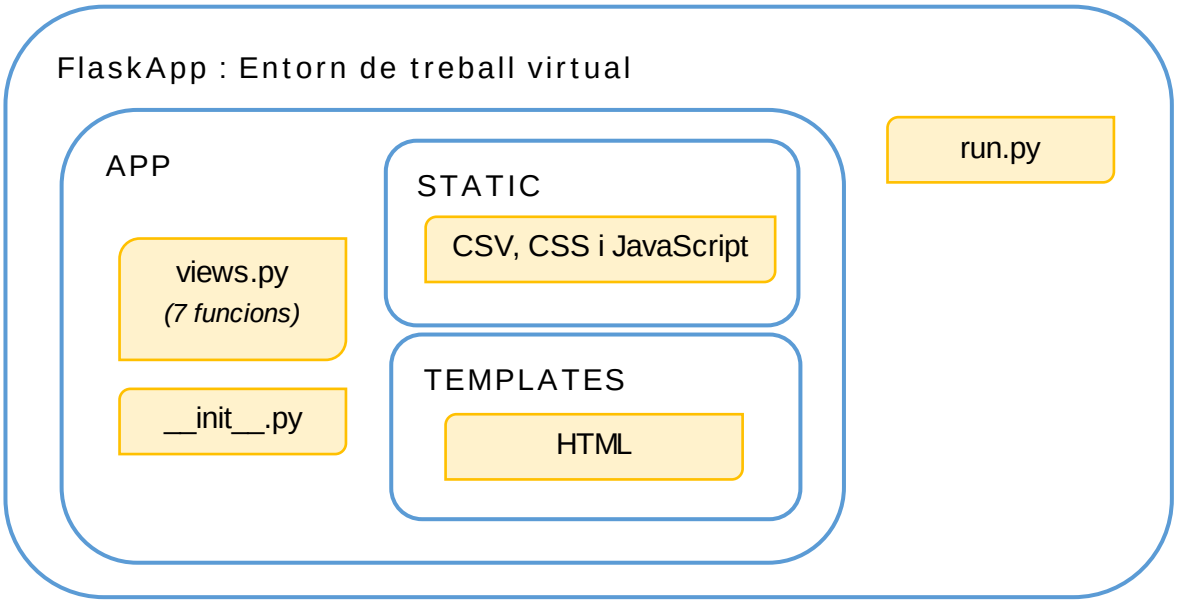



*Figura 7.1 Esquema de l'entorn de treball virtual creat amb Flask.*

Els requadres de color blau simbolitzen carpetes mentre que la resta són fitxers. La carpeta FlaskApp funciona com un entorn virtual de treball on es troba l'aplicació web i els seus components, mentre que el fitxer *run.py* serveix per iniciar el servidor web de desenvolupament amb l'aplicació. La carpeta App conté dos fitxers: *__init__.py* és un script que inicialitza l'objecte aplicació i importa el fitxer *views.py* que conté pròpiament el programa que crea, assigna i governa les diferents adreces URL de l'aplicació. A més App té dos subcarpetes: Static emmagatzema fitxers estàtics com CSV, CSS i JavaScript utilitzats per l'aplicació mentre que Templates conté les diferents pàgines HTML corresponents a les adreces URL assignades i governades per *views.py*. En total s'han fet servir 7 funcions de pyhton, 5 per controlar els diferents Templates i 2 més de correcció ortogràfica ja que el format *csv* no accepta accents ni apòstrofs d'informació provinent de l'aplicació.

## 7.2. La interfície d'usuari

Els llenguatges HTML i CSS permeten descriure pagines web i la seva estètica. El codi escrit en aquests llenguatges és interpretat pels navegadors web (Internet Explorer, Google Chrome, Firefox, Safari, etc.). L'aplicació parteix d'una pàgina web principal que funciona com a índex i a partir d'aquesta es pot accedir a les altres pàgines de l'aplicació que contenen els estudis realitzats. En total l'aplicació ha estat creada a partir de 5 fitxers HTML i 3 fitxers CSS. La següent imatge mostra el que el navegador web ensenya per pantalla en referència al codi de programació de l'aplicació, en aquest cas el codi correspon a l'índex de l'aplicació.

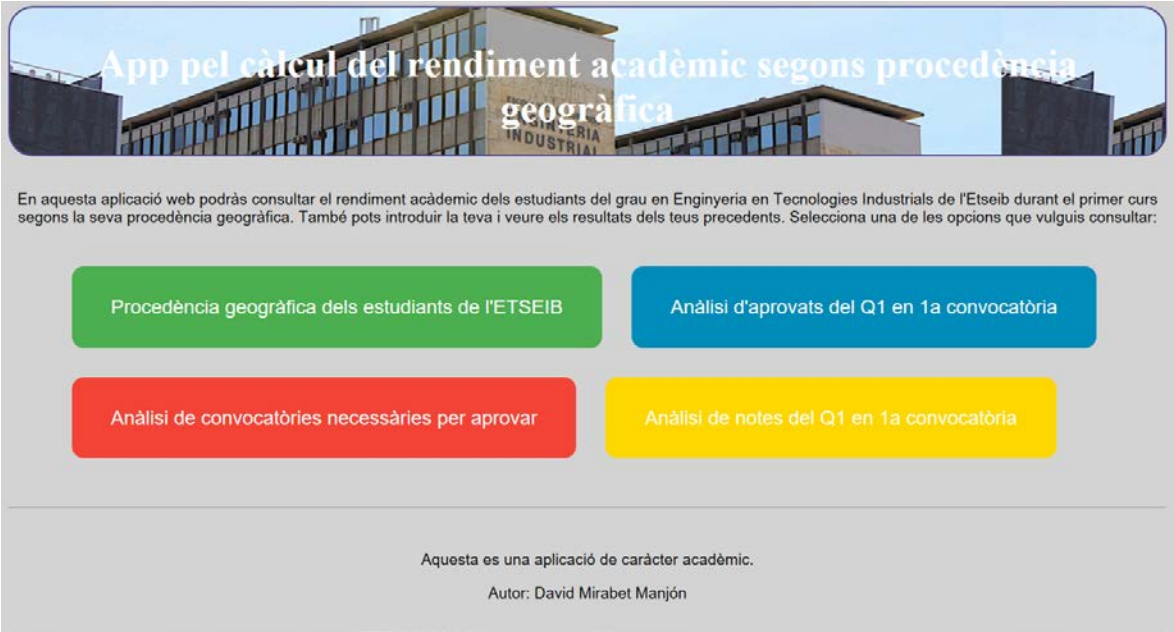



*Figura 7.2 Índex de l'aplicació web.*

## 7.3. JavaScript

JavaScript és el llenguatge utilitzat per programar el comportament de les pàgines web i que permet la interacció amb els usuaris. En referència a l'aplicació, s'ha donat molta importància a aquesta interacció amb l'usuari, intentant que la consulta de la informació fos el més intuïtiu possible. Pels diferents gràfics de Highcharts s'han creat 8 funcions de JavaScript mentre que pel mapa de Catalunya s'han utilitzat les que ja venien definides i se n'ha modificat la més important corresponent a la interacció a l'hora de clicar una comarca.

### 7.3.1. El mapa de Catalunya interactiu

Al realitzar un estudi geogràfic el component més atractiu de l'aplicació és la possibilitat de fer una consulta seleccionant fàcilment una determinada regió geogràfica d'un mapa interactiu. El mapa de Catalunya interactiu que s'ha fet servir s'ha obtingut de la pàgina web de Github i és gentilesa d'Enric Ballò. Github és una plataforma de desenvolupament inspirada en la forma de treballar, de codi obert per als negocis, creació i revisió de codis de programació, gestió de projectes i construcció de programari juntament amb milions d'altres desenvolupadors. (7)

El mapa parteix d'un fitxer SGV (Scalable Vector Graphics) basat en llenguatge HTML, aquesta extensió permet la creació de mapes i dibuixos redimensionables tant estàtics com dinàmics per mitjà de gràfics vectorials. El codi escrit en el fitxer correspon a un "path" (camí) que són les instruccions vectorials alfanumèriques per representar el mapa o dibuix i que els navegadors web interpreten i mostren per pantalla. Aquest "path" pot ser redimensionat i modificat amb JavaScript de tal manera que se li poden afegir complements i funcionalitats al mapa.

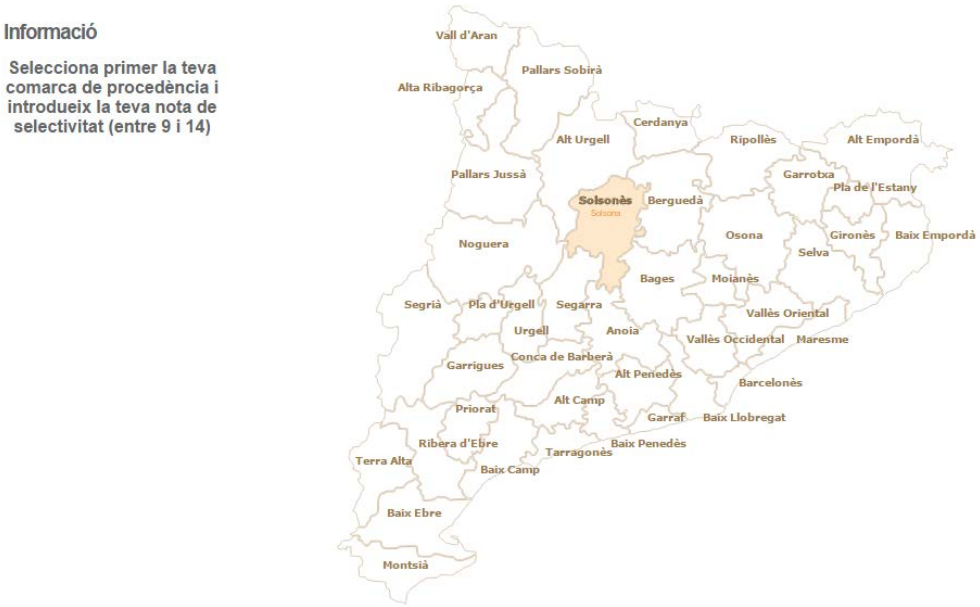



*Figura 7.3 Mapa de Catalunya interactiu.*

El codi JavaScript original del mapa s'ha modificat per tal de complir les necessitats de l'aplicació web, s'ha aprofitat la divisió comarcal i les animacions de les comarques en ser seleccionades. La funcionalitat més important que té el mapa és la de mostrar les estadístiques corresponents quan una comarca és seleccionada. L'algorisme utilitzat s'activa quan l'usuari fa clic sobre una comarca, llavors, apareix un quadre de text amb informació d'aquella comarca i el gràfic corresponent. En el cas del Barcelonès apareix una pestanya on l'usuari pot seleccionar el districte, altrament, s'envia el nom de la comarca seleccionada al fitxer *views.py* usant el mètode "post" de Flask que ho processa i retorna el gràfic corresponent. A més, l'algoritme juga amb el format condicional i l'atribut "display" per ocultar o mostrar en pantalla allò que ha de veure l'usuari. No existeixen dades en la mostra per a les comarques del Moianès i el Priorat, per tant, són tractades com un cas apart i no es mostrarà cap gràfic.

```
function onMapClick(comarcaName, contentText, comarcaLink){
    if(onClick){
        //console.log(comarcaLink);
        window.location=comarcaLink;

        $('#comarcaName').html('<h1>' + comarcaName + '</h1>');
        $('#contentText').html(contentText);
        if(comarcaName == 'Priorat' || comarcaName == 'Moianès'){
            container.style.display='none'
            districte.style.display='none'
            container2.style.display='none'
            cappreceden t.style.display='block'

        }else if(comarcaName == 'Barcelonès'){
            districte.style.display='block'
            cappreceden t.style.display='none'
            container.style.display='none'
        }else{
            container.style.display='block'
            districte.style.display='none'
            container2.style.display='none'
            cappreceden t.style.display='none'
            $.post("/",{
                postejo: comarcaName
            });
```

*Figura 7.4 Algoritme executat en fer clic sobre una comarca.*

### 7.3.2. Els gràfics de Highcharts

Highcharts és una llibreria escrita en Javascript que permet la creació de gràfics oferint un mètode fàcil i interactiu per inserir-los en una aplicació web. A l'aplicació es fan servir 3 tipus de gràfics diferents que demanen una certa organització de les dades en el fitxer *.csv* com s'ha vist en el punt 6.4. Tot i això, el gràfic de barres apilades i el de sectors requereixen d'algoritmes addicionals per representar la informació de manera correcte. (8)

a) *Gràfic de barres*

El gràfic representa els nivells 3,4 i 5 de l'arbre presentat en el punt 5.2 on el nivell 5 en aquest cas correspon a un percentatge. El seu codi és el mateix que proporciona Highcharts, l'única línia de codi addicional és la primera, *$.ajaxSetup({ cache: false });*, que serveix per a que en seleccionar una comarca el gràfic refresqui les dades del gràfic.

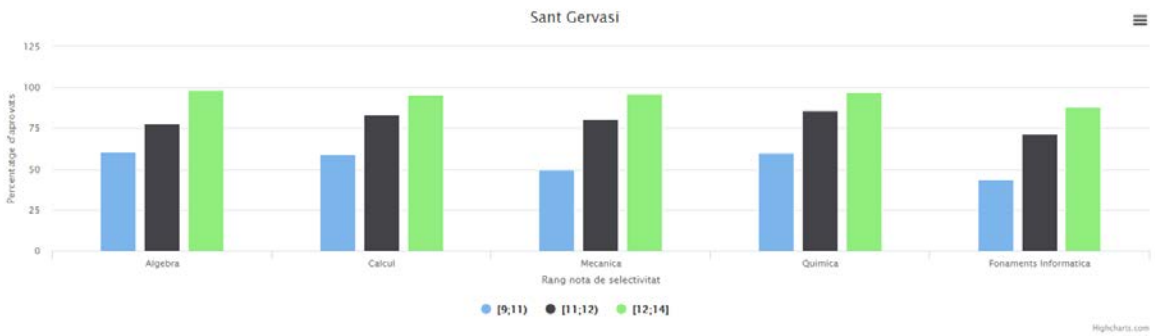
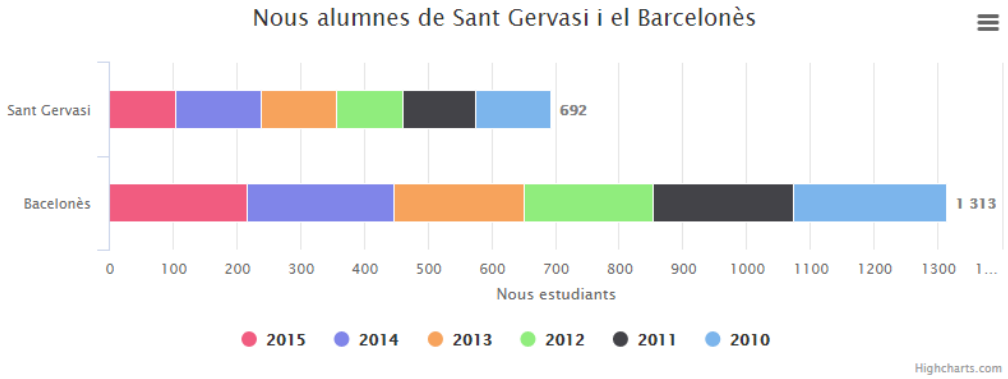



*Figura 7.5 Gràfic de barres del percentatge d'aprovats en primera convocatòria i assignatures del Q1 segons rang de selectivitat i comarca.*

*b) Gràfic de barres apilades*

S'han realitzat algunes modificacions al codi de Highcharts d'aquest gràfic per tal que reconegui les dades del arxiu *csv* de manera correcte. Aquest gràfic es fa servir representar la procedència dels estudiants segons comarca i any.




```
$.get('static/cat2.csv', function(data) {
    var lines = data.split('\n');  // linies del document csv

    $.each(lines, function(lineNo, line) {
        if (line == "") return; //fa que no aparegui la ultima linia csv
        var items = line.split(','); //separa les celes
        var series = {
            data: []
        };
        $.each(items, function(itemNo, item) {
            if (itemNo == 0) {
                series.name = item; //nom = nºconvocatoria
            } else {
                series.data.push(parseFloat(item)); //percentatge
            }
        });

        options.series.push(series);
```

*Figura 7.6 Exemple del gràfic de barres apilades i codi per obtenir les dades d'un arxiu CSV.*

*c) Gràfic de sectors*

El codi del gràfics de sectors també a patit algunes modificacions per adaptar les dades provinents del fitxer *csv*. Aquests gràfics representen els nivells 4 i 5 de l'arbre de classificació dels estudis de notes i convocatòries en assignatures del Q1.

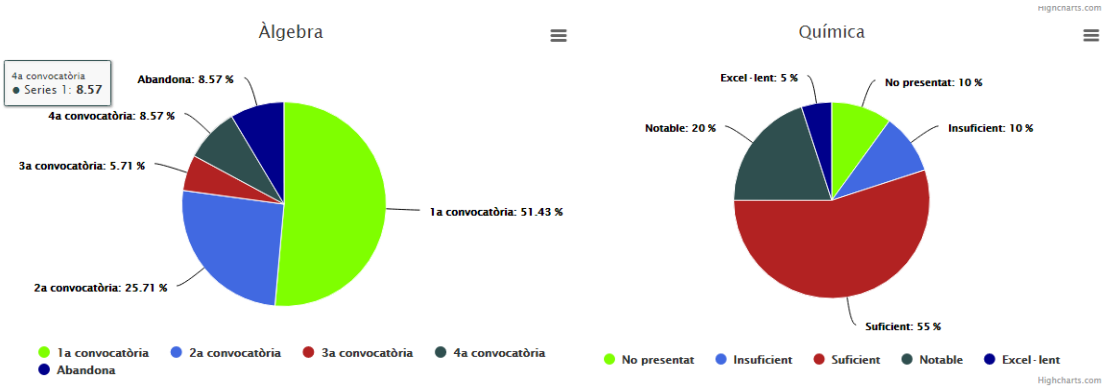



*Figura 7.7 Gràfics de sectors utilitzats en l'estudi de notes i convocatòries.*

# 8. PROPOSTA DE CONTINUACIÓ DEL TREBALL

Aquest treball ofereix la possibilitat de ser continuat estenent les funcionalitats de l'aplicació web. Les propostes de continuació recomanades són les següents:

a) Ampliar el nombre de variables que aporten informació sobre l'alumne abans d'entrar al grau, per exemple notes de selectivitat d'assignatures científiques, si prové d'un batxillerat públic o privat, si viu o no a Barcelona. Més variables, sobretot quantitatives, obririen les portes a poder realitzar models de regressió lineal múltiple amb les que fer prediccions de major confiança.

b) Automatitzar el sistema d'obtenció de fitxers CSV per tal que quan s'afegeixin noves notes a Prisma aquests s'actualitzin.

c) Creació d'un mapa interactiu dels diferents districtes de Barcelona a partir d'un fitxer SVG que es pot trobar fàcilment a Wikimedia Commons.

d) Ampliar l'estudi a assignatures de Q2 i altres cursos.

e) Depurar les prediccions errònies que sorgeixen de l'arbre de classificació i intentar ajustar les respostes nodals per tal de minimitzar l'entropia *H[Y]* i maximitzar la informació *I[Y:C]*

# 9. PLANIFICACIÓ TEMPORAL I COSTOS

## 9.1. Planificació temporal

La primera setmana, es van revisar els coneixements adquirits a les assignatures d'Estadística i Informàtica. Després, es van dedicar unes setmanes a l'aprenentatge de la llibreria Pandas de Python i algunes tècniques d'estadística descriptiva. Seguidament, es va començar a programar els algoritmes en Python i entremig es va començar a adquirir coneixements de programació web i Flask. A mesura que es van anar adquirint aquests coneixements s'anava donant forma a la interfície d'usuari de l'aplicació.

Els algoritmes de Python i la programació web es van treballar bastant en paral·lel ja que hi havia molt components de la programació web com Highcharts que requerien processar d'una manera determinada les dades. Finalment, les 8 últimes setmanes van ser dedicades a la redacció de la memòria.

Seguidament es mostra el diagrama de Gantt seguit per a l'elaboració del projecte.

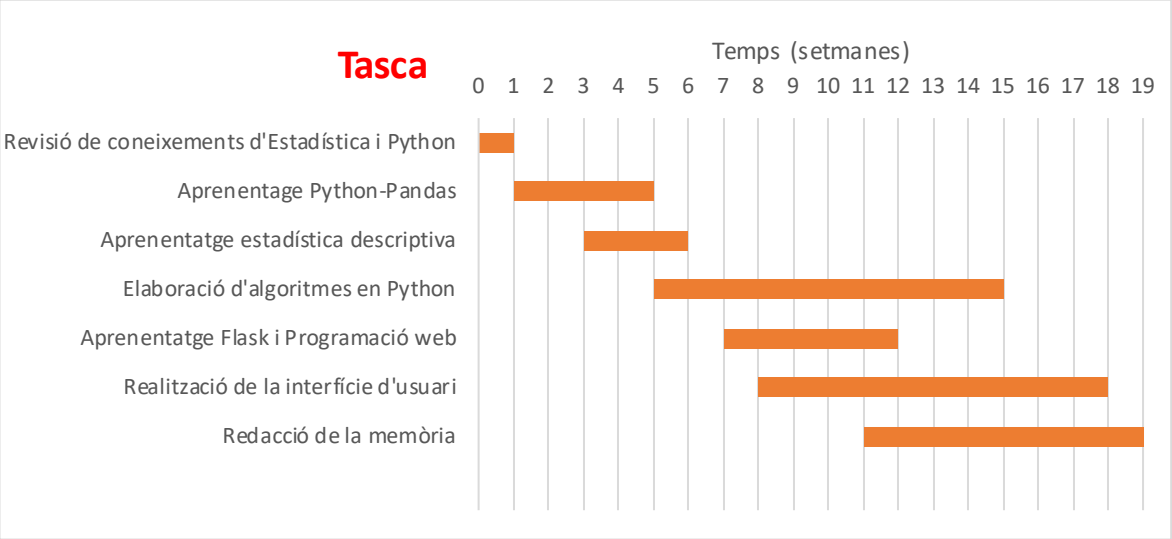



*Figura 9.1 Diagrama de Gantt del projecte.*

## 9.2. Balanç de costos

En primer lloc es tindrà en compte el treball realitzat per l'estudiant, considerant que aquest és un graduat en enginyeria de perfil júnior. El sou estàndard de mercat d'un enginyer graduat júnior és d'uns 20€/hora aproximadament. D'altra banda es considera que el conjunt d'hores treballades és l'equivalent a 12 crèdits ECTS, cada crèdit són 28 hores de mitjana de treball, per tant 336 hores. Així doncs, el cost de l'estudiant és de 336 hores a raó de 25€/hora: 8400€.

En segon lloc, la feina de supervisió realitzada pel tutor del projecte, assimilant el seu sou al d'un enginyer sènior. S'aproxima el temps dedicat 20 hores i el sou de l'enginyer a 35€ l'hora, per tant una despesa total de 700€.

El cost de la llicència dels programes de programació no tenen cap cost ja que són gratuïtes. Les dades amb les que s'ha treballat han estat proporcionades per la universitat i l'Institut d'Estadística de Catalunya sense cap cost a considerar. La llicència dels programes de Microsoft Office usats per redactar i presentar el projecte és de 69,00 € l'any i s'ha utilitzat durant 4 mesos, per tant:

$$Cost\ llicències = \frac{69{,}00\ €}{12\ mesos} \times 4\ mesos = 23\ €$$

Per altra banda, el cost de l'ordinador utilitzat és de 650 € i té una esperança de vida de 5 anys, el cost en proporció és:

$$Cost\ ordinador = \frac{660\ €}{60\ mesos} \times 4\ mesos = 44\ €$$

El consum de llum i internet es considera d'aproximadament 300 €, mentre que el transport correspon a 105 €, l'equivalent a una targeta T-jove dels transports metropolitans de Barcelona. La suma total del projecte és de 9572 € i el resum es mostra a la taula següent:

| Concepte del cost | Cost (€) |
|---|---|
| **Hores estudiant** | 8400 |
| **Hores tutor** | 700 |
| **Llicències** | 23 |
| **Ordinador** | 44 |
| **Llum i internet** | 300 |
| **Transport** | 105 |
| **Cost Total** | **9572** |

*Taula 9.1 Resum de les despeses del projecte.*

# 10. IMPACTE SOCIAL I AMBIENTAL

## 10.1.Impacte social

L'aplicació web desenvolupada serveix com a eina de consulta. Els nous estudiants del Grau en Enginyeria en Tecnologies Industrials tenen la possibilitat de conèixer el rendiment dels seus precedents segons la comarca d'on provenien i d'aquesta manera valorar quines són les assignatures del primer quadrimestre que els resultaran més i menys difícil d'afrontar. Segons la comarca d'origen i la seva proximitat amb la universitat el nou estudiant pot plantejar-se mudar-se a Barcelona o no i així organitzar-se millor acadèmicament.

Per altra banda, el tractament de dades amb cert nivell de confidencialitat com pot ser el tipus de batxillerat cursat, públic o privat, i el lloc de residència de l'estudiant proposats com a continuació del projecte proporcionarien un valor afegit a l'aplicació de manera que les decisions dels estudiants després d'una consulta serien més transcendentals i rellevants.

## 10.2.Impacte ambiental

Degut a la naturalesa de caràcter informàtic del projecte, l'impacte ambiental que presenta és molt baix. Les principals causes d'impacte ambiental són:

a) El consum energètic de l'ordinador, és el més important degut a l'ús que se n'ha fet, sent l'ordinador l'eina principal de treball.

b) Els recursos energètics per a la climatització de la sala on s'ha treballat.

c) L'efecte mediambiental dels residus generats pel material d'oficina.

# Conclusions

En aquest projecte s'ha desenvolupat una aplicació web que permet a l'estudiant consultar el rendiment, just començar el GETI, dels seus precedents segons la comarca Catalana de procedència i organitzar-se acadèmicament en conseqüència.

D'acord amb els objectius del projecte, s'han utilitzat 18 fitxers csv i 18 funcions de python organitzades en diferents mòduls per a la implementació del processat de dades. Amb la llibreria Pandas de Python s'han creat els algoritmes necessaris per crear els arbres de classificació, filtrar i crear noves dades a partir de les existents per treballar de manera òptima. S'han emprat els arbres de classificació juntament amb la predicció distributiva com a mètodes de classificació i predicció del conjunt d'alumnes i el seu rendiment acadèmic.

Una vegada processades les dades s'ha desenvolupat l'aplicació web interactiva a partir de 5 documents HTML i 3 fitxers CSS els quals s'han vinculat als programes python mitjançant l'entorn de treball Flask on s'han creat 7 funcions de python més.

Per a la representació dels resultats s'han utilitzat 8 funcions de JavaScript corresponents als gràfics de Highcarts que representen les prediccions de manera gràfica i entenedora a l'aplicació. També s'han modificat les funcions de JavaScript del mapa de Catalunya per tal d'adaptar-lo als propòsits de l'aplicació.

L'aplicació compleix els objectius inicials, és interactiva, el seu disseny es atractiu i permet fer consultes diverses. Tot i això, el model de classificació i predicció té les seves mancances degut a una mostra insuficient en funció de la comarca Catalana.

Finalment, s'han fet algunes propostes interessants sobre la continuació del projecte i millora de l'aplicació.

# Agraïments

M'agradaria donar gràcies al meu tutor Lluis Solano, per haver-me guiat i ajudat en les etapes d'aquest projecte amb les seves explicacions, consells i sobretot amb el seu bon humor. Es d'agrair també el suport de la meva família i amics que han viscut amb mi el transcurs d'aquest projecte.

# Bibliografia

## Referències bibliogràfiques

**[1]** **IDESCAT.** Institut d'Estadística de Catalunya per a la consulta d'informació estadística. [https://www.idescat.cat/serveis/consultes, 21 de febrer de 2017]

**[2]** **Pydata.org.** *Python Data Analysis Library (Pandas) [http://pandas.pydata.org,* 15 de febrer de 2017]

**[3]** **Miguel Grinberg's blog.** *The Flask Mega-Tutorial, Part I: Hello, World!* [https://blog.miguelgrinberg.com/post/the-flask-mega-tutorial-part-i-hello-world, 10 d'abril de 2017]

**[4]** **Refsnes Data.** *w3school.com The World's Largest Web Developer Site.* [https://www.w3schools.com, 1 d'abril de 2017]

**[5]** **Carnegie Mellon University.** *Classification and Regression Trees 36-350, Data Mining 6 November 2009.* [http://www.stat.cmu.edu/~cshalizi/350/lectures/22/lecture-22.pdf, 4 de maig de 2017]

**[6]** **David Feldman and Shulamith Gross.** *MORTGAGE DEFAULT: CLASSIFICATION TREES ANALYSIS March 24, 2003* [http://mi.eng.cam.ac.uk/~mjfg/local/4F10 /Feldman_Gross.pdf, 14 de maig de 2017]

**[7]** **Github developement platform.** *Interactive Map of Catalunya SVG/VML and RaphaëlJs - Demo.* [https://github.com/eballo/catalunya-map, 12 d'abril de 2017]

**[8]** **Highsoft.** *Interactive JavaScript charts for your web pages.* [https://www.highcharts.com, 20 d'abril de 2017]

## Bibliografia complementària

**[1] Universidad de Cartagena**. *Teoria de los árboles de decisión.* [http://www.dmae.upct.es/~mcruiz/Telem06/Teoria/arbol_decision.pdf, 1 d'abril de 2017]

**[2] CASE -** Center of Applied Statistics and Economics Humboldt University, Berlin. *Classification and Regression Trees (CART) Theory and Applications A Master Thesis Presented by Roman Timofeev 20/12/2004* [http://edoc.hu-berlin.de/master/timofeev-roman-2004-12-20/PDF/timofeev.pdf, 19 de març de 2017]

**[3] Stack Exchange Network.** *Stack Overflow comunity.* [https://stackoverflow.com/, 30 de febrer de 2017]

**[4] Pocoo.org.** *Flask web development, one drop at a time.* [http://flask.pocoo.org/, 10 d'abril de 2017]

# Annex

# 1. Mòduls de Python

## 1.1 Condicionament de fitxers inicials

```
from math import *
import pandas as pd
import programa as p
import numpy as np

districtes_bcn = [‘Badalona’, ‘Ciutat Vella’, ‘Eixample Dreta’,
‘Eixample Esquerra’, ‘Gracia’, ‘Horta-Guinardo’, ‘Hospitalet’, ‘Les
Corts’, ‘Nou Barris’, ‘Sant Andreu’, ‘Sant Gervasi’, ‘Sant Marti’,
‘Sants-Montjuic’]

anys = [‘2010’,’2011’,’2012’,’2013’,’2014’,’2015’]

comarques_cat=[‘Alt Camp’, ‘Alt Emporda’, ‘Alt Penedes’, ‘Alt Urgell’,
‘Alta Ribagorca’, ‘Anoia’, ‘Bages’, ‘Baix Camp’, ‘Baix Ebre’, ‘Baix
Emporda’, ‘Baix Llobregat’, ‘Baix Penedes’, ‘Bergueda’, ‘Cerdanya’,
‘Conca de Barbera’, ‘Garraf’, ‘Garrigues’, ‘Garrotxa’, ‘Girones’,
‘Maresme’, ‘Montsia’, ‘Noguera’, ‘Osona’, ‘Pallars Jussa’, ‘Pallars
Sobira’, ‘Pla Estany’, ‘Pla Urgell’, ‘Ribera Ebre’, ‘Ripolles’, ‘Santa
Coloma’, ‘Segarra’, ‘Segria’, ‘Selva’, ‘Solsones’, ‘Tarragones’,
‘Terra Alta’, ‘Urgell’, ‘Vall Aran’, ‘Valles Occidental’, ‘Valles
Oriental’]

#...MODUL DE CONDIONAMENT DE FITXERS INICIALS
#junta els dos fitxers de dades base
def juntar():
    df1 = pd.read_csv(open(‘DadespersIND16.csv’,’r’))
    df2 = pd.read_csv(open(‘QualifINDfaseini16.csv’,’r’))
    data = pd.merge(df1,df2)
    data.to_csv(‘juntat.csv’)

#afegeix el nom de la comarca o districte
def comarca():
    df = pd.read_csv(open(‘DadespersIND16.csv’,’r’))
    dis = pd.read_csv(open(‘Corresp_cques_CAT_codpost.csv’,’r’))
    df[‘COM’] = pd.Series(‘ ‘, index=df.index)
    dis.set_index(‘CPE’,inplace = True)
    i = 0
    for cpe in list(df.CPE):
        if cpe in dis.index:
            comarca = dis.loc[cpe].COM
            df.set_value(i, ‘COM’, comarca)
            df.set_value(i, ‘DIS’, dis.loc[cpe].DIS)
        i = 1 + i

    df.to_csv(‘indi+com.csv’)
```

## 1.2 Filtrat de dades

```
#... MODUL DE FILTRAT DE DADES
#filtra per grau universitari i per nota de selectivitat
def filtra_grau_i_sele(df):
    df = df[(df.CODI_PROGRAMA == 752) & (9 <= df.SELE <= 14)]
    return df

#arreglar curs i any, un estudiant entra el any que cursa la 1ª
3plit3s3ca
def arregla_any():
    df = pd.read_csv(open('juntat.csv','r'))
    df.set_index('CODEX', inplace=True)
    for i in list(df.index):
        dflocal = df.loc[i]
        if type(dflocal.CURS) is np.int64:
            curs = dflocal.CURS
        else:
            curs = min(dflocal.CURS)
        if type(dflocal.ANY) is np.int64:
            if curs != dflocal.ANY:
                df.set_value(i,'ANY',curs)
        else:
            if curs not in  dflocal.ANY:
                df.set_value(i,'ANY',curs)
    df.to_csv('juntat.csv')

#elimina celles buides i duplicats
def depurar(df):
    df = df.drop.duplicates()
    df = df.replace(np.nan,' ', regex=True)
    for column in df:
        a=0
        for valor in df[column]:
            if valor == ' ':
                df = df.drop(a)
            a=a+1
    df = df.reset_index(drop = True)
```

## 1.3 Creació de variables

```
#... MODUL DE CREACIO DE VARIABLES
#afegim asignatures i la seva nota a indi+com
def nota_1ª_convocatoria():
    df = pd.read_csv(open('indi+com.csv','r'))
    df2 = pd.read_csv(open('juntat.csv','r'))
    df.set_index('CODEX', inplace=True)
    q1 = ['Algebra','Calcul','Mecanica','Quimica','Fonaments
Informatica']
    for assignatura in q1:
        df[assignatura] = pd.Series(' ', index=df.index)

    i = 0 #numero de la fila
    df2 = df2[df2.Q == 1]
    df2 = df2.reset_index(drop = True)
    for assignatura in df2.CODI_UPC_UD:
        alumne = df2.CODEX[i]
        if alumne in df.index:
            if (df2.ANY[i] == df2.CURS[i]):
                if assignatura in q1:
                    df.set_value(alumne, assignatura,
nota_categorica(float(df2.NF[i])))
        i = i + 1
    df.to_csv('indi+com.csv')


#obte el rang de selectivitat
def rangsele(sele):
    if 9 <= sele < 11:
        rang = 1
    elif 11 <= sele < 12:
        rang = 2
    elif 12 <= sele <= 14:
        rang = 3
    return rang


#posa les notes en 4plit4s4cal
def nota_categorica(nota):
    if  nota == 0.0:
        nota = 'No presentat'
    elif 0.0 < nota < 5:
        nota = 'Insuficient'
    elif 5 <= nota < 7:
        nota = 'Suficient'
    elif 7 <= nota < 9:
        nota = 'Notable'
    elif nota >= 9:
        nota = 'Excelent'
    else:
        nota = ''
    return nota
```

```
#convocatories que han realitzat els alumnes per cada assignatura
def convocatoria():
    df = pd.read_csv(open('juntat.csv','r'))
    df.set_index('CODEX', inplace=True)
    df2 = pd.read_csv(open('indi+com.csv','r'))
    df2.set_index('CODEX', inplace=True)
    #el fitxer juntat.csv conte el nom de les assignatures i no els
codis
    q1 = ['Algebra','Calcul','Mecanica','Quimica','Fonaments
Informatica']
    for assignatura in q1:
        df2[assignatura] = pd.Series(' ', index=df2.index)

    for alumne in list(df2.index):
        for assignatura in q1:
            q1alumne = df.loc[alumne] #DataFrame
            if assignatura in list(q1alumne['CODI_UPC_UD']):
                q1alumne = q1alumne[q1alumne['CODI_UPC_UD'] ==
assignatura]
                convocatories = list(q1alumne.APR)
                if not 'S' in convocatories:
                    df2.set_value(alumne, assignatura, 100)
                else:
                    df2.set_value(alumne, assignatura,
len(convocatories))
            else:
                df2.set_value(alumne, assignatura, 0)

    df2.to_csv('convo.csv')
```

## 1.4 Arbre de classificació

```
#... MODUL ARBRE DE CLASSIFICACIO
#comptador d'aprovats de cada asignatura segons comarca i rang
def percentatge_aprovats():
    df = pd.read_csv(open('indi+com.csv','r'))
    dc = pd.read_csv(open('com+asig.csv','r'))
    dc.set_index(['COM','RANG'], inplace=True)

    l = ['A','C','M','Q','F'] #columnes amb el numero d'aprovats
    t = ['TA','TC','TM','TQ','TF'] #columnes del total d'alumnes
    ll = [9,10,11,12,13] #posicio de les columnes assignatura en les
tuples

    for row in df.itertuples():
        com = row.COM
        rang = rangsele(row.SELE)
        c = 0  #variable que recorre les llistes l i t
        for assignatura in ll:
            nota = row[assignatura]
            aprovats = dc.loc[(com,rang)][l[c]]
            total = dc.loc[(com,rang)][t[c]]
            if nota == '':
                pass
            else:
                dc.set_value((com,rang), t[c], (total + 1))
                if (nota != 'No presentat') and (nota !=
'Insuficient'):
                    dc.set_value((com,rang), l[c], (aprovats + 1))
            c = c+1
            if c > 4:
                c = 0
    dc.to_csv('com+asig.csv')


#comptador nota categorica per assignatura
def comptador_nota_categorica():
    df = pd.read_csv(open('indi+com.csv','r'))
    dc = pd.read_csv(open('nota_categorica.csv','r'))
    dc.set_index(['COM','RANG','NOTA'], inplace=True)
    l = ['A','C','M','Q','F'] #columnes amb el numero d'aprovats
    ll = [9,10,11,12,13] #posicio de les columnes assignatura en les
tuples

    for row in df.itertuples():
        com, rang = (row.COM,  rangsele(row.SELE))
        c = 0 #variable per recorre les llistes l i t
        for assignatura in ll:
            nota = row[assignatura]
            if nota == ' ':
                pass
            else:
                comptador = dc.loc[(com,rang,nota)][l[c]]
                dc.set_value((com,rang,nota), l[c], ( comptador + 1))
                c = c+1
                if c > 4:
                    c = 0

    for assig in l:
        dc['P'+ assig] = pd.Series('', index=dc.index)

    for row in dc.iterrows():
```

```
        com, rang, nota = (row[0][0], row[0][1], row[0][2])
        for assig in l:
            total = sum(dc.loc[(com,rang)][assig])
            if total == 0:
                pass
            else:
                percentatge =
"{0:.2f}%".format(float(row[1][assig])/total*100.0)
                dc.set_value((com,rang,nota), 'P'+assig, percentatge)
    dc.to_csv('nota_categorica.csv')


#comptador convocatories segons comarca i rang
def convocatories_aprobar():
    df = pd.read_csv(open('convo.csv','r'))
    dc = pd.read_csv(open('com+convo.csv','r'))
    dc.set_index(['COM','RANG','CONVO'], inplace=True)
    l = ['A','C','M','Q','F'] #columnes amb el numero d'aprovats
    ll = [9,10,11,12,13] #posicio de les columnes assignatura en les
tuples

    for row in df.itertuples():
        com, rang = (row.COM,  rangsele(row.SELE))
        c = 0 #variable per recorre les llistes l i t
        for assignatura in ll:
            convocatoria = row[assignatura]
            if convocatoria == 0:
                pass
            else:
                convocats = dc.loc[(com,rang,convocatoria)][l[c]]
                dc.set_value((com,rang,convocatoria), l[c], (convocats
+ 1))
                c = c+1
                if c > 4:
                    c = 0
    dc.to_csv('com+convo.csv')


#percentages per comarca i rang de la contribucio de les convocatories
def percentatge_convocatories():
    dc = pd.read_csv(open('com+convo.csv','r'))
    dc.set_index(['COM','RANG','CONVO'], inplace=True)
    l = ['A','C','M','Q','F'] #columnes amb el numero d'aprovats
    for assig in l:
        dc['P'+ assig] = pd.Series('', index=dc.index)

    for row in dc.iterrows():
        com, rang, convo = (row[0][0], row[0][1], row[0][2])
        for assig in l:
            total = sum(dc.loc[(com,rang)][assig])
            if total == 0:
                pass
            else:
                percentatge =
"{0:.2f}%".format(float(row[1][assig])/total*100.0)
                dc.set_value((com,rang,convo), 'P'+assig, percentatge)
    dc.to_csv('com+convo.csv')
```

## 1.5 Condicionament de fitxers per a Highcharts

```
#... MODUL DE CONDICIONAMENT DE FITXERS PER A HIGHCHARTS

#csv del grafic de barres apilades per l'estudi de procedencia dels
alumnes
def procedencia_alumnes():
    df = pd.read_csv(open('indi+com.csv','r'))
    cat = pd.DataFrame(0, index = comarques_cat, columns = anys)
    bcn = pd.DataFrame(0, index = districtes_bcn, columns = anys)

    for row in df.itertuples():
        com = row.COM
        Any = str(row.ANY)
        if com in districtes_bcn:
            bcn.set_value(com, Any, bcn.loc[com][Any]+1)
        else:
            cat.set_value(com, Any, cat.loc[com][Any]+1)

    cat.set_index('2010',inplace = True)
    bcn.set_index('2010',inplace = True)
    (cat.transpose()).to_csv('cat2.csv')
    (bcn.transpose()).to_csv('bcn2.csv')


#csv del grafic de barres per l'estudi de percentatge d'aprovats en 1a
convocatoria
def obtecomarca(comarca):
    df = pd.read_csv(open('app/static/com+asig.csv','r'))
    df = df[df.COM == comarca]
    df.set_index('RANG',inplace=True)
    q1 = ['Algebra','Calcul','Mecanica','Quimica','Fonaments
Informatica']
    d = [] #llista necessaria per
    for assig  in q1:
        percentatge = []
        for rang in df.index:

percentatge.append("{0:.2f}".format(df.loc[rang][assig[0]]/df.loc[rang
]['T'+assig[0]]*100))
        d = d + [(assig,percentatge)]

    taula = pd.DataFrame.from_items(d, orient='index', columns =
list(df.index))
    taula.to_csv('app/static/taula.csv')


#csv del grafic de sectors per l'estudi de distribucio de
convocatories
def obte_convocatoria(comarca,sele):
    df = pd.read_csv(open('app/static/quesitos/com+convo.csv','r'))
    df = df[(df.COM == comarca) & (df.RANG == rangsele(sele))]
    df = df[['CONVO','PA','PC','PM','PQ','PF']]
    df.set_index('CONVO', inplace = True)
    assig = list(df.transpose().iterrows())
    if len(assig[0][1]) == 0:
        return False  #informa al JavaScript de l'inici en blanc
    else:
        neteja(assig[0][1]).to_csv('app/static/quesitos/algebra.csv')
        neteja(assig[1][1]).to_csv('app/static/quesitos/calcul.csv')
```

```
        neteja(assig[2][1]).to_csv('app/static/quesitos/mecanica.csv')
        neteja(assig[3][1]).to_csv('app/static/quesitos/quimica.csv')

neteja(assig[4][1]).to_csv('app/static/quesitos/fonaments.csv')
        return True

#csv del grafic de sectors per l'estudi de distribuci□r nota
categorica
def obte_notes(comarca,sele):
    df = pd.read_csv(open('app/static/notes/nota_categorica.csv','r'))
    df = df[(df.COM == comarca) & (df.RANG == rangsele(sele))]
    df = df[['NOTA','PA','PC','PM','PQ','PF']]
    df.set_index('NOTA', inplace = True)
    assig = list(df.transpose().iterrows())
    if len(assig[0][1]) == 0:
        return False  #informa al JavaScript de l'inici en blanc
    else:
        neteja(assig[0][1]).to_csv('app/static/notes/algebra.csv')
        neteja(assig[1][1]).to_csv('app/static/notes/calcul.csv')
        neteja(assig[2][1]).to_csv('app/static/notes/mecanica.csv')
        neteja(assig[3][1]).to_csv('app/static/notes/quimica.csv')
        neteja(assig[4][1]).to_csv('app/static/notes/fonaments.csv')
        return True

#elimina files amb 0% perque no siguin representades en els grafics de
sectors
def neteja(df):
    for convocatoria in df.index:
        if df.loc[convocatoria] == '0.00%':
            df = df.drop(convocatoria)
    return df
```

## 1.6 Funcions de Flask

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-

from flask import render_template
from app import app
import pandas as pd
from flask import request, jsonify
import sys
from unicodedata import normalize, category

@app.route('/', methods=['GET','POST'])
@app.route('/index')
def index():
    return render_template('index.html')



@app.route('/assignatures', methods=['GET','POST'])
def assig():
    #obte el nou csv preparat per el highchart de la comarca en
questio
    comarca = request.form.get('postejo','Noguera')
    cap=comarca
    mostra ='none'
    comarca = apostrof(normaliza(comarca))
    if comarca in l:
        mostra='block'

    taula = obtecomarca(comarca)

    return render_template("assignatures.html",
                           mostra=mostra,
                           cap=cap)


@app.route('/convocatoria', methods=['GET','POST'])
def convo():
    comarca = request.form.get('postejo','inici')
    sele = float(request.form.get('sele',11.0))
    capfals="No existeixen precedents per " + comarca + " i nota de
selectivitat de "+str(sele)
    cap = "Convocatòries per " + comarca + " i nota de selectivitat de
"+str(sele)
    comarca = apostrof(normaliza(comarca))
    funciona = obte_convocatoria(comarca,sele)

    mostradistricte='none'
    mostrasele='none'
    mostracap='none'
    mostra='none'

    if funciona:
        if comarca in l:
            mostradistricte='block'

        mostra='block'
        mostracap='block'
```

```
    else:
        if comarca != 'inici':
            cap=11plit11s
            mostracap='block'

    return render_template("convocatoria.html",
                           cap=cap,
                           mostradistricte=mostradistricte,
                           nomcomarca=comarca,
                           mostra=mostra,
                           mostracap=mostracap)

@app.route('/notes', methods=['GET','POST'])
def nota_categorica():
    comarca = request.form.get('postejo','inici')
    sele = float(request.form.get('sele',11.0))
    capfals="No existeixen precedents per " + comarca + " i nota de
selectivitat de "+str(sele)
    cap = "Notes categòriques per " + comarca + " i nota de
selectivitat de "+str(sele)
    comarca = apostrof(normaliza(comarca))
    funciona = obte_notes(comarca,sele)

    mostradistricte, mostrasele , mostracap, mostra =('none', 'none',
'none', 'none')

    if funciona:
        if comarca in l:
            mostradistricte='block'

        mostra='block'
        mostracap='block'

    else:
        if comarca != 'inici':
            cap=11plit11s
            mostracap='block'

    return render_template("notes.html",
                           cap=cap,
                           mostradistricte=mostradistricte,
                           nomcomarca=comarca,
                           mostra=mostra,
                           mostracap=mostracap)


@app.route('/estadistiques', methods=['GET','POST'])
def estadis():
    return render_template("estadistiques.html")

def normaliza(s):
   return ''.join((c for c in normalize('NFD', s) if category© !=
'Mn'))

def apostrof(comarca):
     if "'" in comarca:
        c = comarca.split("'")
        if c[1]=='Urgell':
            c='Pla'
        else:
            return c[1]
     else: return comarca
```

# 2. Codi HTML de l'aplicació web (templates)

## 2.1 Índex de l'aplicació

```
<!DOCTYPE html>
<html lang="ca">
  <head>

    <meta charset="UTF-8">
    <title>GEOETSEIB</title>
    <link rel="stylesheet" href="static/css/styleindex.css">
  </head>

  <body bgcolor="#D3D3D3">
    <div align="center" class="container">
      <h2 align="center"> App pel càlcul del rendiment acadèmic segons
procedència geogràfica</h2>

    </div>
    <br>
    <p > En aquesta aplicació web podràs consultar el rendiment
acàdemic dels estudiants del grau en Enginyeria en Tecnologies
Industrials de l'Etseib durant el primer curs segons la seva
procedència geogràfica. També pots introduir la teva i veure els
resultats dels teus precedents. Selecciona una de les opcions que
vulguis consultar:</p>

    <form action="/estadistiques" method="POST">
      <input  class="boto boto1" type="submit"
value="Proced&egrave;ncia geogr&agrave;fica dels estudiants de
l'ETSEIB"></input>
    </form>
    <!--boto convcatoria-->
    <form action="/assignatures" method="POST">
      <input class="boto boto2" type="submit"   value="Anàlisi
d'aprovats del Q1 en 1a convocatòria"></input>
    </form>
    <form action="/convocatoria" method="POST">
      <input  class="boto boto3" type="submit"  value="Anàlisi de
convocat&ograve;ries necessàries per aprovar    "></input>
    </form>
    <form action="/notes" method="POST">
      <input  class="boto boto4" type="submit"  value="Anàlisi de
notes del Q1 en 1a convocatòria    "></input>
    </form>
    <br>
    <br><hr>
    <br>

    <p> Aquesta es una aplicació de caràcter acadèmic.</p>
    <p> Autor: David Mirabet Manjón</p>

  </body>
</html>
```

## 2.2 Estadístiques de la procedència geogràfica

```
<body>
    <br>
  <form action="/" method="POST">
      <h2 style="display: inline;" > Gràfic de barres apilades per
anys dels nous estudiants del Grau en Enginyeria en Tecnologies
Industrials </h2><input position="left" class="boto boto2"
type="submit"   value="Índex"></input>
  </form>
  <br>
    <div id="container2" style="min-width: 310px; max-width: 800px;
height: 1200px; margin: 0 auto"></div>
    <script src="static/cat.js" type="text/javascript"></script>
    <br><br>
    <div id="container3" style="min-width: 310px; max-width: 800px;
height: 1200px; margin: 0 auto"></div>
    <script src="static/bcn.js" type="text/javascript"></script>
    <br><br>
    <div id="container4" style="min-width: 310px; max-width: 800px;
height: 300px; margin: 0 auto"></div>
    <script src="static/santgervasi.js"
type="text/javascript"></script>
    <br><br>
  </body>
```

## 2.3 Estudi dels aprovats en 1a convocatoria

```
<body>
  <br>
  <form action="/" method="POST">
      <h2 style="display: inline;" > Analitzador probabilístic
d'aprovats en el Q1 i primera convocatòria
      </h2><input position="left"
class="boto boto2" type="submit"   value="Índex"></input>
  </form>
  <br>
  <!--//highchart-->
  <p style="display:none;" id="cappreceden t">No existeix cap precedent
per a la comarca seleccionada</p>
  <script type="text/javascript" src="static/barres.js"></script>
  <div id="container"  style="min-width: 310px; display: none; height:
400px; margin: 0 auto"></div>
  <div id="container2" style="min-width: 310px; display: none; height:
400px; margin: 0 auto"></div>


  <!-- boto que fa canviar el highchart-->
  <form action="/assignatures" method="POST" style="display: none"
id="districte" >
    <br>
    <select name="postejo">
      <option value="Badalona">Badalona</option>
      <option value="Ciutat Vella">Ciutat Vella</option>
      <option value="Eixample Dreta">Eixample Dreta</option>
      <option value="Eixample Esquerra">Eixample Esquerra</option>
```

```
      <option value="Gr&agrave;cia">Gr&agrave;cia</option>
      <option value="Horta-Guinard&oacute;">Horta-
Guinard&oacute;</option>
      <option value="Hospitalet de Llobregat">Hospitalet de
Llobregat</option>
      <option value="Les Corts">Les Corts</option>
      <option value="Nou Barris">Nou Barris</option>
      <option value="Sant Andreu">Sant Andreu</option>
      <option value="Sant Gervasi">Sant Gervasi</option>
      <option value="Sant Mart&iacute;">Sant Mart&iacute;</option>
      <option value="Santa Coloma">Santa Coloma</option>
      <option value="Sants-Montjuic">Sants-Montjuic</option>

    </select>  Selecciona el teu districte o poblaci&oacute;
del Barcelon&egrave;s  
    <input type="submit"   value="Actualitza"></input>
  </form>


  <script>

    var titular='{{cap}}';
    document.getElementById("container2").style.display='{{mostra}}';
    document.getElementById("districte").style.display='{{mostra}}';
  </script>


  <!--mapa catalunya-->
  <div class="mapWrapper">
      <div id="map"></div>
      <div id="text">
        <div id="comarcaName"><h1>Informaci&oacute;</h1></div>
        <div id="contentText">Selecciona una comarca del mapa i fes
clic per veure el seu contingut</div>
      </div>
  </div>

  </body>
</html>
```

## 2.4 Estudi de las convocatòries del Q1

```
  <body>
    <br>
    <form action="/" method="POST">
      <h2 style="display: inline;" >Analitzador probabilístic de
convocatòries necessàries per superar les assignatures del Q1
      </h2><input position="left"
class="boto boto2" type="submit"   value="Índex"></input>
  </form>
    <br>

    <p id="titula"> {{cap}}</p>

    <script type="text/javascript"
src="static/quesitos/quesitoalgebra.js"></script>
    <script type="text/javascript"
src="static/quesitos/quesitocalcul.js"></script>
    <script type="text/javascript"
src="static/quesitos/quesitomecanica.js"></script>
    <script type="text/javascript"
src="static/quesitos/quesitoquimica.js"></script>
    <script type="text/javascript"
src="static/quesitos/quesitofonaments.js"></script>

    <div id="container1" style="min-width: 310px; display: none;
height: 400px; max-width: 600px; margin: 0 auto" ></div>
    <div id="container2" style="min-width: 310px; display: none;
height: 400px; max-width: 600px; margin: 0 auto"></div>
    <div id="container3" style="min-width: 310px; display: none;
height: 400px; max-width: 600px; margin: 0 auto"></div>
    <div id="container4" style="min-width: 310px; display: none;
height: 400px; max-width: 600px; margin: 0 auto"></div>
    <div id="container5" style="min-width: 310px; display: none;
height: 400px; max-width: 600px; margin: 0 auto"></div><br>

    <form action="/convocatoria" method="POST" style="display: none"
id="districte">
      <br>
      <select name="postejo">
      <option value="Badalona">Badalona</option>
      <option value="Ciutat Vella">Ciutat Vella</option>
      <option value="Eixample Dreta">Eixample Dreta</option>
      <option value="Eixample Esquerra">Eixample Esquerra</option>
      <option value="Gr&agrave;cia">Gr&agrave;cia</option>
      <option value="Horta-Guinard&oacute;">Horta-
Guinard&oacute;</option>
      <option value="Hospitalet de Llobregat">Hospitalet de
Llobregat</option>
      <option value="Les Corts">Les Corts</option>
      <option value="Nou Barris">Nou Barris</option>
      <option value="Sant Andreu">Sant Andreu</option>
      <option value="Sant Gervasi">Sant Gervasi</option>
      <option value="Sant Mart&iacute;">Sant Mart&iacute;</option>
      <option value="Santa Coloma">Santa Coloma</option>
      <option value="Sants-Montjuic">Sants-Montjuic</option>

      </select>  Selecciona el teu districte o
poblaci&oacute; del Barcelon&egrave;s
```

```
      <br ><br>
      <input type="number" max="14" min="9" step="0.01" name="sele"
value="" >   Introdueix la teva nota de
selectivitat  </input>
      <input type="submit" value="Actualitza"></input>

    </form>
    <form action="/convocatoria" method="POST" style="display: none"
id="comsele">
      <br>
        <input type="text" style="display: none;" id="comarcainput"
value='' name="postejo">
        <input type="number" step="0.01"  max="14" min="9" name="sele"
value="" >  Introdueix la teva nota de
selectivitat  </input>
        <input type="submit" value="Actualitza"></input>

    </form>

    <script>

     var titular ='{{cap}}';
     document.getElementById("container1").style.display='{{mostra}}';
     document.getElementById("container2").style.display='{{mostra}}';
     document.getElementById("container3").style.display='{{mostra}}';
     document.getElementById("container4").style.display='{{mostra}}';
     document.getElementById("container5").style.display='{{mostra}}';
     document.getElementById("titula").style.display='{{mostracap}}';
      document.getElementById("districte").style.display='{{mostradist
ricte}}';
    </script>


    <!--mapa catalunya-->
  <div class="mapWrapper">
      <div id="map"></div>
      <div id="text">
        <div id="comarcaName"><h1>Informaci&oacute;</h1></div>
        <div id="contentText"> Selecciona primer la teva comarca de
proced&egrave;ncia i introdueix la teva nota de selectivitat (entre 9
i 14)</div>
      </div>
  </div>

  </body>
</html>
```

# 3. Codi JavaScript

## 3.1 Highcharts

### 3.1.1 Gràfic de barres apilades

```
$(function () {
    var options = {
        chart: {
            type: 'bar',
          renderTo: 'container2'
        },
        title: {
            text: 'Nous alumnes que entren a l\'Etseib segons comarca'
        },
        xAxis: {
          title: {
            text: 'Comarca'
          },
            categories: ['Alt Camp', 'Alt Empordà', 'Alt Penedès',
'Alt Urgell', 'Alta Ribagorça', 'Anoia', 'Bages', 'Baix Camp', 'Baix
Ebre', 'Baix Empordà', 'Baix Llobregat', 'Baix Penedès', 'Berguedà',
'Cerdanya', 'Conca de Barberà', 'Garraf', 'Garrigues', 'Garrotxa',
'Gironès', 'La Selva', 'Maresme', 'Montsià', 'Noguera', 'Osona',
'Pallars Jussà', 'Pallars Sobirà', "Pla d'Urgell", "Pla de l'Estany",
"Ribera d'Ebre", 'Ripollès', 'Segarra', 'Segrià', 'Solsonès',
'Tarragonès', 'Terra Alta', 'Urgell', "Vall d'Aran", 'Vallès
Occidental', 'Vallès Oriental']

        },

        yAxis: {
            min: 0,
          stackLabels: {
            enabled: true,
            style: {
                fontWeight: 'bold',
                color:'gray'
            }
          },
            title: {
                text: 'Nous estudiants'
            }
        },
        legend: {
            reversed: true
        },
        plotOptions: {
            series: {
                stacking: 'normal',
            pointWidth: 20
            }
        },
        series: []
    };
    $.get('static/cat2.csv', function(data) {
```

```
      var lines = data.split('\n');  // linies del document csv

      $.each(lines, function(lineNo, line) {
          if (line == "") return; //fa que no aparegui la ultima linia
csv
            var items = line.split(','); //separa les celes
            var series = {
                data: []
            };
            $.each(items, function(itemNo, item) {
                if (itemNo == 0) {
                    series.name = item; //nom = nonvocatoria
                } else {
                    series.data.push(parseFloat(item)); //percentatge
                }
            });

          options.series.push(series);
      });
      // Crea el grafic
      var chart = new Highcharts.Chart(options);
    });
});
```

### 3.1.2 Gràfic de barres

```
$.ajaxSetup({ cache: false });
$.get('static/taula.csv', function(csv) {
    $('#container2').highcharts({
        chart: {
            type: 'column'},
        data: {
            csv: csv
        },
        title: {
            text: titular
        },
        yAxis: {
          allowDecimals: true,
            title: {
                text: 'Percentatge d\'aprovats'
            }
        },
      xAxis: {
            title: {
                text: 'Rang nota de selectivitat'
            }
        },
      tooltip: {
            formatter: function () {
            return '<b>' + this.series.name + '</b><br/>' +
                    this.point.y +'%' + ' ' +
this.point.name.toLowerCase();

            }
      }
    });

});
```

### 3.1.3 Gràfic de sectors

```
$.ajaxSetup({ cache: false });
$(function() {
   var options = {
      chart: {
          renderTo: 'container1',
          defaultSeriesType: 'pie',
        plotBackgroundColor: null,
          plotBorderWidth: null,
          plotShadow: false,
          type: 'pie'

      },
      title: {
         text: "\300lgebra"
      },
      plotOptions: {
         pie: {
             allowPointSelect: true,
             cursor: 'pointer',
           showInLegend: true,
             dataLabels: {
             enabled: true,
             color: '#000000',
             connectorColor: '#000000',
             formatter: function() {
                    return '<strong>'+ this.point.name +'</strong>:
'+ this.y +' %';
             }
             }
         }
      },
      colors: [
        '#7FFF00',
          '#4169E1',
        '#B22222',
        '#2F4F4F',
        '#00008B',
        '#FFFF00',
        '#F08080'

      ],
       series : []
   };

    $.get('static/quesitos/algebra.csv', function(data) {
      // Split the lines
      var lines = data.split('\n');
      var series = {
            data: []
      };

      // Iterate over the lines and add categories or series
      $.each(lines, function(lineNo,line) {
          if (line == "") return;
            var items = line.split(',');
            if (items[0] == 'Abandona'){
            corregit = items[0]
          }else{
            corregit = items[0].split(' ');
```

```
            corregit = corregit[0]+' convocatòria';
          }

            series.data.push({
            type:'pie',
            name: corregit,
            y:parseFloat(items[1])
            });
      });

      options.series.push(series);
      // Create the chart
      var chart = new Highcharts.Chart(options);
    });
});
```

## 3.2 El mapa de Catalunya

```
/**
     * On Map click show the information text
     * @return {[type]} [description]
     */
    function onMapClick(comarcaName, contentText, comarcaLink){
        if(onClick){
            //console.log(comarcaLink);
            if(newWindow){
                window.open(comarcaLink);
            }else{
                window.location=comarcaLink;
            }
        }else{

            $('#comarcaName').html('<h1>' + comarcaName + '</h1>');
            $('#contentText').html(contentText);
          if(comarcaName == 'Priorat' || comarcaName == 'Moianès'){
            container1.style.display='none'
            districte.style.display='none'
            comsele.style.display='none'
            container2.style.display='none'
            container3.style.display='none'
            container4.style.display='none'
            container5.style.display='none'
            document.getElementById("titula").innerHTML = "No existeix
cap precedent per a la comarca del "+comarcaName+".";
            21itular.style.display='block'

          }else if(comarcaName == 'Barcelonès'){
            districte.style.display='block'
            comsele.style.display='none'
            container1.style.display='none'
            container2.style.display='none'
            container3.style.display='none'
            container4.style.display='none'
            container5.style.display='none'
            21itular.style.display='none';
          }else{
            districte.style.display='none'
            comsele.style.display='block'
            container1.style.display='none'
            container2.style.display='none'
            container3.style.display='none'
            container4.style.display='none'
            container5.style.display='none'
            21itular.style.display='none';
            comarcainput.value=comarcaName
            }
      }
```

# 4. Codi CSS

## 4.1 Índex de l'aplicació

```
.boto {
    background-color: #4CAF50;
    border: none;
    color: white;
    padding: 30px 40px;
    text-align: center;
    text-decoration: none;
    display: inline-block;
    font-size: 20px;
    margin: 15px 15px;
    cursor: pointer;
    border-radius: 12px;
    position: relative;
    left: 50px;
    top: 10px;
}

.boto2 {background-color: #008CBA;}
.boto1{float:left;}

.boto3 {background-color: #f44336;}
.boto4 {background-color:     #FFD700}
.boto3{float:left;}

.container{
    border-radius: 25px;
    border: 2px solid #483D8B;
    background-image: url("banner.jpg");
    height: 150px;
}

h2{
    font-size: 40px;
    padding: 0px 0 10px;
    color: white;
}
p{
    background: "#D3D3D3";
    font-family: 'Helvetica Neue', Helvetica, Arial, sans-serif;
    text-align: center;
    color: #2F4F4F:
    font-size: 7px;

}
```

## 4.2 Mapa de Catalunya

```
body {
      background: #fff;
      font-family: 'Helvetica Neue', Helvetica, Arial, sans-serif;
      text-align: center;
      color: #2F4F4F
}
h1 {
      font-size: 25px;
      padding: 30px 0 10px;

}
h2 {
      font-size: 25px;
      padding: 30px 0 10px;
      color: #A9A9A9;
p {
    padding-bottom: 30px;
    color: maroon;
}

a {
      color: #66bbdd;
      text-decoration: none;
}
a:hover {
      text-decoration: underline;
}

#container {
      margin:20px auto;

}

.description {
      margin-left: auto;
      margin-right: auto;
      width: 99%;
}

.description strong {
      padding: 10px;
```