laSalle
UNIVERSITAT RAMON LLULL

Escola Tècnica Superior d'Enginyeria Electrònica i Informàtica La Salle

Treball Final de Màster

Máster Universitario en Ingenieria de Telecomunicaciones

Early Glover Churn

Nombre Alumno
Sergi Robles Lladó

Nombre Profesor Ponente
Lluís Formiga Fanals

ACTA DEL EXAMEN
DEL TRABAJO FINAL DE MÁSTER

Reunido el Tribunal calificador en la fecha indicada, el alumno

D. Sergi Robles Lladó

expuso su Trabajo Final de Máster, titulado:

Early Glover Churn

Acabada la exposición y contestadas por parte del alumno las objeciones formuladas por los Sres. miembros del tribunal, éste valoró dicho Trabajo con la calificación de

Barcelona,

VOCAL DEL TRIBUNAL VOCAL DEL TRIBUNAL

PRESIDENTE DEL TRIBUNAL

Resum

L'anàlisi del *churn* es pot entendre com un problema per predir i comprendre l'abandonament de l'ús d'un producte o servei. Diferents indústries, des de l'entreteniment fins a financeres, passant per proveïdors de *cloud* fan ús de plataformes digitals on els usuaris accedeixen als seus productes o serveis. L'ús d'aquestes plataformes condueix a deixar rastres de conducta. Aquests rastres es poden extreure per comprendre'ls millor, millorar el producte o servei i per predir el *churn*. En aquesta tesis, realitzarem l'anàlisi del *churn* sobre les dades reals dels repartidors (glovers) de l'empresa de *food delivery* Glovo, amb senyals com el comportament d'aquests a l'hora de repartir les ordres, comunicacions o ingressos. Després de fer el corresponent preprocessament de dades comparem un *decision tree*, un r*andom fores*t, un *gradient boosting*, un *XGBoost* i addicionalment una xarxa neuronal per a la predicció de *churn* dels glovers. Hem trobat que tots els models excloent el *decision tree* tenen un rendiment similar, d'aquests el r*andom fores*t aconsegueix un rendiment lleugerament superior als altres.

Resumen

El análisis del *churn* se puede entender como un problema para predecir y comprender el abandono del uso de un producto o servicio. Diferentes industrias, des del entretenimiento hasta las financieras, pasando por proveedores de *cloud* hacen uso de plataformas digitales donde los usuarios acceden a sus productos o servicios. El uso de estas plataformas conduce a dejar rastros de conducta. Estos rastros se pueden extraer para entenderlos mejor, para mejorar el producto o servicio o para predecir el *churn*. En esta tesis, realizaremos el análisis del *churn* sobre los datos reales de los repartidores (glovers) de la empresa de *food delivery* Glovo, con señales como el compartimiento de estos a la hora de repartir órdenes, comunicaciones o ingresos. Después de realizar el correspondiente preprocesamiento de datos se comparan un *decision tree*, un r*andom fores*t, un *gradient boosting*, un *XGBoost* y addicionalment una red neuronal para la predicción del churn de los glovers. Se ha encontrado que todos los modelos menos el *decision tree* tienen un rendimiento similar, de estos el *random forest* consigue un rendimiento ligeramente superior a los demás.

Abstract

Churn analysis can be understood as a problem to predict and understand the abandonment of the use of a product or service. Different industries, from entertainment to finance, to cloud providers, use digital platforms where users access their products or services. The use of these platforms leads to traces of behaviour. These traces can be extracted to better understand them, to improve the product or service, and to predict the churn. In this thesis, we will perform an analysis of the churn on the actual data of the couriers (glovers) of the food delivery company Glovo, with signals such as the behaviour of these when delivering orders, communications or income. After performing the corresponding data pre-processing, we compare a decision tree, a random forest, a gradient boosting, an *XGBoost* and additionally a neural network for the churn prediction of the glovers. We found that all models excluding the decision tree have a similar performance, of which the random forest achieves a slightly higher performance than the others.

Contingut

Índex de figures

Índex de taules

Índex d'equacions

Acrònims

LTV. *Live Time Value*.

SWE: *South West Europe*.

LATAM: *Latin America*.

EEMEA: *Eastern Europe, Middle East, and Africa*.

CFO: *Courier (Glover) First Order*.

CSM: *Customer Success Managers*.

SaaS: *Software-as-a-Service*.

XGBoost: *Extreme Gradient Boosting*.

MLP: *Multi Layer Perceptron*.

DWH: *Data Ware House*.

SQL: *Structured Query Language*.

ANN: *Artificial Neural Network*.

CRM: *Customer Relationship Management*.

1 Introducció

1.1 Motivació de la recerca

El *churn* és una de les mètriques i KPIs que s´han posat més de mode a les companyies que ofereixen productes o serveis a clients. La raó per la qual el *churn* es tant important es perquè afecta directament als beneficis de l'empresa. Per a cada client hi ha un cost d'adquisició, un cop els clients es converteixen en usuaris actius aquest cost es pot compensar al cap d'un període determinat, depenent de la formulació de les empreses del life time value (LTV) dels seus clients. Però si un client marxa abans d'arribar a aquest punt, després la companyia té una pèrdua a curt termini del cost d'adquisició i una a llarg termini pels beneficis que li podria haver generat aquest client. Aquest concepte s'aplica als serveis financers, com ara de banca o inversió, serveis de telecomunicacions, empreses d'entreteniment com Netflix i Apple o empreses de *food delivery* com Glovo.

Tot i les diferències entre els productes dels sectors mencionats amb anterioritat, aquests productes o serveis comparteixen algunes característiques comunes. Primer, el que fan i com es comporten els usuaris dins de les seves plataformes es pot traduir en esdeveniments, i aquestes en dades que ens poden donar informació de la seva experiència o procés. Per exemple, si un glover reparteix una ordre, podem estudiar quan temps tarda, si plou en aquell moment, quan cobra, etc. O si fan una inversió, podem mirar a quin tipus de negoci inverteixen o quin percentatge inverteixen dels seus estalvis. En segon lloc, el comportament d'aquests usuaris tal i com s'ha descrit anteriorment pot canviar en el temps. L'eficiència d'un glover o la capacitat d'un inversor per arriscar-se pot variar segons la temporada o etapa.

Aquest treball neix amb el propòsit d'intentar ajudar a Glovo a millorar el *churn* dels glovers, per tal d'optimitzar els costos d'adquisició i maximitzar el LTV dels repartidors.

1.2 Objectius i abast de la recerca

El *churn* es un tema molt complex on hi ha moltes tècniques i opinions. Sense voler fer un anàlisi complexa d'altres estudis, aquest treball es centra en predir el *churn* dels glovers a partir de dades històriques.

Per tant, es proposa aplicar tècniques d'aprenentatge artificial per tal de predir el *churn* dels glovers estudiats, on es compararan diferents tècniques d'intel·ligència artificial.

Amb aquestes especificacions, els objectius de la recerca són els següents:

1. **Definició del glover *churn.***
2. **Extracció i estructuració les dades històriques dels glovers.**
3. **Aplicació d'un preprocessament de dades.**
4. **Implementació i comparativa de diferents algorismes d'aprenentatge automàtic.**

1.3 Estructura del treball

Pel que fa a l'estructura del document, a continuació s'expliquen breument els diferents apartats:

- **Capítol 2**. Dóna una contextualització sobre què es el *churn* i defineix el glover *churn.*
- **Capítol 3**. Conté l'estat de l'art d'estudis existents sobre el *churn*, així com de les tècniques d'aprenentatge automàtic aplicades.
- **Capítol 4**. S'especifiquen l'entorn de desenvolupament així com l'obtenció i estructuració de les dades.
- **Capítol 5**. S'explica pas a pas el preprocessament que han rebut les dades.
- **Capítol 6**. Conté una preu explicació sobre la metodologia d'anàlisi de resultats així com la presentació d'aquests.
- **Capítol 7**. Compara, comenta i analitza els resultats obtinguts del capítol anterior.
- **Capítol 8**. Es fa un resum i es recullen les conclusions del projecte així com les línies de futur.

1.4 Breu introducció a Glovo

Glovo és una plataforma tecnològica que permet a l'usuari demanar el que vulgui de la seva ciutat. L'usuari té una app, amb una llista de comerços, botigues, locals i restaurants i un botó màgic per demanar el que sigui. L'aplicació es connecta amb un glover, un repartidor, que accepta la comanda i reparteix l'ordre.

La clau del seu èxit és aquest "el que sigui". Glovo s'ha diferenciat dels seus competidors perquè no només et porta l'hamburguesa, la pizza o les tallarines amb gambes fins al sofà de la teva casa, sinó que el glover també et compra els regals de Nadal o de Sant Valentí, et porta les claus de casa al teu marit si s'ha quedat al carrer o toca al timbre per a despertar al teu fill. Fins i tot et fa la compra si se t'ha fet tard al treball.

Tot i així, el 70% de les ordres a Espanya són només de menjar. Predominen, en aquest ordre, les hamburgueses, el sushi i la pizza. Glovo ha repartit en tot el món més de 10 milions d'ordres, té més de 10.000 establiments associats i més de tres milions d'usuaris en la seva aplicació. [1]

Figura 1-1. Glover.

La comunitat de glovers està formada per autònoms que disposen de vehicle i que s'apunten a la plataforma. L'empresa els forma, i aquests decideixen quan treballar o acceptar les ordres, de manera que ho poden alternar amb els seus treballs o altres activitats. Quan algú genera una ordre, els glovers reben una notificació i decideixen si se'n poden fer càrrec o no. A més, la companyia té una assegurança contractada per si els hi passés alguna cosa.

Per facilitar les operacions als països on ofereix serveis la companyia s'estructura de la següent manera:

- **SWE**. Espanya, Espanya, Portugal, Itàlia i Polònia.
- **LATAM**. Argentina, Costa Rica, República Dominicana, Equador, Guatemala, Hondures, Panamà i Perú.
- **EEMEA**. Ucraïna, Kazakhstan, Geòrgia, Romania, Croàcia, Sèrbia, França, Marroc, Kenya i Costa D'ivori.

L'empresa va arrencar al gener de 2015 amb una inversió inicial de 140.000 euros. Al 2019 va tancar una ronda de finançament de 150 milions d'euros, que li deixa una valoració superior als 1.000 milions d'euros. Des del seu naixement però, Glovo sempre ha registrat pèrdues degut al seu creixement exponencial: en quatre anys, ha arribat a 200 ciutats de 26 països d'Europa, Àfrica i Llatinoamèrica. El seu model de negoci, consisteix en cobrar al voltant d'un 30% de cada ordre, el que resulta insuficient per a dur a terme aquesta expansió. Segons els resultats del 2018, l'empresa va facturar 52 milions, un 268% més, però va multiplicar per vuit les seves pèrdues, fins als 42 milions. [2]

L'estratègia actual de Glovo es concentra en tres direccions:

- Reforçar els últims mercats on ha entrat, entre ells destaquen països de l'Europa de l'Est, com Ucranïa, Georgia i Kazakhstan.
- Invertir en tecnologia per millorar l'eficiència de l'aplicació.
- Apostar per les categories de supermercats i parafarmàcia.

2 Churn

2.1 Què es?

El *churn* succeeix quan un client o subscriptor deixa d'utilitzar els serveis d'una companyia. El *churn rate* és una mètrica crítica ja que resulta molt menys costós retenir clients existents que adquirir-ne de nous, ja que al adquirir has de conduir els *leads* a través del *funnel* de venta, utilitzant els recursos de marquèting i ventes durant aquest procés. D'altre banda, la retenció de clients es més rendible ja que ja has obtingut la seva confiança i fidelitat.

El *churn* d'un client impedeix el creixement d'una companyia, per això aquestes han de tenir un mètode definit per calcular-lo en un període de temps determinat. Al conèixer i supervisar el *churn rate*, les companyies estan equipades per determinar els percentatges d'èxit de la retenció de clients i identificar-ne les estratègies de millora.

Hi ha diferents maneres de calcular el *churn rate*, ja que pot representar el nombre total de clients perduts, el percentatge de clients perduts respecte el nombre total de clients de l'empresa, el valor d'oportunitats de compra o servei perduts o el percentatge de valor recurrent. Altres companyies el calculen respecte un període de temps determinat, com ara períodes trimestrals o exercicis fiscals. Un dels mètodes més utilitzats per calcular el *churn* es dividir el nombre total de clients que té una empresa al començament d'un període de temps determinat entre el nombre de clients perduts durant el mateix període de temps.

Existeixen una multitud de problemes que poden portar els clients a abandonar els serveis oferts per una empresa, però n'hi ha algunes que se'n consideren les principals. Com per exemple una mala atenció al client, un estudi va trobar que gairebé 9 de cada 10 clients que han abandonat un servei degut a un una mala experiència [3]. Actualment els clients exigeixen uns serveis i una experiència excepcional, i quan els clients no la troben a part d'accedir als competidors comparteixen la seva mala experiència a les xarxes socials. Per això una mala atenció al client pot afectar en que molts més clients facin *churn* a part del que ha tingut la mala experiència. Altres causes poden ser un mal procés d'*onboarding*, comunicacions de baixa qualitat, fidelització de marca, causes naturals que totes les empreses tenen, etc.

Hi ha una relació directe entre el LTV d'un client i l'habilitat per fer créixer un negoci. Com més alt sigui el *churn* d'una companyia menys possibilitats tindrà de créixer. Tot i tenir l'assessorament de la millors companyies de màrqueting, si una empresa té *un churn rate* alt la seva caixa patirà, ja que el cost d'adquisició de nous clients és molt elevat. Hi ha molts estudis sobre el costos de retenir clients versus adquirir-ne, i estudi rere estudi demostra que els costos d'adquisició superen amb consideració els de retenció. Generalment, les empreses gasten 7 vegades més en l'adquisició de clients que en la retenció, i el valor mig global d'un client perdut és de 243 dòlars [4]. D'aquí es pot veure la importància que té el *churn* pels negocis.

La capacitat per predir que un client té una probabilitat de churn elevada quan encara hi ha temps per fer alguna cosa al respecte, pot representar una gran font d'ingressos. A més de la

pèrdua directe d'ingressos que es deriva d'un client que abandona el producte o servei, pot ser que els costos d'adquisició no hagin estats coberts per les despeses fetes pel client fins al moment de *churn* (és a dir, adquirir aquest client ha estat una mala inversió).

Per tenir èxit a l'hora de retenir clients que d'altre manera abandonarien el servei, els *marketers* i els experts en retenció han de ser capaços de:

- Predir amb antelació quins clients faran *churn*.
- Saber i definir quines accions de màrqueting tindran major impacte en retenció.

2.2 Definició de glover *churn*

En aquest projecte ens centrarem en el glover *churn*, és a dir, quan un glover deixarà d'oferir els seus serveis com a repartidor (no reparteix més).

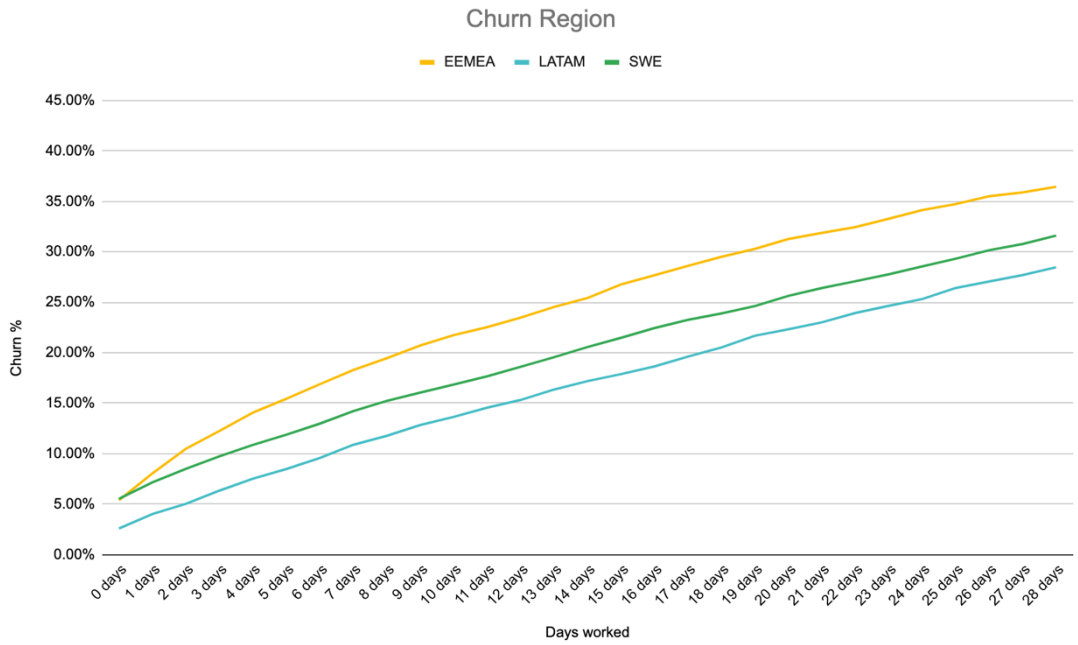

Figura 2-1. % Churn per dies treballats pels glovers.

El primer que farem per definir el *glover churn* serà escollir la regió d'estudi, si observem la Figura 2-1 veurem que la regió més afectada pel *churn* es EEMEA, on es concentren la majoria de nous països on Glovo ha obert mercat. Per aquest motiu i perquè l'estratègia actual es reforçar aquests mercats prendrem com a regió d'estudi EEMEA.

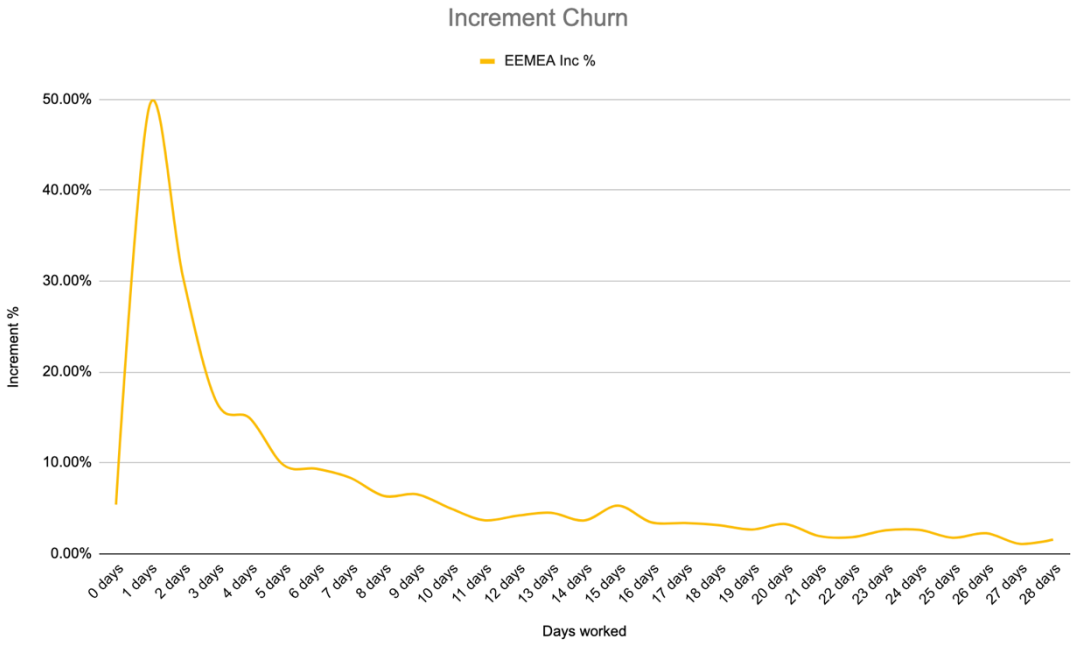

Figura 2-2. Increment Churn EEMEA.

Després de definir la regió toca escollir la finestra temporal d'estudi. La Figura 2-2 ens il·lustra l'increment en % del *churn* respecte els dies treballats pels glovers, aquí podem observar com als primers dies tenim un increment del creixement del *churn* molt alt però a mesura que passen els dies treballats aquest increment s'estabilitza. Com a conclusió podem extrapolar que reactivar als glovers que han fet *churn* durant els primers dies es clau per afavorir retencions futures. Es per això que agafarem com a regió d'estudi 5 dies on ja tenim gairebé un 15 % de *churn* (de cada 100 nous glovers adquirits 15 deixen de treballar en els seus primers 5 dies) i els increments es comencen a estancar.

Per tant, definim glover *churn* com aquells glovers que passats 5 dies després d´haver fet la seva primera ordre (CFO) no hagin repartit més. De les anteriors definicions arribem a la següent fórmula:

$$Glover\ churn\ = \frac{CFO\ 5d - Churned\ CFO\ 5d}{CFO\ 5d}$$

Equació 2-1. Glover *churn*.

On:

- $CFO\ 5d$. Glovers que han fet CFO fa 5 dies i han repartit algun cop al cap de 5 dies.
- $Churned\ CFO\ 5d$. Glovers que han fet CFO fa 5 dies i no han repartit més al cap de 5 dies.

3 Estat de l'art

En aquest capítol es repassen els estudis existents sobre el *churn*, les tècniques d'aprenentatge automàtic aplicades al *churn* i altres punts relacionats amb el tema. Tots aquests treballs serveixen d'inspiració i punt de partida d'aquest projecte

3.1 *Churn*

Aquelles empreses que monitoritzen constantment com els seus clients interactuen amb els seus productes, els animen a compartir opinions i resolen els seus problemes ràpidament tenen una major oportunitat d'obtenir una relació rendible. Si a part les companyies recopilen les dades referents al client, podran utilitzar-la per identificar patrons de comportament de possibles *churners*, segmentar aquests clients en risc i emprendre les accions oportunes per recuperar la seva confiança. Aquelles empreses que segueixen de manera proactiva el *churn* dels seus clients utilitzen analítiques predictives, que permeten preveure la probabilitat de resultats, esdeveniments o valors futurs mitjançant l'anàlisi de dades històriques i actuals. L'analítica predictiva utilitza diverses tècniques estadístiques, com la mineria de dades (reconeixement de patrons) i l'aprenentatge automàtic.

La principal virtut de l'aprenentatge automàtic és la creació de sistemes capaços de trobar patrons de dades, aprenent-ne sense programació explícita. En el context de la predicció del *churn*, ho podem veure com el comportament que tenen els clients i que indica si tenen una satisfacció menor o major en l'ús de productes/serveis d'una empresa.

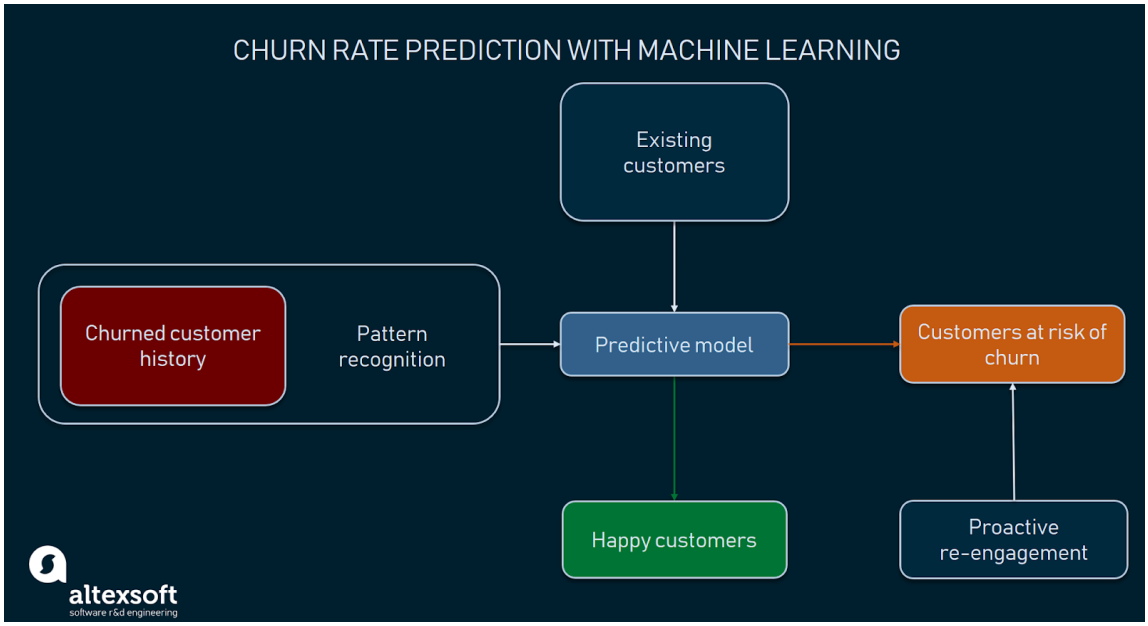

Figura 3-1. Predicció del *churn rate* aplicant aprenentatge automàtic.

Detectar clients en risc de *churn* pot ajudar a prendre mesures abans de que ja sigui massa tard. I aquí es on els algorismes d'aprenentatge automàtic poden fer un gran treball com veiem a la Figura 3-1, ja que poden mostrar alguns patrons de comportament compartits pels

clients que ja han abandonat l'empresa. Després comproven els comportaments dels clients actuals davant d'aquests patrons i donen una alarma si descobreixen possibles *churners* [5].

L'ús de modelatge predictiu de *churn* permet als experts en retenció definir quins clients han de ser contactats. És a dir, els empleats poden assegurar-se que parlen amb els clients adequats en el moment adequat. Per exemple, si un client mostra signes de risc de *churn*, probablement no serà un bon moment perquè l'equip de ventes el contacti per donar-li informació sobre serveis addicionals que li puguin interessar. Seria millor que l'acció la dugués a terme CSM perquè ajudin al client a tornar a veure el valor dels productes o serveis de que disposa. Igual que ventes, l'equip de màrqueting pot relacionar-se amb els clients de maneres diferents segons la seva probabilitat de *churn*, ja que per exemple un client que no està en risc de *churn* serà més bon candidat per participar en un cas d'estudi que un que ho està. L'estratègia d'interacció amb els clients s'ha de basar en l'ètica i en el sentit del temps, i l'aprenentatge automàtic pot aportar molta informació sobre aquesta estratègia.

Com a qualsevol tasca d'aprenentatge automàtic, el primer que els *data scientists* necessiten per treballar són dades. En funció de l'objectiu final, defineixen quines dades han d'utilitzar. A continuació, es preparen, es preprocessen i es transformen en dades intel·ligibles pels models d'aprenentatge automàtic. Trobar els mètodes adequats per entrenar i ajustar els models és una altre part important del seu treball. Un cop escollit el model que faci les prediccions amb la màxima precisió es pot posar en producció.

L'abast d'un projecte que duria a terme un per fer un *data scientist* per fer un model d'aprenentatge automàtic seria el següent:

- Comprensió del problema i objectiu final.
- Recopilació de dades.
- Preparació i preprocessament de dades.
- Modelat i proves.
- Desplegament i seguiment del model.

Normalment l'objectiu que tenen aquests models és el de classificació, determinar a quina classe o categoria pertany un punt de dades (en el nostre cas, el *churn* d'un client). Per a problemes de classificació, els experts utilitzen dades històriques amb etiquetes predefinides (*churner* / no-*churner*). Aquestes etiquetes són les que l'algorisme ha d'encertar quan li entren dades noves.

Aquest conjunt de dades històriques estan formades per *features* (variables), i aquestes segons l'expert en màrqueting i emprenedor Neil Patel [6] es poden dividir en quatre grups:

- ***Customer features***. Informació bàsica sobre el client (per exemple, l'edat, ingressos, valor de la seva casa, formació universitària, etc.).

- ***Support features***. Carecterístiques de les intereaccions del client amb els equips de suport (per exemple, nombre d'intereccions o contactes, temes de les preguntes, valoracions de satifacció, etc.).

- ***Usage features***. Com un client utilitza un producte o servei (per exemple, el nombre de vegades que inicia sessió, l'hora del dia en què s'utilitza activament un producte, les funcions o mòduls utilitzats, etc.)

- ***Contextual features***. Representen altre informació contextual d'un client.

El modelatge predictiu és l'aprenentatge de la relació entre observacions fetes durant un període (finestra) que finalitza abans d'un punt de temps específic i les prediccions que es fan sobre el període que comença després del mateix punt de temps. Tal i com veiem a la Figura 3-2 el *event history* del client es on es fan les observacions, i a la *performance window* es on predim els esdeveniments futurs (*churn* o no *churn*).

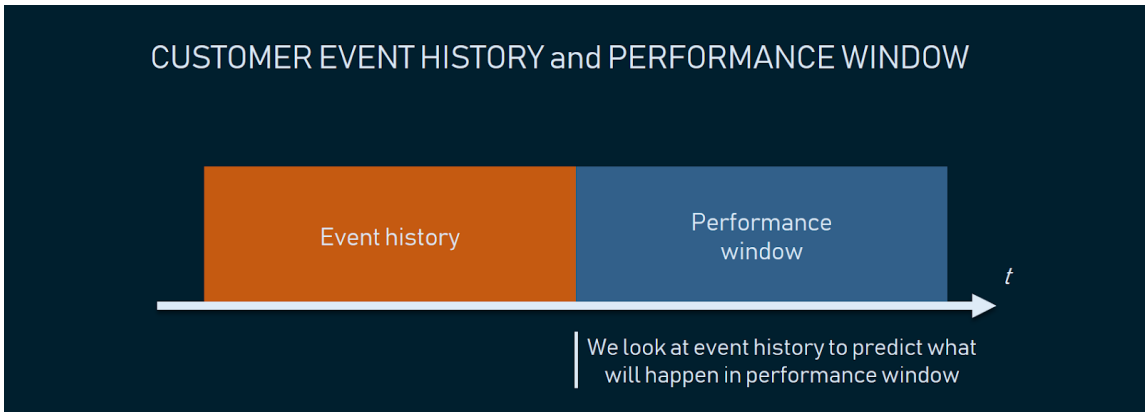

Figura 3-2. Event history del client i performance window.

L'enginyer de *Spotify*, Guilherme Dinis, en la seva tesi, va estudiar el comportament dels nous usuaris de *Spotify* registrats a un pla gratuït per definir si marxaven o continuaven actius durant la segona setmana després del registre. Va escollir la primera setmana com a *event history* i la segona com a *performance window*. Si els usuaris registrats en la primera setmana seguien escoltant música la segona se'ls classificava com a no *churners*. Aquesta selecció la va fer a través d'uns estudis interns que deien que al cap de dues setmanes la probabilitat de *churn* incrementava considerablement [7].

Com a conclusió podem dir que el *churn rate* és un indicador de la salut per a les empreses. La capacitat d'identificar clients que no estan satisfets amb les solucions proporcionades permet a les empreses aprendre sobre els punts dèbils del seu producte, pla de preus o problemes d'operació, així com les preferències i les expectatives dels clients per reduir de forma proactiva els seus motius de *churn*.

3.2 Aprenentatge automàtic

Per predir el *churn* s'utilitzen models clàssics d'aprenentatge automàtic, com per exemple *decisions tree*, *random forests* entre d'altres. Normalment s'utilitza un model més senzill com un *decision tree* o un *random forest* com a model de referència per després comparar les mètriques d'algorismes més complexos, com el *XGBoost* o les xarxes neuronals.

Un grup d'investigadors de la universitat de Virgínia va estudiar un conjunt dades lligades a l'eix temporal, com número de *logins*, número de contactes o comentaris per predir el *churn*

de client SaaS en una finestra de 3 mesos. Els autors van comparar el rendiment del model entre quatre algorismes de classificació, i el model *XGBoost* va obtenir els millors resultats en identificar quins clients havien fet *churn* i quins no. Segons l'article, la capacitat del model per definir quines són les funcions més significatives que representen com els clients utilitzen el programari SaaS pot ajudar als proveïdors a llançar campanyes de màrqueting més efectives orientades a clients potencials [8].

A continuació es detallen i comenten el conjunt d'algorismes que es faran servir en el projecte.

3.2.1 *Decision Tree*

Un arbre de decisions és una eina de suport a la decisió que utilitza un gràfic o model de decisions semblants a un arbre i les seves possibles conseqüències, inclosos els resultats d'esdeveniments casuals, els costos dels recursos i la utilitat. És una manera de mostrar un algorisme que només conté instruccions de control condicionals.

Una arbre de decisions és una estructura similar a un diagrama de flux en la qual cada node intern representa un "test" en un atribut (per exemple, si una cara o una creu apareix en una moneda, cada branca representa el resultat de la prova i cada node de la fulla representa una etiqueta de la classe (decisió pressa per calcular tots els atributs). Els camins d'arrel a la fulla representen les regles de classificació [11].

Es considera que els algorismes d'aprenentatge basat en arbres són un dels millors i més utilitzats en el món aprenentatge supervisat. Els mètodes basats en arbres permeten obtenir models predictius amb alta precisió, estabilitat i facilitat d'interpretació. A diferència dels models lineals, mapegen bé les relacions no lineals. També són adaptables a la resolució de qualsevol tipus de problema (classificació o regressió).

Tal i com veiem a la figura 3-3, aquests són alguns dels termes més habituals utilitzats amb arbres de decisió:

- ***Root node*** (node arrel). Representa tota la població o mostra, i es divideix entre dos o més conjunts homogenis.
- ***Splitting*** (dividir). Es un procés de dividir un node en dos o més subnodes.
- ***Decision node*** (node de decisió). Quan un subnode es divideix en altres subnodes, es diu node de decisió.
- ***Leaf/Terminal node*** (fulla/*node terminal*). Quan els nodes ja no es divideixen més es diuen fulla o node terminal.
- ***Pruning*** (poda). Quan eliminem els subnodes d'un node de decisió, aquest procés s'anomena poda.
- ***Branch/Sub-Tree*** (branca/subarbre). Una subsecció del arbre s'anomena branca o subarbre.

- ***Parent i Child Node*** (node parent i fill). Un node, que es divideix en subnodes es denomina node parental de subnodes, mentre que els subnodes són el fill del node progenitor.

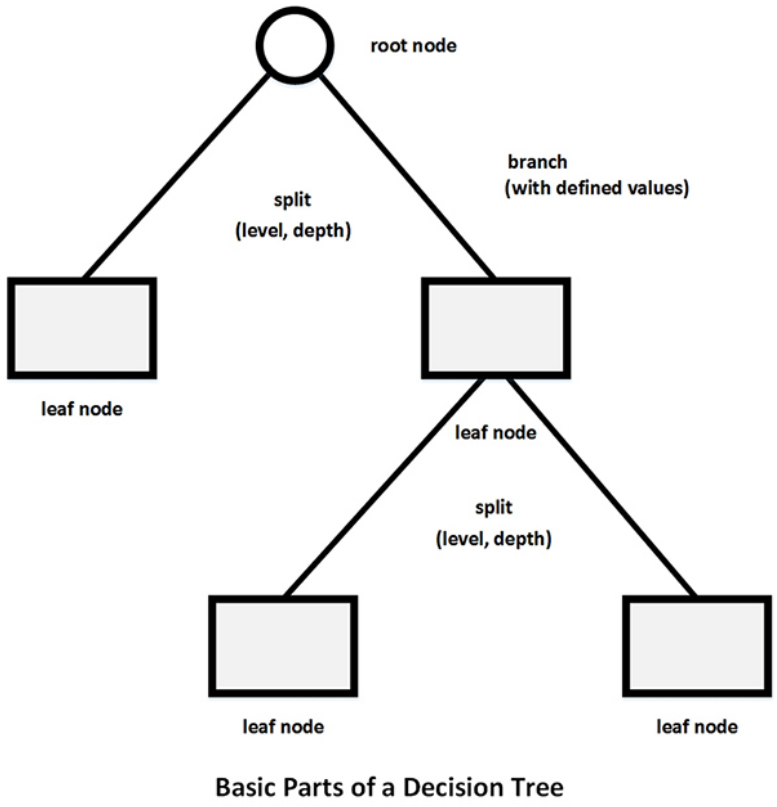

Figura 3-3. Parts d'un abre de decisions.

Un possible pseudocodi podria ser:

- Situar el millor atribut del *dataset* al *root node*.
- Dividir el conjunt d'entrenament en subconjunts. Els subconjunts s'han de fer de tal manera que cada subconjunt contingui dades amb el mateix valor per atributi i sigui el més homogeni possible.
- Repetir el primer i el segon pas per cada subconjunt fins trobar *leaf nodes* a totes les branques de l'arbre.

Alguns avantatges dels arbres de decisions són:

- Senzill i fàcil d'interpretar. Els arbres es poden visualitzar.
- Requereix poca preparació de dades, altres tècniques sovint requereixen normalització de dades.
- És capaç de manejar dades tant numèriques com categòriques.

I algunes desavantatges són:

- Es poden crear arbres sobrecomplexs que no generalitzen bé les dades (*overfitting*). Per evitar aquest problema, són necessaris mecanismes com la poda, establir el

nombre mínim de mostres necessàries en un node de fulla o establir la profunditat màxima de l'arbre.

- Els arbres de decisió poden ser inestables perquè petites variacions en les dades podrien originar un arbre completament diferent.
- Els arbres de decisió creen arbres esbiaixats si dominen algunes classes. Per tant, es recomana balancejar el conjunt de dades abans d'aplicar-li el model.

3.2.2 *Random Forest*

El *random forest* (bosc aleatori), tal i com el seu nom indica, consisteix en un gran nombre d'arbres de decisió individuals que funcionen com a conjunt. Cada arbre del bosc aleatori fa una predicció de la classe, i la classe amb més vots es converteix en la predicció del model tal i com veiem a la figura 3-4. La predicció acaba estant 1 ja que 6 arbres han predit 1´s mentrestant que només 3 han predit 0.

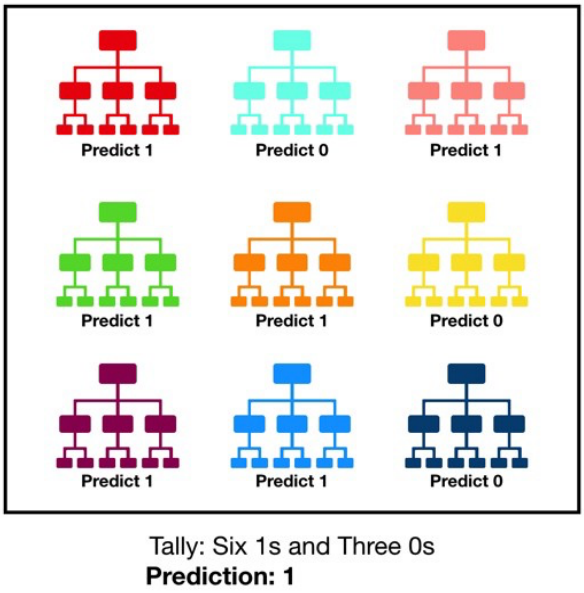

Figura 3-4. *Random forest* fet una predicció.

El concepte fonamental del *random forest* és simple però potent: la saviesa de les multituds. En ciències de dades, el motiu pel qual el model de bosc aleatori funciona tan bé és: una gran quantitat de models (arbres) relativament no correlacionats que funcionen com a comitè superaran així qualsevol model individual.

La baixa correlació entre models és la clau, els models no correlacionats poden produir prediccions de conjunts més exactes que qualsevol de les prediccions individuals. La raó d'aquest efecte és que els arbres es protegeixen els uns dels altres dels seus errors individuals (sempre que no s'equivoquin constantment en la mateixa direcció). Tot i que alguns arbres poden estar equivocats, molts altres arbres tindran raó, de manera que, en grup, els arbres poden moure's en la direcció correcta.

Els arbres de decisions són molt sensibles a les dades sobre les quals s'entrenen. Petits canvis en el conjunt d'entrenament poden donar lloc a estructures d'arbres significativament diferents. El bosc aleatori s'aprofita d'això permetent que cada arbre individual mostregi

aleatòriament el conjunt de dades amb reemplaçament, donant lloc a diferents arbres. Aquest procés es coneix com a *bagging*.

Si tenim una mostra de la mida N, estem donant a cada arbre un conjunt d'entrenament de mida N (tret que s'especifiqui el contrari). Però en lloc de les dades d'entrenament originals, agafem una mostra aleatòria de mida N amb reemplaçament. Per exemple, si les nostres dades de formació són [1, 2, 3, 4, 5, 6], podríem donar a un dels nostres arbres la llista següent [1, 2, 2, 3, 6, 6]. Podem observar que ambdues llistes tenen una longitud de sis i que "2" i "6" es repeteixen a les dades d'entrenament seleccionades aleatòriament que donem al nostre arbre (perquè mostregem amb reemplaçament).

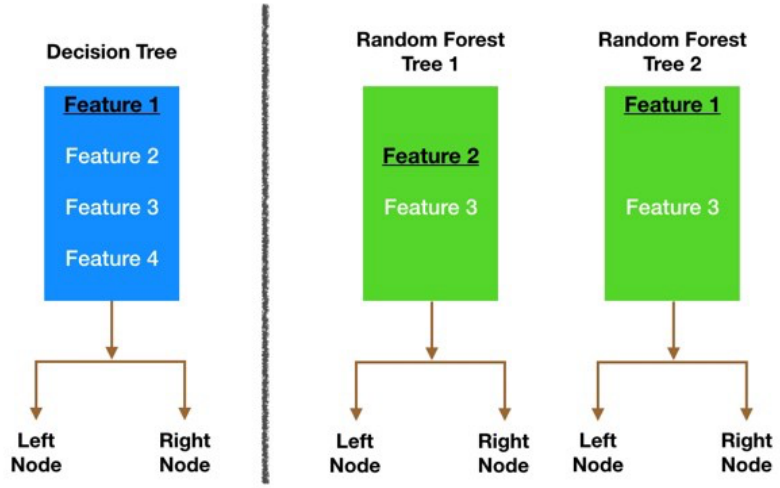

Figura 3-5. ***Node splitting decision tree vs random forest.***

Si mirem la figura 3-5, l'arbre de decisions tradicional (en blau) pot seleccionar entre les quatre *features* quan es decideix fer el node. En aquest cas, decideix anar amb la *feature* 1, ja que divideix les dades en els grups més separats possibles. Si ara mirem el *random forest* en verd, on en aquest exemple només s'examinen dos arbres del bosc, si fem un cop d'ull a *random forest Tree 1*, veiem que només pot considerar les *features* 2 i 3 (seleccionades a l'atzar) per la seva decisió de divisió de nodes. Sabem del nostre arbre de decisions tradicional (en blau) que la *feature* 1 és la millor característica per dividir, però l'arbre 1 no pot veure la *feature* 1, de manera que es veu obligat a anar amb la *feature* 2. L'arbre 2, en canvi, només pot veure les *features* 1 i 3 de manera que és capaç de triar la *feature* 1. Així doncs, un bosc aleatori, acaba amb arbres que no només s'entrenen en diferents conjunts de dades (gràcies al *bagging*) sinó que també utilitza *features* diferents per prendre decisions. I aquests fets creen arbres no correlacionats que es protegeixen mútuament dels seus errors [12].

Alguns avantatges dels *random forest* són:

- Mètode altament precís i robust a causa del nombre d'arbres de decisió que participen en el procés i requereix poca preparació de dades.
- No pateix *overfitting* ja que pren la mitjana de totes les prediccions, cosa que anul·la els biaixos (*biases*).
- Et poden donar una *feature importance* relativa.

Algunes desavantatges que presenta són:

- Són lents a l'hora de generar prediccions, ja que tots els arbres del bosc han de fer una predicció i després fer la votació final.
- Són més difícils d'interpretar si el comparem amb els arbres de decisions.

3.2.3 Gradient Boosting (GBM)

Abans d'explicar el model en detall cal entendre el concepte de *boosting*, que no es res més que un mètode per convertir aprenents dèbils en forts. En *boosting*, per tal de millorar cada nou arbre s'ajusta a una versió modificada del conjunt de dades original (figura 3-6). L'algorisme de GBM es pot explicar més fàcilment introduint primer l'algorisme d'AdaBoost. L'AdaBoost comença formant un arbre de decisió en què se li assigna a cada observació un pes igual, després d'avaluar el primer arbre, augmentem els pesos d'aquelles observacions més difícils de classificar i disminueix els pesos d'aquelles més fàcils de classificar. Per tant, el segon arbre creix d´aquestes dades ponderades. El nostre nou model és, per tant, *Tree 1* + *Tree 2*. A continuació, calculem l'error de classificació d'aquest nou model de conjunt de 2 arbres i fem créixer un tercer arbre per predir els residus, i repetim aquest procés per a un nombre d'iteracions. Els nous arbres ens ajuden a classificar les observacions que no estan ben classificades pels arbres anteriors, per tant, les prediccions del model final són la suma ponderada de les prediccions fetes pels models anteriors [13].

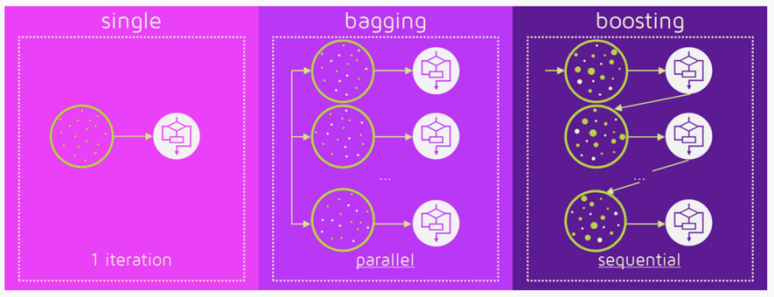

Figura 3-6. Bagging vs Boosting.

El *Gradient Boosting* entrena els models de forma gradual, additiva i seqüencial. La diferència principal entre l'*AdaBoost* i el GBM és com els dos algorismes identifiquen les mancances dels aprenents dèbils (per exemple, arbres de decisió). Mentre que el model *AdaBoost* identifica les mancances mitjançant els punts de dades de pes elevat, el GBM ho fa mitjançant gradients en la funció de pèrdua (*loss function*). El descens en gradients (*gradient descent*) és un algorisme d'optimització molt genèric capaç de trobar solucions òptimes a una àmplia gama de problemes. La idea general de descendència per gradients és ajustar els paràmetres iterativament per minimitzar una funció de cost. Suposem un esquiador de baixada que corre una cursa, una bona estratègia per vèncer seria prendre el camí amb el pendent més pronunciat. Això és exactament el que fa un descens de gradient (figura 3-7): mesura el gradient local de la funció de pèrdua (cost) per a un conjunt de paràmetres determinat (Θ) i fa passos en direcció al gradient descendent. Un cop el gradient és zero, hem arribat al mínim.

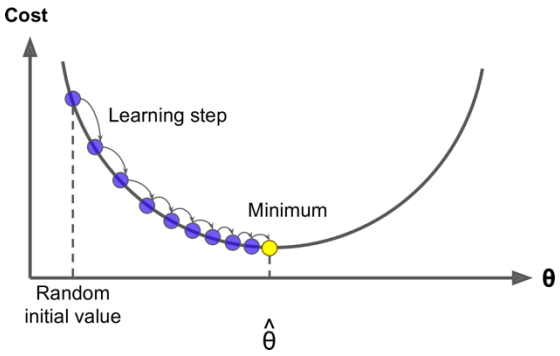

Figura 3-7. *Gradient descent*.

Es pot utilitzar un *gradient descent* en qualsevol funció de pèrdua que es pugui diferenciar. En conseqüència, això permet als GBM optimitzar diferents funcions de pèrdua segons es desitgi. Un paràmetre important en el descens de gradients és la mida dels passos que es determina en funció de la taxa d'aprenentatge (*learning rate*). Tal i com podem veure a la figura 3-8, si la taxa d'aprenentatge és massa petita, l'algoritme requereix moltes iteracions per trobar el mínim. D'altra banda, si el percentatge d'aprenentatge és massa alt, és possible que es salti el mínim i acabi més lluny del que va començar [14].

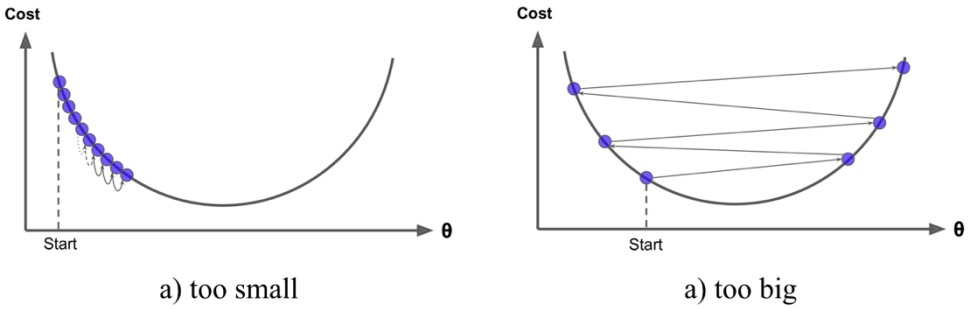

Figura 3-8. Comparació *learning rate*.

Els paràmetres generals d'aquest model de conjunt es poden dividir en tres categories:

- **Paràmetres específics de l'arbre**. Afecten cada arbre individual del model, els composen alguns dels que hem comentat al punt 3.2.1.
- **Paràmetres de *boosting***. Afecten l'operació de *boosting* del model, podrien ser per exemple la *learning rate* o el número d'abres.
- **Paràmetres diversos**. Altres paràmetres per al funcionament global.

Alguns dels seus avantatges són:

- Requereix poca preparació de dades, altres tècniques sovint requereixen normalització de dades.
- Ofereix molta flexibilitat, ja que pot optimitzar-se en diferents *loss functions*.

Algunes desavantatges que presenta són:

- Al minimitzar constantment tots els errors, això pot fer que es centri en intentar agafar els *outliers* i causar així *overfitting*. Per neutralitzar-la caldria fer *cross-validation*.
- Normalment es requereixen molts arbres, i això el pot fer un algorisme exhaustiu en temps i memòria.

- Són més difícils d'interpretar.

3.2.4 *Extreme Gradient Boosting* (*XGBoost*)

El *XGBoost* es un algorisme d'aprenentatge automàtic basat en arbres de decisions que utilitza un entorn de *gradient boosting*. En problemes de predicció que impliquen dades no estructurades (imatges, texts, etc.) les xarxes neuronals artificials solen superar a tots els altres algorisimes. Tantmateix, quan es tracta de dades estructurades de petit o mitjà volum el *XGBoost* es considera dels millors algorismes de l'actualitat. En la següent figura 3-9 podem veure l'evolució d'algorismes basats en arbres al llarg els anys.

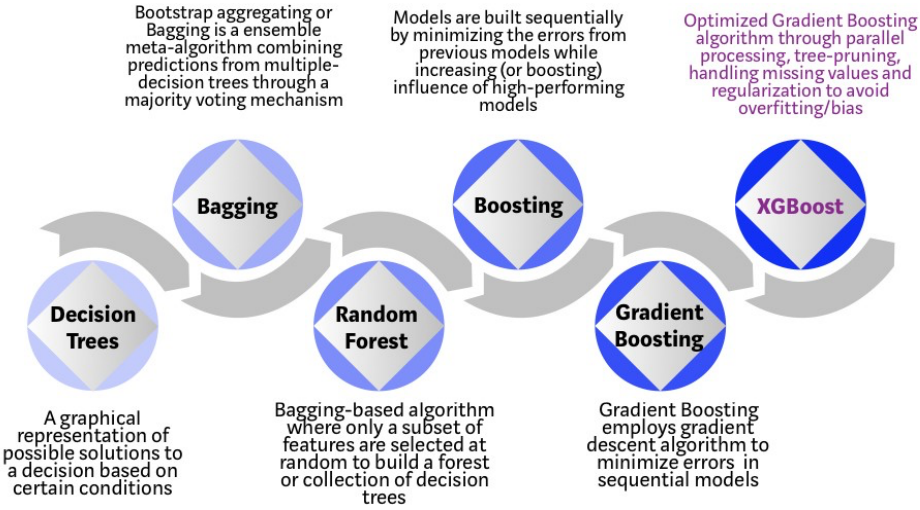

Figura 3-9. Evolució dels algorismes basats en arbres.

El *XGBoost* es va desenvolupar com a projecte de recerca a la Universitat de Washington [15], on Tianqi Chen i Carlos Guestrin el van presentar a la seva ponència al SIGKDD del 2016. Des de la seva introducció, aquest algorisme no només ha estat acreditat per guanyar nombroses competicions de *Kaggle*, sinó també per ser el fil conductor de la indústria. El *XGBoost* és una implementació ajustable i precisa dels *gradient boosting machines*, i ha demostrat impulsar els límits de la potència de càlcul per als algorismes d'arbres, ja que va ser construït i desenvolupat amb l'únic propòsit de potenciar el rendiment del model i la velocitat computacional. Concretament, es va dissenyar per explotar tots els recursos de memòria i maquinari per als algorismes de *boosting* basats en arbres.

El GBM i el *XGBoost* són tots dos mètodes que apliquen els principis de *boosting* als arbres de decisió fent servir l'arquitectura de *gradient descent*. Tot i això, tal i com veiem a la figura 3-10, el *XGBoost* supera al GBM amb millores algorítmiques i optimització de sistemes [16].

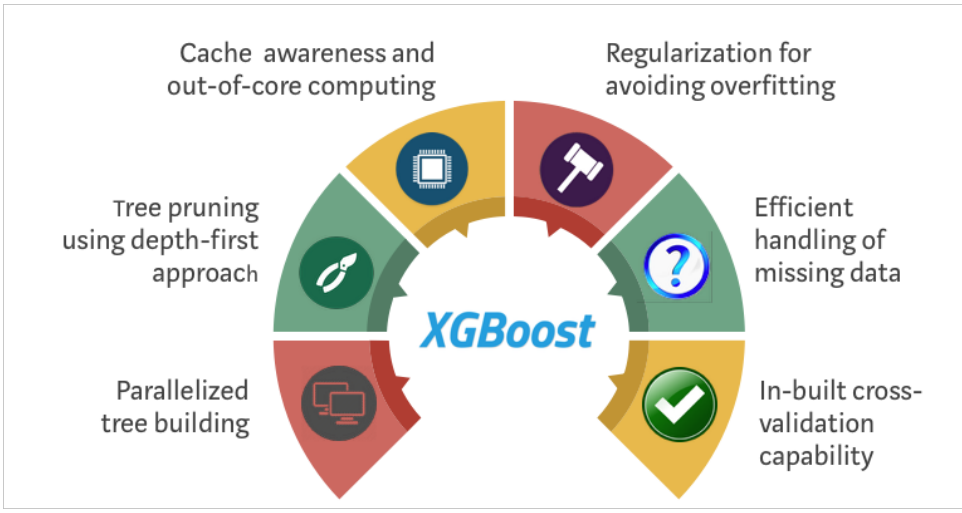

Figura 3-10. Optimitzacions del *XGBoost*.

Com a millores d'optimització del sistema tenim:

- **Paral·lelització**. *XGBoost* executa el procés de construcció d'arbres seqüencials mitjançant una implementació paral·lela.
- **Poda d'arbre**. El criteri d'aturada per a la divisió d'arbres en el marc GBM depèn del criteri de pèrdua negativa al punt de fraccionar. El *XGBoost* en canvi, utilitza el paràmetre *max_depth* i comença a podar arbres cap enrere. Aquest enfocament de primera profunditat millora significativament el rendiment computacional.
- **Optimització de *hardware***. Aquest algorisme ha estat dissenyat per fer un ús eficient dels recursos de *hardware*. Aquesta característica optimitza l'espai de disc disponible i maximitza el seu ús al gestionar conjunts de dades enormes que no entren en la memòria

I com a millores algorítmiques:

- **Regularització**. Penalitza models més complexos a través de la regularització *LASSO* (L1) i *Ridge* (L2) per evitar l'ajustament excessiu.
- **Falta de dades**. Aprèn automàticament el millor valor que falta en funció de la pèrdua d'entrenament.
- ***Cross-validation***: L'algorisme inclou un mètode de *cross-validation* integrat a cada iteració, eliminant la necessitat de programar explícitament aquesta cerca i d'especificar el nombre exacte d'iteracions necessàries en una sola execució.

Els paràmetres generals del *XGBoost* han estat dividits entre 3 pels seus autors:

- **Paràmetres generals**. Es relacionen amb el *booster* que estigui utilitzant, normalment un arbre de decisions. Dins d'aquests paràmetres també podem escollir els *threads*.
- **Paràmetres *booster***. Fan referència al booster que haguem escollit dins dels paràmetres generals. Podem escollir els paràmetres relacionats amb l'arbre de decisions així com els paràmetres de regularització.
- **Paràmetres de *learning task***. S'utilitzen per definir l'objectiu d'optimització de la mètrica a calcular en cada pas.

3.2.5 Xarxes neuronals

Una xarxa neuronal (ANN) és un model computacional que s'inspira en la forma en què les xarxes neuronals biològiques processen la informació en el cervell humà. Les xarxes neuronals artificials han generat molt d'interès en la indústria i en la investigació de l'aprenentatge automàtic gràcies a molts avenços en el reconeixement de veu, *computer vision* i processament de text. Una gran part del recent èxit de les xarxes neuronals s'explica per l'augment i disponibilitat de dades juntament amb una millora del poder computacional dels ordinadors, això ha superat els límits dels algorismes tradicionals d'aprenentatge automàtic. Aquesta situació s'il·lustra a la figura 3-11, on el rendiment de l'aprenentatge automàtic tradicional continua sent millor per a volums petits de dades però en major quantitat de dades les xarxes neuronals superen als algorismes tradicionals [17].

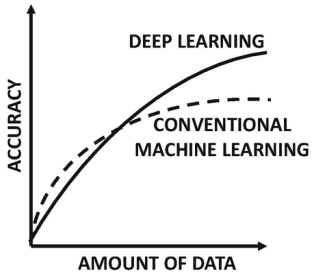

Figura 3-11. Precisió vs Quantitat de dades

La unitat bàsica de càlcul en una xarxa neuronal és la neurona, sovint també anomenada node o unitat. Rep l'entrada d'alguns altres nodes, o d'una font externa i calcula una sortida. Cada entrada té un pes (w) associat, que s'assigna a partir de la seva importància relativa a altres entrades. El node aplica una funció f a la suma ponderada de les seves entrades tal i com es mostra a la figura 3-12 [18].

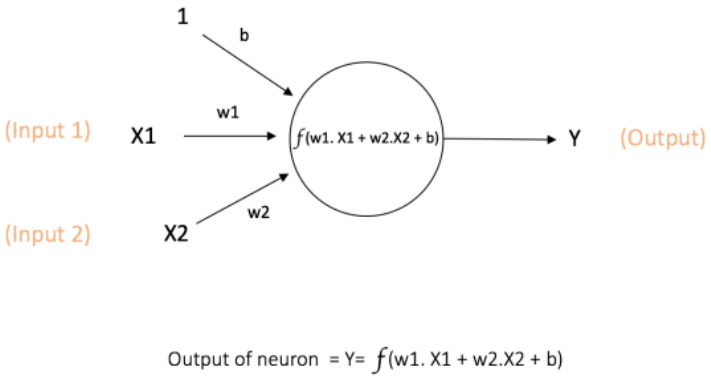

Figura 3-12. Neurona.

La xarxa anterior té les entrades numèriques X1 i X2 i té pesos w1 i w2 associats a aquestes entrades. Addicionalment, hi ha una altra entrada 1 amb el pes b anomenat *bias*. La funció del bias es proporcionar a cada node un valor constant entrenable (a més de les entrades normals que rep el node). La sortida Y de la neurona es calcula com es mostra a la figura 3-12, la funció f és una funció no lineal i rep el nom també de funció d'activació. L'objectiu de la funció d'activació és introduir la no linealitat a la sortida de la neurona. Això resulta important ja que la majoria de dades del món real no són lineals i es vol que les neurones aprenguin aquestes representacions Cada funció d'activació agafa un sol número i realitza una operació

matemàtica sobre d'aquest. Les funcions d'activació més rellevants són les següents (Figura 3-13):

- ***Sigmoid***. Agafa una entrada de valor real i la situa entre 0 i 1.
- ***TanH***. Agafa una entrada de valor real i la posa a l'abast de línterval [-1,1].
- ***ReLU***. Significa Rectified Linear Unit i agafa un valor real d'entrada i el llinda a zero, substituint els valors negatius per 0.

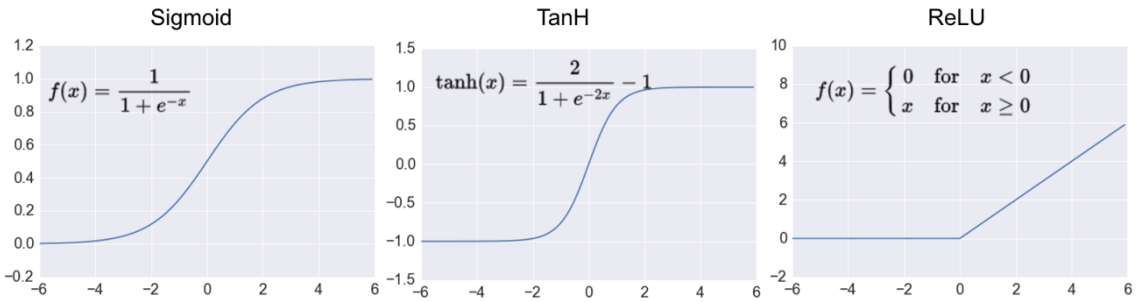

Figura 3-13. Funcions d'activació.

La *feedforward neural net*work va ser la primera i més simple xarxa neuronal artificial dissenyada. Conté múltiples neurones (nodes) disposades en capes, aquests nodes de les capes adjacents tenen connexions entre ells i totes aquestes connexions tenen pesos associats. Un exemple d'aquesta xarxa neuronal seria la que podem veure a la Figura 3-14:

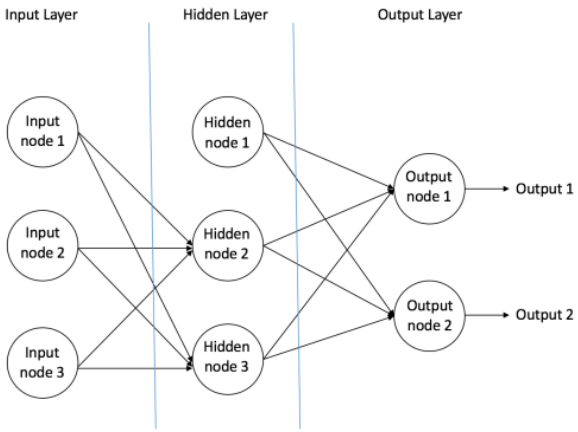

Figura 3-14. *Feedforward neural net*work.

Una *feedforward neural net*work pot constar de tres tipus de nodes:

- **Nodes d'entrada**. Proporcionen informació del món exterior a la xarxa i s'anomenen conjuntament com a capa d'entrada (*input layer*). No es realitza cap tipus de càlcul en aquests nodes, només transmeten la informació als nodes ocults.
- **Nodes ocults**. No tenen una connexió directa amb el món exterior (d'aquí el nom d'ocult). Realitzen càlculs i transfereixen informació des els nodes d'entrada als nodes de sortida. Una col·leció de nodes ocults forma una capa oculta (*hidden layer*). Pot tenir 0 o múltiples capes ocultes.

- **Nodes de sortida**. Es denominen col·lectivament capa de sortida (*output layer*) i són responsables de les computacions i la transferència d'informació de la xarxa al món exterior.

En una *feedforward neural net*work la informació es desplaça en una sola direcció (cap endavant) des dels nodes d'entrada, pels nodes ocults (si n'hi ha) i cap als nodes de sortida. Dos exemples de *feedforward net*wors són:

- ***Single Layer Perceptron***. Es la xarxa neuronal més simple i no conté cap capa oculta.
- ***Multi Layer Perceptron***. Conté una o més capes ocultes.

Una *Multi Layer Perceptron* (MLP) conté una o més capes ocultes (a part d'una entrada i una capa de sortida). Si bé un *Single Layer Perceptron* només pot apendre funcions lineals, un MLP pota apendre funcions no lineals. La figura 3-15 mostra un MLP amb una sola capa oculta, cal tenir en compte que totes les connexions tenen pesos associats, però en la figura només es mostren tres pesos (w0, w1 i w2).

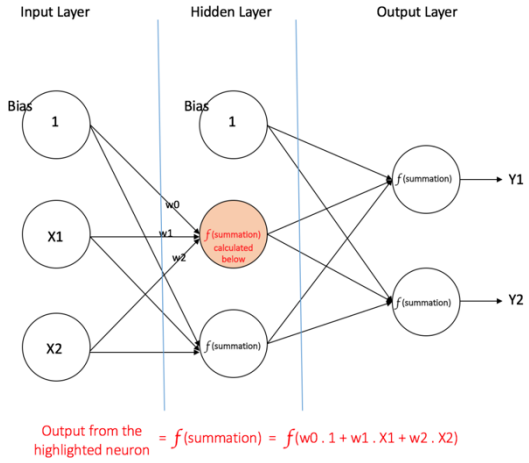

Figura 3-15. MLP amb una capa oculta.

La capa d'entrada té tres nodes, el primer es el *bias* que té un valor de 1 i els altres dos nodes prenen els valors de X1 i X2 com a entrada externa (que són valors numèrics segons el conjunt de dades d'entrada). Com ja s'ha comentat, no es fa cap càlcul a la capa d'entrada, de manera que les sortides dels nodes de la capa d'entrada són 1, X1 i X2 que s'introdueixen a la capa oculta. La capa oculta també té tres nodes amb el node *bias* inclòs, la sortida dels altres dos nodes de la capa oculta depenen de les sortides de la capa d'entrada i dels pesos associats a les connexions. A la figura 3-15 es mostra el càlcul de les sortida d'un dels nodes ocults ressaltat en taronja, aquestes sortides s'entreguen als nodes de la capa de sortida. La capa de sortida té dos nodes que agafen entrades de la capa oculta i realitzen càlculs similars al node ocult ressaltat. Els valors calculats (Y1 i Y2) com a resultat d'aquests càlculs actuen com a sortides del MLP.

Tenint en compte un conjunt de funcions X = (X1, X2, etc.) i una variable de sortida y, un MLP pota aprendre la relació entre les funcions i l'objectiu, ja sigui per una tasca de classificació o regressió.

El procés mitjançant el qual aprèn un MLP s'anomena algorisme de *Backpropagation* (*BackProp*), que és una de les diverses maneres en què es pot entrenar una xarxa neuronal artificial (ANN). En manera resumida, el *BackProp* corregeix a la ANN sempre que comet errors. Com ja s'ha comentat, una ANN consta de nodes distribuïts en diferents capes, i les connexions entre els nodes de capes adjacents tenen associats pesos. L'objectiu de l'aprenentatge automàtic és assignar els millors pesos possibles a aquestes connexions. Inicialment tots els pesos de les connexions s'assignen aleatòriament. Per a cada entrada del conjunt de dades d'entrenament, l'ANN està activada i s'observa la seva sortida. Aquesta sortida es compara amb al sortida desitjada que ja coneixem i l'error es propaga de nou a la capa anterior. Es pren nota del error i els pesos s'ajusten en conseqüència. Aquest procés es repeteix fins que l'error de sortida es troba per sota d'un llindar determinat. Un cop el *BackProp* acaba, es diu que tenim una ANN apresa que, considerem que està a punt per treballar amb dades noves [19].

A mode resum, els paràmetres més importantes de les ANN [20] són:

- **Loss function**. És la funció que s'utilitza per estimar el rendiment d'un model amb un conjunt específic de pesos sobre exemples de dades d'entrenament. Les funcions de pèrdues més escollides són l'entropia creuada per a problemes de classificació o l'error quadrat mitjà per problemes de regressió.

- ***Weight Initialitzation***. Es el procediment pel qual s'assignen els primers valors aleatoris inicials als pesos del model al començament del procés d'entrenament. Un dels més utilitzats és el *Glorot normal initializer*, que dibuixa mostres d'una distribució normal truncada centrada en 0.

- ***Optimizers***. Són algorismes o mètodes utilitzats per canviar els atributs de la ANN, com ara els pesos o la *learning rate*, per tal de reduir l'error. Un dels més utilitzats es l'Adam, que és un mètode de descens de gradients estocàstics que es basa en l'estimació adaptativa dels moments de primer i segon ordre. Aquesta mètode es dels més populars en l'actualitat ja que requereix poca memòria, és computacionalment molt eficient i s'adapta bé a problemes amb grans volums de dades.

- ***Learning rate***. Controla la quantitat d'actualització dels pesos del model i la rapidesa amb que aprèn el model de les dades d'entrenament.

- ***Epochs***. El procés d'entrenament s'ha de repetir moltes vegades fins que s'arribi a un conjunt de paràmetres prou bo. El nombre total d'aquestes iteracions s'anomena *epochs*.

Alguns avantatges de les ANN són:

- Treballa bé amb grans volums de dades a diferència dels algorismes d'aprenentatge automàtic tradicional.
- La informació s'emmagatzema a tota la xarxa, no en una base de dades. Per això si es perden dades no restringeix el funcionament de la ANN.
- Poden realitzar múltiples tasques en paral·lel sense afectar al rendiment del sistema.

I algunes desavantatges són:

- No tenim una explicació sobre el funcionament de la xarxa, és a dir, quan una ANN dóna un solució no sabem ni el per què ni com l'ha tret.
- No hi ha cap norma específica per determinar l'estructura de les ANN. S'obté l'estructura de la xarxa mitjançant experiència, i prova i error.

4 Entorn de desenvolupament

En primer lloc, s'explicarà el per què de l'ús de *Python* com a llenguatge de programació i les potencials eines que aquest ofereix per tractar dades i analitzar-les. A continuació, es farà una breu explicació sobre la font d'on s'han extret les dades i la forma que tenen.

4.1 Tria de *Python* com a llenguatge de progamació

Python es un dels llenguatges més utilitzats en el món. Es tracta d'un llenguatge de que té darrere una gran comunitat (sense ànim de lucre) que s'encarrega d'enriquir-lo (mitjançant la creació de noves llibreries, funcions, etc.) i millorar-ho.

La pàgina web Rebelion Research [21] ofereix una gràfica que ens permet analitzar quines són les ofertes de feina més demandades en funció dels paràmetres *machine learning* o *data science*, on es pot veure amb claredat que *Python* està en la primera posició just davant de *Java* i R. Tant *Python* com R són llenguatge que treballen molt bé amb matrius, per aquest motiu són tant populars entre els nous desenvolupadors.

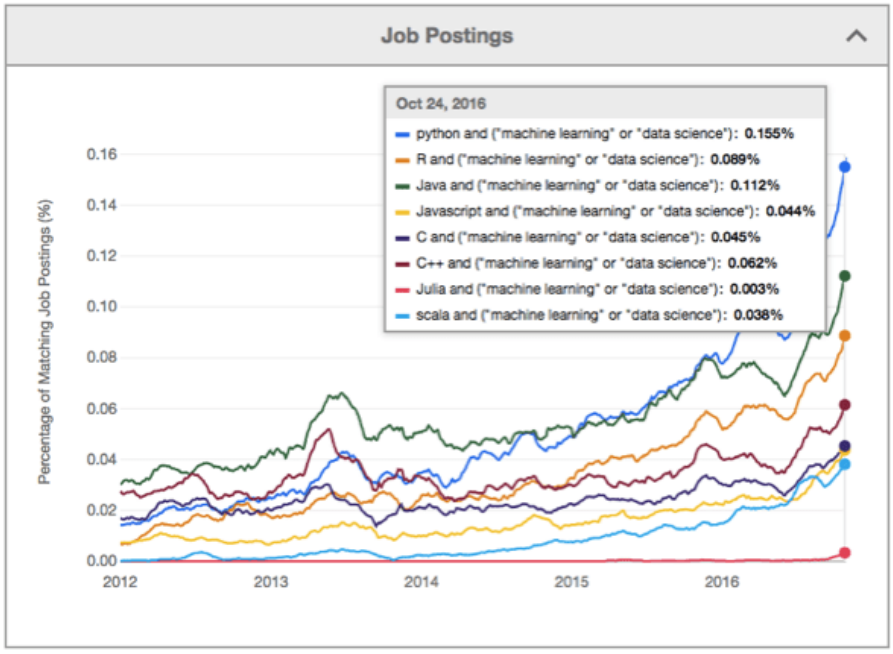

Figura 4-1. Ofertes de feina per llenguatge de programació.

S´ha escollit *Python* al ser el llenguatges de programació més utilitzat i valorat del món. A més a més, té un entorn de desenvolupament (IDE) molt complet gratuït anomenat *Jupyter* que s'executa des de *Anaconda-Navigator*. Aquest entorn a part de tenir una interfície gràfica molt atractiva i senzilla permet guardar les variables, i la visualització d'aquestes resulta excel·lent. A part, *Python* compte amb el ventall de llibreries més extens dedicades a l'aprenentatge automàtic.

4.2 Obtenció de les dades

Les dades utilitzades en aquest projecte s'han extret del DWH de Glovo. Un DWH és un sistema que agrupa dades de moltes fonts diferents dins d'una organització per després ser utilitzades com a font de *reporting* o anàlisis. Un DWH emmagatzema dades històriques sobre l'empresa, no emmagatzema informació actual, ni s'actualitza en temps real. A Glovo les dades crues primer van a una base de dades anomenada *LiveDB* i posteriorment cada nit es pugen i s'actualitzen a DWH en les seves taules pertinents.

Existeixen multitud de taules dins de DWH, com per exemple taules que fan referència a les ordres, als glovers, als *earnings*, als slots (espai temporal que reserva un glover per repartir), *cash* (hi ha països on es pot pagar en efectiu, fet que fa els glovers hagin de gestionar-lo, llavors es monitoritza quant efectiu tenen, quan han tornat, etc.) entre d'altres.

Per obtenir les dades de DWH s´han d'emprar scripts o *queries* de SQL, que es el llenguatge que es fa servir per comunicar-se amb bases de dades (extreure, eliminar o afegir dades).

4.2.1 Definició de blocs

Fer facilitar el procés d'obtenció de les variables que alimentaran el model, i degut a la complexitat que seria dissenyar una *query* que les agrupés totes s´han dividit aquestes en diferents blocs. Aquests blocs contenen conjunts de variables que comparteixen una temàtica comuna dels glovers. Per entrenar el model s'han agafat dades del *churn* dels glovers des de desembre del 2019 fins al mes d'abril del 2020, mentrestant que per la validació s'han agafat dades del mes de maig del 2020. A continuació s'exposen les definicions dels diferents blocs utilitzats.

4.2.1.1 Bloc 1: General

Aquest bloc fa referència a les característiques generals dels glovers, podem trobar variables geogràfiques, temporals o d'*onboarding*. A la Taula 4-1 podem veure el conjunt de variables que formen part del Bloc 1.

Taula 4-1. *Features* del Bloc 1.

Feature Name	Description	Type
city_code	City code	object
courier_id	Courier id	int64
country_code	Contry code	object
city_level_longevity_weeks	City longevity (from city's first order to the month of courier's last order)	int64
city_level_monthly_active_couriers	Number of active couriers (delivered at least one order)	int64

city_level_capacity_over_couriers	Total slots capacity divided by active couriers	float64
region	Region (EEMEA, LATAM and SWE)	object
first_order_id	First order id	int64
first_order_time	First order time	datetime64
last_order_id	Last order id	int64
last_order_time	Last order time	datetime64
first_order_month	Name of first order month y (e.g. november)	int64
first_order_week_day	Name of first order week day (e.g. friday)	object
first_order_hour	First order hour 24h clock (e.g. 22, 10)	int64
first_order_time_period	First order time period (hour >= 7 & hour < 12 then 'morning' else hour >= 12 & hour < 16 then 'lunch' else hour >= 18 & hour < 24 then 'evening' else 'night')	object
first_order_weekday_period	First week day period (weekday or weekend)	object
last_order_month	Name of last order month y (e.g. november)	int64
last_order_week_day	Name of last order week day (e.g. friday)	object
last_order_hour	Last order hour 24h clock (e.g. 22, 10)	int64
last_order_time_period	Last order time period (hour >= 7 & hour < 12 then 'morning' else hour >= 12 & hour < 16 then 'lunch' else hour >= 18 & hour < 24 then 'evening' else 'night')	object
last_order_weekday_period	Last order week day period (weekday or weekend)	object
churn	Churn (1 = churn / 0 = no churn)	int64
total_orders_during_study_period	Total orders done during study period	int64
total_time_during_study_period_days	Total days study period (first order to last order time)	int64
orders_per_day_ratio	Orders per day ratio	float
age	Courier Age	int64

vehicle	Vehicle	object
time_to_cfo	Time to cfo (onboarding to first order in hours)	float64
time_lead	Time lead (hour >= 7 & hour < 12 then 'morning' else hour >= 12 & hour < 16 then 'lunch' else hour >= 18 & hour < 24 then 'evening' else 'night')	object
time_to_onboarding	Time to onboarding (time lead to onboarding in hours)	float64
channel	Channel from where the leads comes from (e.g. facebook, google)	object
referred	Lead referred	int64

4.2.1.2 Bloc 2: *Slots*

Aquest bloc fa referència a les característiques dels *slots* treballats i reservats pels glovers. A la Taula 4-2 podem veure les variables que formen part:

Taula 4-2. *Features* del Bloc 2.

Feature Name	Description	Type
courier_id	Courier id	int64
churn	Churn (1 = churn / 0 = no churn)	int64
total_time_during_study_period_hours	Total hours study period (first order to last order time)	int64
slots_worked_between_first_and_last_order	Slots worked between first and last order	int64
slots_worked_before_first_order	Slots worked before first order	int64
kick_outs_before_first_order	Kick outs before first order	int64
kick_outs_between_first_and_last_order	Kick outs between first and last order	int64
total_orders_over_slots_ratio	Ratio total orders over slots (total orders / slots worked until last order)	float64
total_orders_over_time_ratio	Ratio total orders over time (total orders / total hours study period)	float64

slots_worked_between_first_and_last_over_time_ratio	Ratio slots worked over time (total slots worked / total hours study period)	float64
kick_outs_between_first_and_last_over_time_ratio	Ratio kick outs over time (total kick outs / total hours study period)	float64

4.2.1.3 Bloc 3: *Orders*

Aquest bloc fa referència a les característiques de les ordres repartides pels glovers. A la Taula 4-3 podem veure les variables que formen part:

Taula 4-3. *Features* del Bloc 3.

Feature Name	Description	Type
courier_id	Courier id	int64
churn	Churn (1 = churn / 0 = no churn)	int64
first_order_quiero	First order quiero type	int64
first_order_shipment	First order shipment type	int64
first_order_rain	First order with rain	int64
first_order_rush_bonus	First order with rush bonus	int64
first_order_adj_bonus	First order with adj bonus	int64
first_order_partner	First order partner type	int64
first_order_non_partner	First order non partner type	int64
first_order_groceries	First order non groceries type	int64
first_order_canceled	First order canceled	int64
first_order_number_of_assignments	First order number of assignments	int64
first_order_reassigned	First order reassigned	int64
last_order_quiero	Last order quiero type	int64
last_order_shipment	Last order shipment type	int64
last_order_rain	Last order with rain	int64
last_order_rush_bonus	Last order with rush bonus	int64

last_order_adj_bonus	Last order with adj bonus	int64
last_order_partner	Last order partner type	int64
last_order_non_partner	Last order non partner type	int64
last_order_groceries	Last order non groceries type	int64
last_order_canceled	Last order canceled	int64
last_order_number_of_assignments	Last order number of assignments	int64
last_order_reassigned	Last order reassigned	int64
avg_order_quiero	Average order quiero type	float64
avg_order_shipment	Average order shipment type	float64
avg_order_rain	Average order with rain	float64
avg_order_rush_bonus	Average order with rush bonus	float64
avg_order_adj_bonus	Average order with adj bonus	float64
avg_order_partner	Average order partner type	float64
avg_order_non_partner	Average order non partner type	float64
avg_order_groceries	Average order non groceries type	float64
avg_order_canceled	Average order canceled	float64
avg_order_number_of_assignments	Average order number of assignments	float64
avg_order_reassigned	Average order reassigned	float64

4.2.1.4 Bloc 4: *Distances & Times*

Aquest bloc fa referència a les distàncies i temps en que han recorregut i tardat per el conjunt d'ordres que han repartit els glovers. A la Taula 4-4 podem veure el conjunt de variables que formen part del Bloc 4.

Taula 4-4. *Features* del Bloc 4.

Feature Name	Description	Type
courier_id	Courier id	int64

churn	Churn (1 = churn / 0 = no churn)	int64
first_order_courier_delivery_time	First order courier delivery time (min)	float64
first_order_order_delivery_time	First order delivery time (min)	float64
first_order_waiting_time_at_pickup	First order waiting time at pickup (min)	float64
first_order_distance_glover_to_pickup	First order distance glover to pickup (km)	float64
first_order_distance_pickup_to_delivery	First order distance pickup to delivery (km)	float64
first_order_total_distance	First order total distance (km)	float64
first_order_average_speed	First order average speed (km/min)	float64
first_order_quick	First order delivery time < 30 min	int64
first_order_delayed	First order delivery time > 45 min & < 60 min	int64
first_order_highly_delayed	First order delivery time > 60 min	int64
last_order_courier_delivery_time	Last order courier delivery time (min)	float64
last_order_order_delivery_time	Last order delivery time (min)	float64
last_order_waiting_time_at_pickup	Last order waiting time at pickup (min)	float64
last_order_distance_glover_to_pickup	Last order distance glover to pickup (km)	float64
last_order_distance_pickup_to_delivery	Last order distance pickup to delivery (km)	float64
last_order_total_distance	Last order total distance (km)	float64
last_order_average_speed	Last order average speed (km/min)	float64
last_order_quick	Last order delivery time < 30 min	int64
last_order_delayed	Last order delivery time > 45 min & < 60 min	int64
last_order_highly_delayed	Last order delivery time > 60 min	int64
avg_courier_delivery_time	Average order courier delivery time (min)	float64
avg_order_order_delivery_time	Average order delivery time (min)	float64
avg_order_waiting_time_at_pickup	Average order waiting time at pickup (min)	float64
avg_distance_glover_to_pickup	Average order distance glover to pickup (km)	float64

avg_distance_pickup_to_delivery	Average order distance pickup to delivery (km)	float64
avg_total_distance	Average order total distance (km)	float64
avg_average_speed	Average order average speed (km/min)	float64
avg_order_quick	Average order delivery time < 30 min	float64
avg_order_delayed	Average order delivery time > 45 min & < 60 min	float64
avg_order_highly_delayed	Average order delivery time > 60 min	float64

4.2.1.5 Bloc 5: *Communications*

Aquest bloc fa referència a les característiques de les comunicacions que han dut a terme els glovers. A la Taula 4-5 podem veure les variables que formen part:

Taula 4-5. *Features* del Bloc 5.

Feature Name	Description	Type
courier_id	Courier id	int64
churn	Churn (1 = churn / 0 = no churn)	int64
first_order_contacted	First order contacted	int64
first_order_number_of_contacts_count	First order number of contacts count	int64
first_order_first_answer_duration	First order first answer duration	float64
first_order_chat_duration	First order chat duration (min)	float64
first_order_max_response_time	First order max response time (min)	float64
first_order_avg_response_time	First order average response time (min)	float64
last_order_contacted	Last order contacted	int64
last_order_number_of_contacts_count	Last order number of contacts count	int64
last_order_first_answer_duration	Last order Last answer duration	float64
last_order_chat_duration	Last order chat duration (min)	float64
last_order_max_response_time	Last order max response time (min)	float64
last_order_avg_response_time	Last order average response time (min)	float64

avg_order_contacted	Average order contacted	float64
avg_order_number_of_contacts_count	Average order number of contacts count	float64
avg_order_first_answer_duration	Average order Average answer duration	float64
avg_order_chat_duration	Average order chat duration (min)	float64
sum_order_chat_duration_minutes	Sum order chat duration (min)	float64
avg_order_max_response_time	Average order max response time (min)	float64
avg_order_response_time	Average order average response time (min)	float64

4.2.1.6 Bloc 6: *Cash & Earnings*

Aquest bloc fa referència a les característiques del *cash* i els *earnings* acumulats i guanyats pels glovers. A la Taula 4-6 podem veure les variables que formen part:

Taula 4-6. *Features* del Bloc 6.

Feature Name	Description	Type
courier_id	Courier id	int64
churn	Churn (1 = churn / 0 = no churn)	int64
first_order_courier_earnings_per_order	First order courier earnings	float64
first_order_change_in_cash_balance	First order change in cash balance	float64
first_order_change_in_customers_payment_method	First order change in customer payment method	int64
first_order_customer_payment_method_is_cash	First order customer payment method is cash	int64
first_order_courier_payment_method_is_cash	First order courier payment method is cash	int64
first_order_courier_payment_method_is_bankable	First order courier payment method is bankable	int64
first_order_courier_withdrew_cash_atm	First order courier withdrew cash atm	int64
last_order_courier_earnings_per_order	Last order courier earnings	float64
last_order_change_in_cash_balance	Last order change in cash balance	float64

last_order_change_in_customers_payment_method	Last order change in customer payment method	int64
last_order_customer_payment_method_is_cash	Last order customer payment method is cash	int64
last_order_courier_payment_method_is_cash	Last order courier payment method is cash	int64
last_order_courier_payment_method_is_bankable	Last order courier payment method is bankable	int64
last_order_courier_withdrew_cash_atm	Last order courier withdrew cash atm	int64
avg_courier_earnings_per_order	Average order courier earnings	float64
avg_order_change_in_balance_cash	Average order change in cash balance	float64
avg_order_change_in_customers_payment_method	Average order change in customer payment method	float64
avg_order_customer_payment_method_is_cash	Average order customer payment method is cash	float64
avg_order_courier_payment_method_is_cash	Average order courier payment method is cash	float64
avg_courier_payment_method_is_bankable	Average order courier payment method is bankable	float64
avg_order_courier_withdrew_cash_atm	Average order courier withdrew cash atm	float64

4.2.2 Estructura de les dades

Definir l'estructura de les dades era un repte ja que com veiem la Figura 4-2 cada glover pot tenir un número d'ordres diferents, el que complica crear una estructura uniforme per tots. El glover que va en bicicleta ha fet 5 ordres mentrestant que el que va amb motocicleta només n'ha fet 2, si volguéssim incloure informació de totes les ordres una a una acabaríem tenint un conjunt de dades diferents per cada glover el que acabaria resultant en una impossibilitat per construir un model d'aprenentatge automàtic.

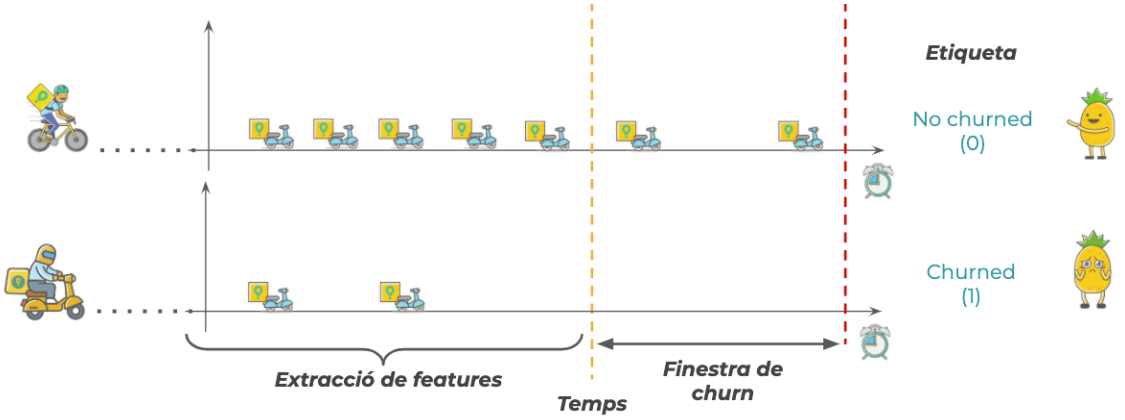

Figura 4-2. Glovers *churn.*

Per resoldre aquest problema el que s'ha fet es dividir la informació de les features entra la primera ordre, la última ordre i la mitjana del que ha passat entre totes les ordres entre la primera i la última. Tal i com veiem a la Figura 4-3, amb aquesta solució aconseguim una estructura uniforme per tots els glovers, ja que tots disposen d'almenys una ordre. Per exemple, si un glover només ha repartit una ordre aquesta serà la primera i l'última a la vegada, en canvi si un n´ha repartit 10 també tindrà primera i última.

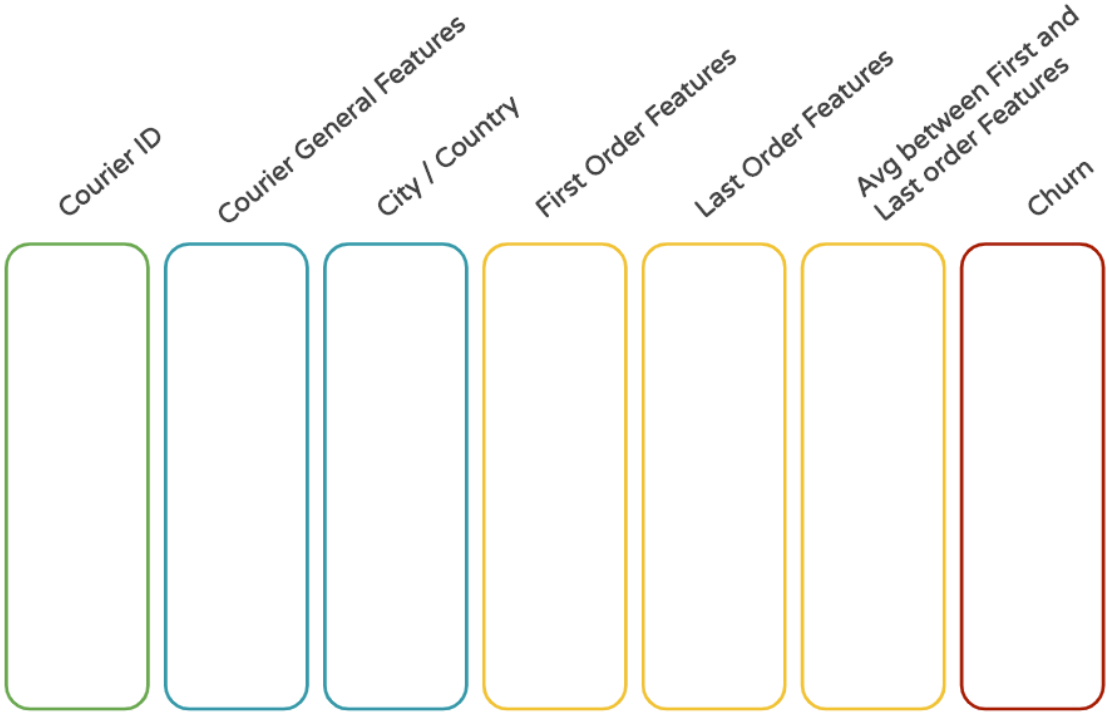

Figura 4-3. Estructura de les dades.

5 Preprocessament de dades

Els algorismes d'aprenentatge automàtic aprenen de les dades. És per això que es fonamental que tinguin les dades adequades per resoldre el problema desitjat. Encara que es disposin de bones dades, han d'estar en un format útil perquè l'algorisme aprengui correctament. Un cop seleccionades les dades (Bloc 4.2) es moment de preprocessar-les, que consisteix en dóna'ls-hi aquesta forma útil.

En aquest capítol s'explicaran algunes de les tècniques més utilitzades en preprocessament, a part de l'explicació es veurà la seva aplicació en els diferents blocs de variables comentades en l'anterior capítol. També veurem com queda el bloc final amb totes les variables individuals de cada bloc que han passat el preprocessmanet. Finalment aplicarem una tècnica de *feature selection* per reduir la dimensió del nostre *dataset* final i quedar-nos només amb aquelles variables significatives.

5.1 Gestió de duplicats

Tal i com indica el seu nom es tracta de gestionar/eliminar dades que estan repetides (duplicades).

A la taula 5-1 s'il·lustren les accions que s´han dut a terme en la gestió de duplicats de les dades.

Taula 5-1. Gestió de duplicats.

Block	Feature Name	Number Of Duplicates	Type
1	courier_id	31	int64
5	courier_id	10	int64
6	courier_id	3	int64

5.2 *Missing values*

Hi pot haver una gran quantitat de raons per a les quals trobem valors que falten: des d'errors humans durant l'entrada de dades, lectures incorrectes de sensors, fins a errors de *software* al *pipeline* de processament de dades.

Si trobem algunes *features* que els hi falten dades (per exemple, el 5% dels clients no especifiquen la seva edat), s´ha de decidir si es vol ignorar aquesta dades, aquesta *feature*, emplenar els valors que falten (per exemple, amb l'edat mitjana), o fer un model amb que inclogui aquesta *feature* i un altre que no per veure el que passa.

No hi ha una solució única per arreglar aquest problema, s'ha d'estudiar cas per cas individualment i després escollir la tècnica que es consideri millor.

A la taula 5-2 s'il·lustren les accions que s´han dut a terme per emplenar amb alguna dada aquelles variables que els hi faltaven valors.

Taula 5-2. Missing values.

Block	Feature Name	Filled	Type
1	time_to_onboarding	median	float64
1	channel	"organic"	object
2	total_orders_over_slots_ratio	1	float64
4	first_order_courier_delivery_time	avg_courier_delivery_time	float64
4	first_order_order_delivery_time	avg_order_order_delivery_time	float64
4	first_order_average_speed	avg_average_speed	float64
4	last_order_courier_delivery_time	avg_courier_delivery_time	float64
4	last_order_order_delivery_time	avg_order_order_delivery_time	float64
4	last_order_average_speed	avg_average_speed	float64

5.3 Categorical encoding

Les dades categòriques són habituals en molts problemes de ciències de dades i d'aprenentatge automàtic, però solen ser més difícils de tractar que les dades numèriques. En particular, moltes algorismes d'aprenentatge automàtic requereixen que la seva entrada sigui numèrica i, per tant, les *features* categòriques s´han de transformar a numèriques abans d'utilitzar qualsevol d'aquest tipus d'algorisme.

Una de les maneres més habituals de fer aquesta transformació es codificar amb un one-hot la *feature*, especialment quan no existeix una ordenació natural entre les categories (per exemple, una *feature* "*City*" amb noms de ciutats com "Barcelona", "Londres", "Berlín", etc.). Per cada valor únic d'una *feature* es crea una nova *feature* (és a dir "*City*_Barcelona") on el valor és 1 si per aquesta instància la *feature* original pren aquest valor i 0 en cas contrari [9].

S'ha aplicat aquesta tècnica de *one-hot encoding* a les variables categòriques amb la variant que el primer valor no l´hem convertit en una nova *feature*, ja que quan la resta de noves *features* estan desactivades vol dir que la que hem eliminat està activada.

A la taula 5-3 s'il·lustren les features que se´ls hi ha aplicat *one-hot encoding.*

Taula 5-3. *One-hot encoding.*

Block	Feature Name	Extra Features	Type
1	country_code	9	object
1	first_order_week_day	6	object
1	first_order_time_period	2	object
1	first_order_weekday_period	1	object
1	last_order_week_day	6	object
1	last_order_time_period	2	object
1	last_order_weekday_period	1	object
1	vehicle	3	object
1	time_lead	2	object
1	channel	29	object

5.4 Correlacions

Hi pot haver la possibilitat de que existeixen relacions complexes i desconegudes entre les *features* que conformen un *dataset*, per exemple:

- Una variable pot causar o dependre dels valors d'una altre variable.
- Una variable podria estar associada lleugerament a una altre.
- Dues variables podrien dependre d'una tercera variable desconeguda.

Pot ser de gran utilitat en l'anàlisi i modelatge de dades entendre millor les relacions entre variables. La relació estadística entre dues variables es coneix com la seva correlació. Una correlació pot ser positiva, és a dir, ambdues variables es mouen en la mateixa direcció o negativa, quan el valor d'una variable augmenta la de l'altre disminueix. La correlació també pot ser neutre o zero, és a dir, que les variables no tenen relació.

El rendiment d'alguns algorismes es pot deterior si dues o més variables estan estretament relacionades, anomenat com multicol·linealitat. Un exemple podria és la regressió lineal, on s'hauria d'eliminar una de les variables correlacionades per tal de millorar la precisió del model.

Per avaluar les correlacions del nostre data set em fet servir la correlació d'Sperman, que no assumeix una relació lineal entre les variable sinó una relació montònica. El coeficient de correlació es mou entre -1 i 1, essent 1 el màxim de correlació i -1 el mínim. Per aquest projecte hem eliminat totes aquelles variables que entre elles tenen una correlació absoluta superior al 0,7, de les dues ens hem quedat aquella que té mes correlació absoluta amb la variable *churn*.

Amb una matriu de correlacions podem veure d'una manera gràfica les relacions que existeixen entre les variables. A la diagonal sempre tenim 1's ja que una variable correlacionada amb ella mateixa es 1, però als altres extrems podrem apreciar les correlacions que es tenen entre les diferents variables.

A la Figura 5-1 podem veure la matriu de correlacions pel Bloc 1, d'on s'han eliminat 4 variables.

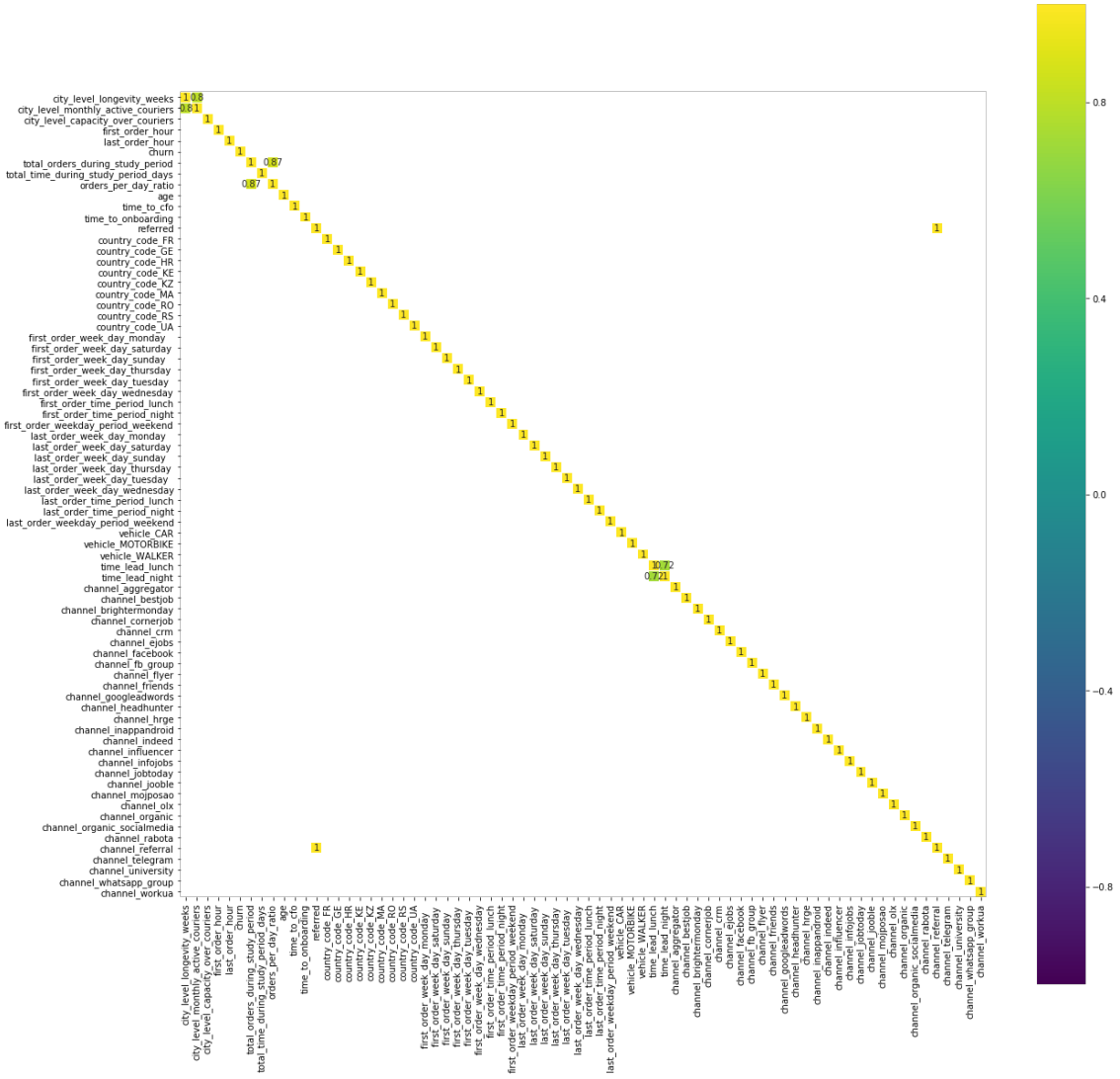

Figura 5-1. Matriu de correlacions del Bloc 1.

A la Figura 5-2 podem veure la matriu de correlacions pel Bloc 2, d'on s'ha eliminat 1 *feature*.

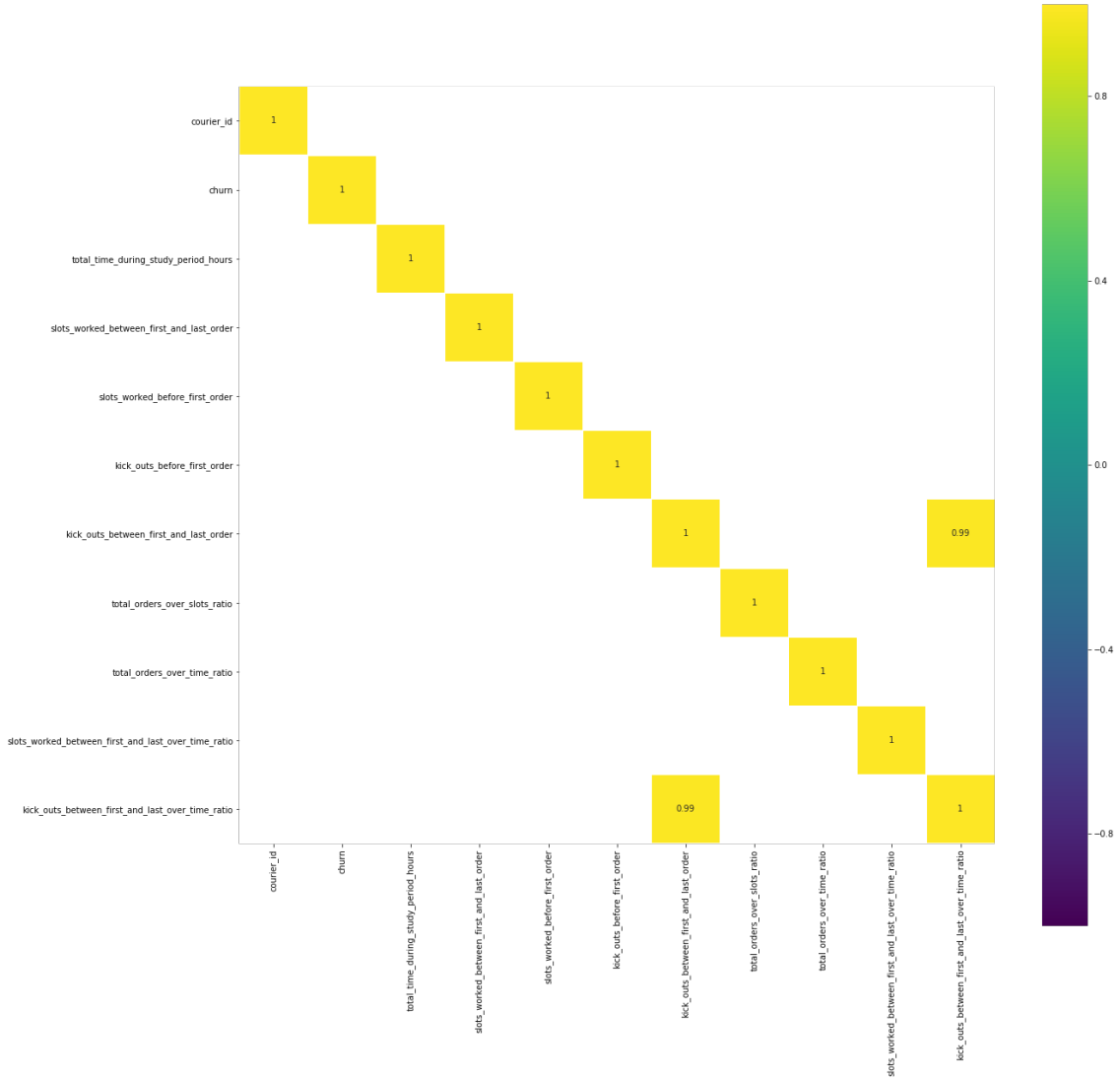

Figura 5-2. Matriu de correlacions del Bloc 2.

A la Figura 5-3 podem veure la matriu de correlacions pel Bloc 3, d'on s'han eliminat 7 *features*.

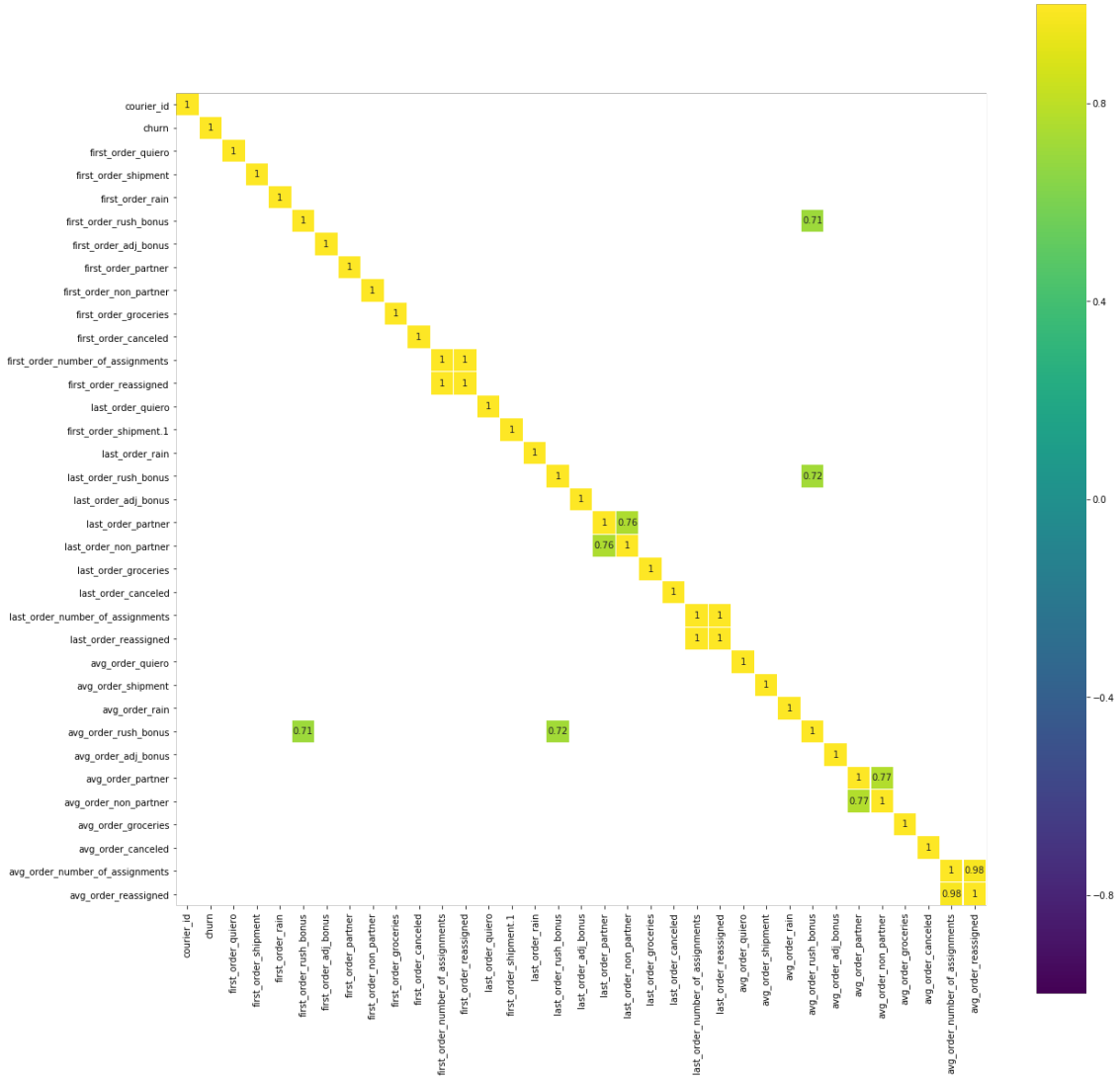

Figura 5-3. Matriu de correlacions del Bloc 3.

A la Figura 5-4 podem veure la matriu de correlacions pel Bloc 4, d'on s'han eliminat 8 *features*.

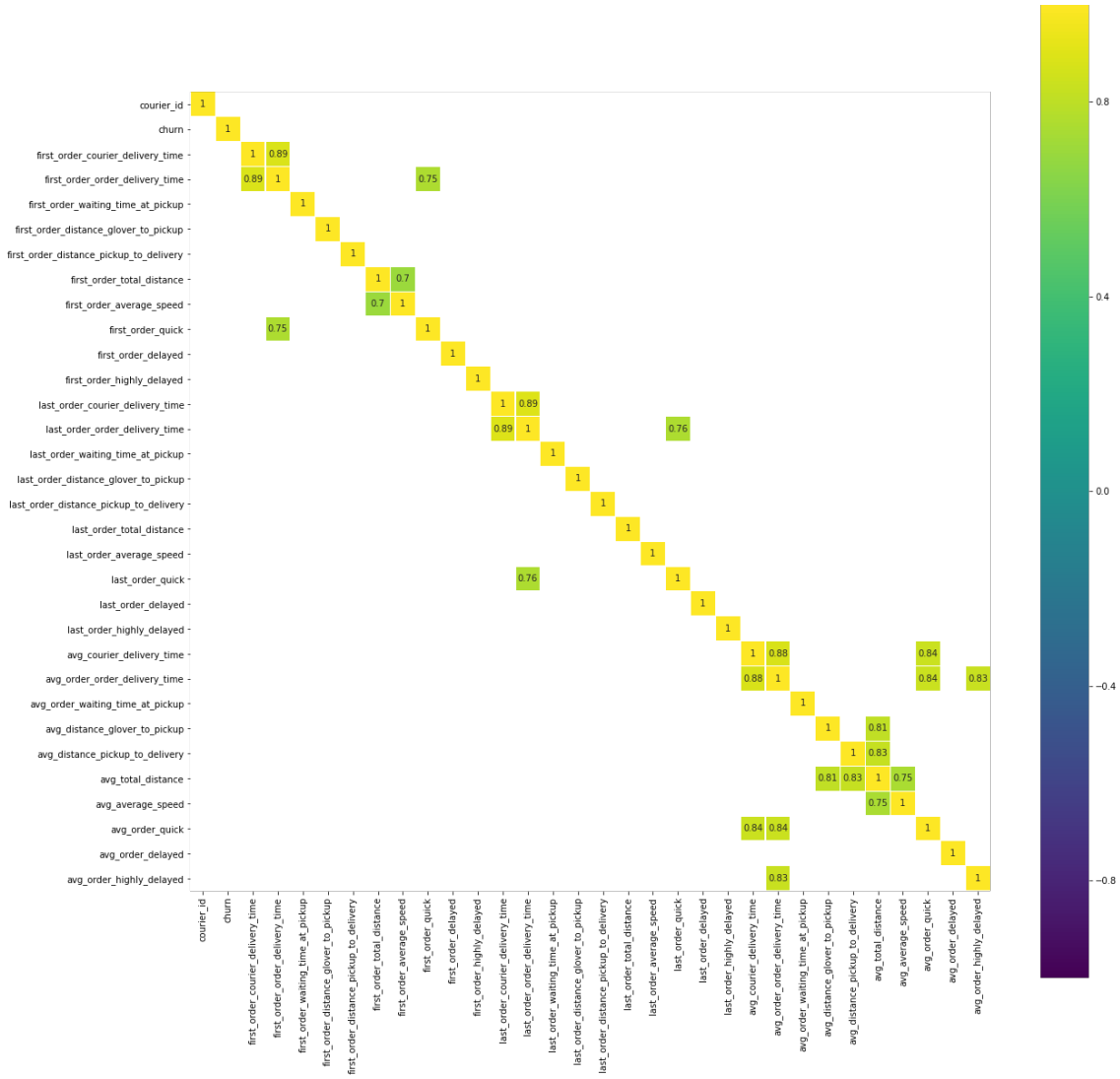

Figura 5-4. Matriu de correlacions del Bloc 4.

A la Figura 5-5 podem veure la matriu de correlacions pel Bloc 5, d'on s'han eliminat 16 *features*.

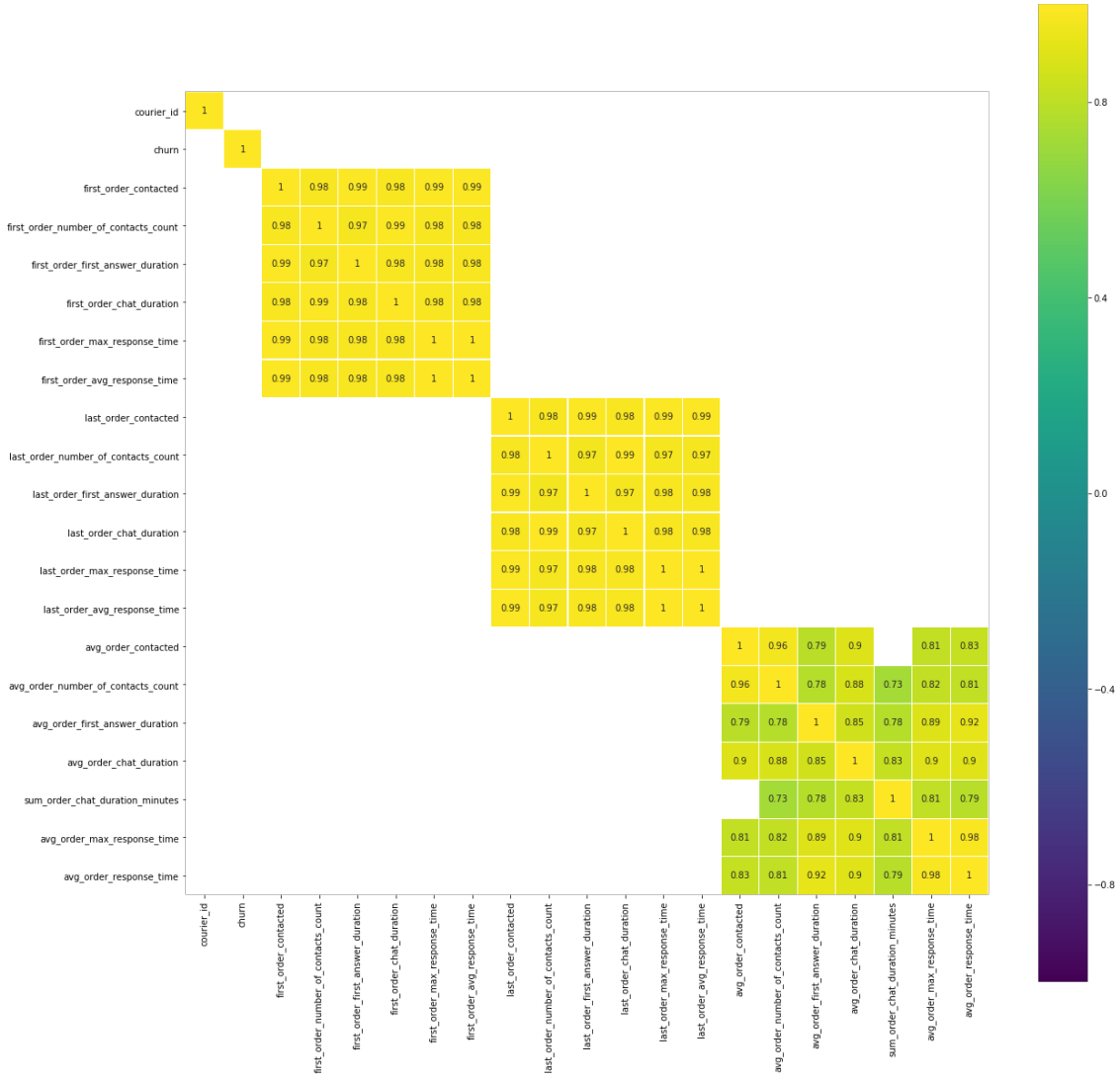

Figura 5-5. Matriu de correlacions del Bloc 5.

A la Figura 5-6 podem veure la matriu de correlacions pel Bloc 6, d'on s'han eliminat 4 *features*.

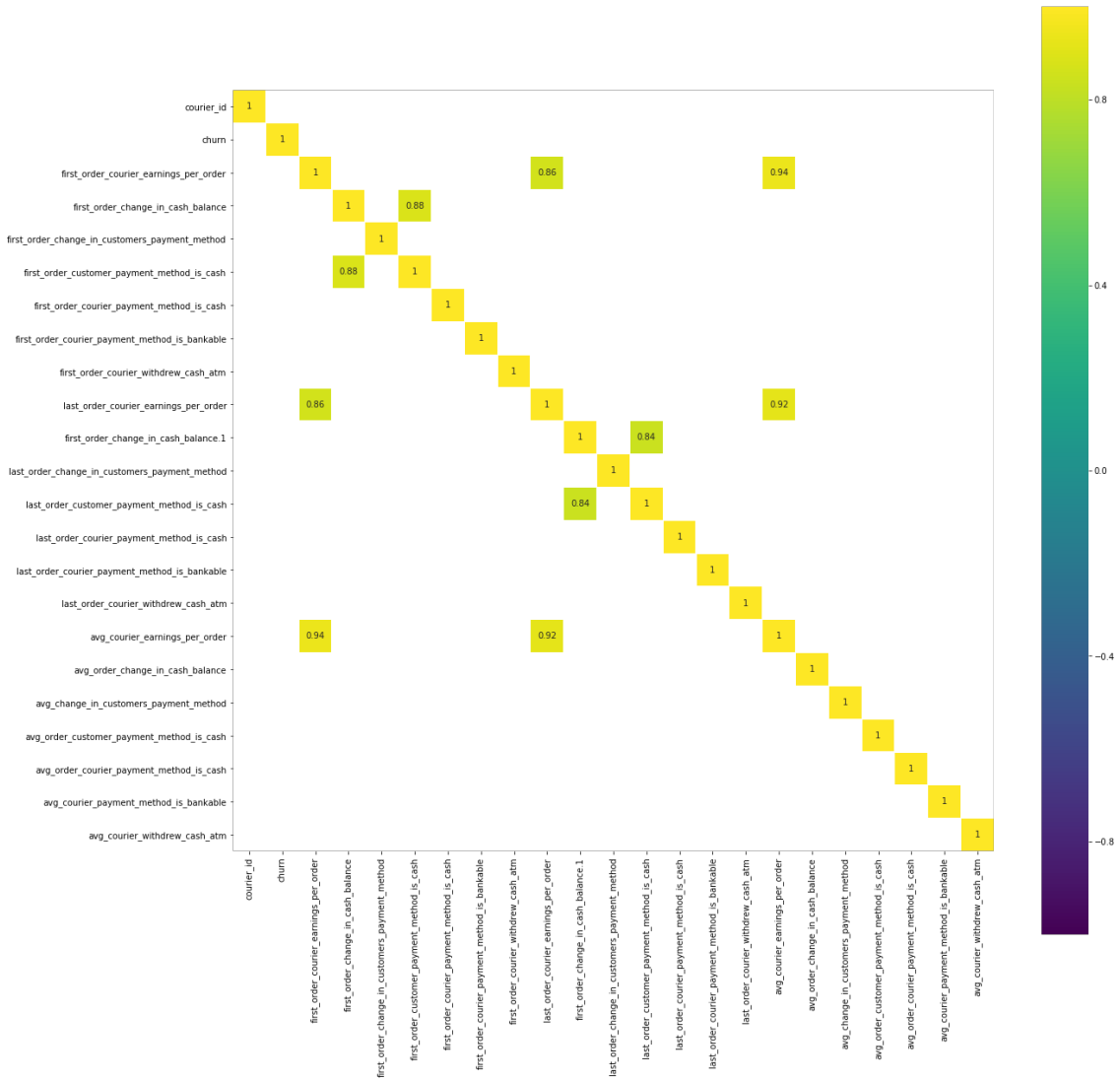

Figura 5-6. Matriu de correlacions del Bloc 6.

5.5 Bloc final

Després d'unir tots els blocs mitjançant el *Courier Id* (identificador únic per glover) acabem obtenint un total de 146 *features* com veiem a la Taula 5-4.

Taula 5-4. Distribució de *features* final.

Block	Initial Features	Final Features
1	74	70
2	9	8
3	33	26
4	30	22
5	19	3
6	21	17
Total	**186**	**146**

Un cop unides totes les variables tornem a repetir el procés per mirar si existeix alguna correlació entre elles, en aquest cas trobem 2 variables que estan relacionades i procedim a eliminar-les (el període d´hores d'estudi i el total d'ordres efectuades en el període d'estudi). El que ens deixa un total de 144 *features* en el *dataset*.

5.6 *Step forward feature selection* (SFFS)

Els mètodes de *feature selection* pretenen reduir les variables d'entrada a aquell número que es creu que seran més útils perquè el model predigui la variable objectiu.

Alguns *datasets* poden tenir una gran nombre de variable que poden retardar el desplegament i temps d'entrenament, a part de requerir una gran quantitat de memòria. A més, com ja hem comentat el rendiment d'alguns models es pot degradar al incloure variables d'entrada que no son rellevants per predir la variable objectiu.

Les principals avantatges d'aplicar *feature selection* són:

- **Reducció *d'overfitting***. Menys dades redundants impliquen menys oportunitats per prendre decisions basades en soroll.
- **Millora de la precisió**. Menys dades irrellevants implica que la precisió del model augmenti.
- **Reducció del temps d'entrenament**. Menys punts de dades redueixen la complexitat de l'algorisme, el que resulta en un entrenament més ràpid.

Una manera de pensar en els mètodes de *feature selection* és en termes de supervisats o no supervisats. La diferència recau entre si les *features* són seleccionades en funció de la variable objectiu o no. Els mètodes de *feature selection* no supervisats ignoren la variable objectiu, i es dediquen a per exemple eliminar variables redundants mitjançant la correlació, que és el que hem fet a l'apartat 5.4 . En canvi, els mètodes supervisats si que utilitzen la variable objectiu per eliminar variables irrellevants.

Una altre manera es considerar el mecanisme utilitzat per seleccionar les variables que es pot dividir entre mètodes *wrapper* i *filter*. Aquest mètodes són en gran mesura supervisats i s'avaluen en funció del rendiment d'un model. El mètode *filter feature selection* utilitza tècniques estadístiques per avaluar la relació entre cada variable d'entrada i la variable objectiu, i aquestes puntuacions s'utilitzen com a base per escollir (filtrar) aquelles variables d'entrada que s'utilitzaran en el model.

Els mètodes *wrapper*, tal i com veiem a la figura 5-7, fan servir un procés disciplinat que uneix el procés de selecció de les features amb el tipus de model que s'està construint, avaluant el subconjunt de *features* per detectar el rendiment del model entre les diferents variables i seleccionar posteriorment el subconjunt que hagi tingut el major rendiment. En altres paraules, enlloc d'existir com a procés independent que es produeix abans de la creació dels models, els mètodes *wrapper* intenten optimitzar el procés de *feature selection* per un algorisme d'aprenentatge automàtic en concret en tàndem amb aquest algorisme [10].

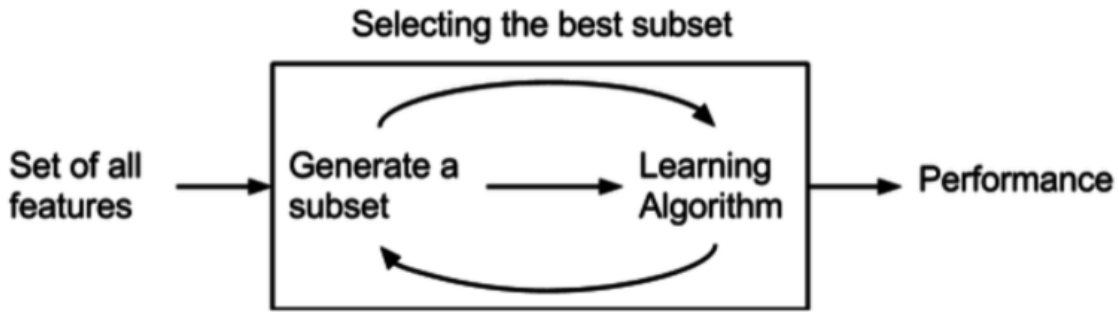

Figura 5-7. Mètode *wrapper feature selection.*

Un dels mètodes *wrapper* més destacats és el *step forward feature selection*, aquest mètode comença amb l'avaluació de cada *feature* del model, i selecciona la que aconsegueix el millor rendiment del algorisme seleccionat. El millor rendiment depèn dels criteris d'avaluació definits (AUC, precisió, RMSE, etc.). Un cop escollida la primera *feature*, es procedeixen a avaluar totes les combinacions possible entre la primera *feature* seleccionada i les altres, un cop avaluades se´n selecciona la millor i aquest procés es repeteix successivament fins a arribar al nombre de *features* desitjat.

En aquest treball hem utilitzat el *step forward feature selection* amb un *Random Forest Classifier* (100 *estimators*). Hem escollit com a criteri d'avaluació la *precision*. Tal i com veiem en la figura 5-8, veiem que el conjunt de *features* òptim es de 107 amb una *precision* del 86,9%.

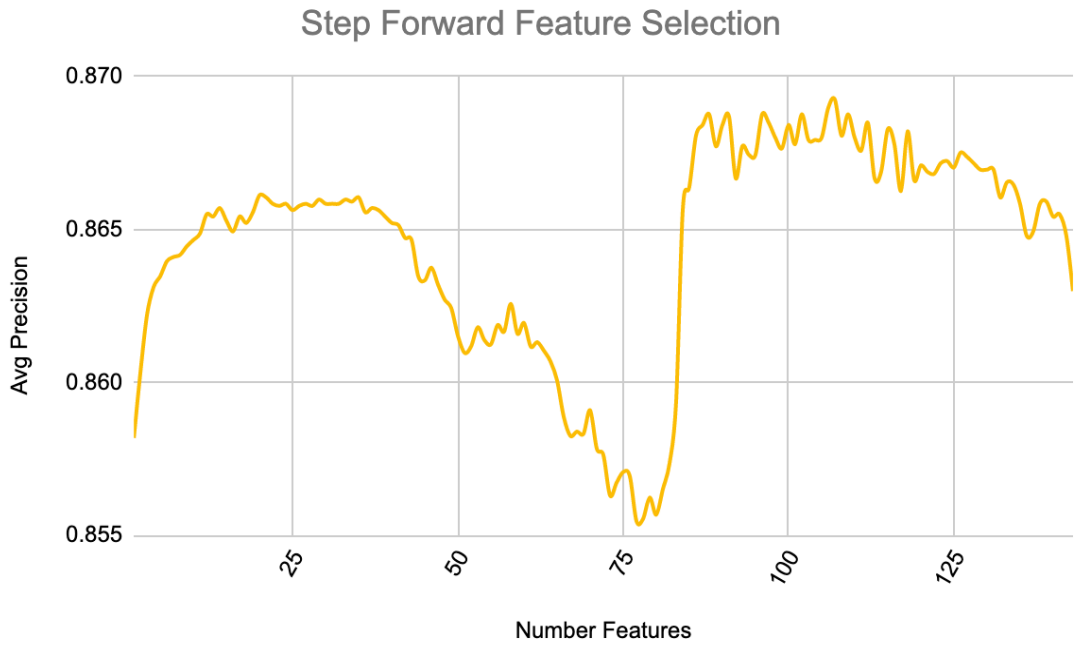

Figura 5-8. Avg precision vs Número features (*Step forward feature selection*)

6 Aprenentatge automàtic aplicat al *churn*

6.1 Metodologia d'anàlisi de resultats

Abans de començar qualsevol dels experiments, és necessari determinar com se'n mesurarà l'èxit. En aquesta secció, s'explica com s'avaluen els resultats i com es comparen entre ells. Els resultats s'avaluaran segons la metodologia pròpia de l'aprenentatge automàtic aplicat al problema que s'està resolent, que es la predicció del *churn* dels glovers.

En els experiments de classificació, la sortida dels algoritmes són un conjunt finit de categories. Per analitzar els resultats, és molt comú utilitzar la matriu de confusió:

		Valors reals		
		Positiu	Negatiu	Total
Valors predits	Positiu	a	b	$a+b$
	Negatiu	c	d	$c+d$
	Total	$a+c$	$b+d$	N

Figura 6-1. Matriu de confusió.

La matriu de confusió (figura 6-1) permet observar visualment l'encert d'un model classificador. Si se sumen tots els valors de la matriu (N), el resultat ha de ser el mateix que el nombre d'elements que han estat classificats pel model. La matriu té tantes columnes com classes pot tenir un element. I és en les columnes on hi ha els valors reals. Si se sumen els valors de la primera columna (a+c), s'obté el nombre d'element que té la classe positiu. En canvi, sumant els valors de la primera fila (a + b), s'obté el nombre d'elements que el model ha classificat com a classe positiu. A partir d'aquesta matriu, es poden extreure informació sobre el classificador que es descriu a continuació:

- ***Accuracy*** (Exactitud). És la fracció d'elements classificats correctament.

$$Accuracy = \frac{a+d}{N}$$

Equació 6-1. *Accuracy*.

- ***Recall*** (Sensitivitat). Mesura la proporció d'elements classificats correctament de la classe positiva. També es pot taxa de positius vertaders.

$$Recall = \frac{a}{a+c}$$

Equació 6-2. *Recall*.

- ***Specificity*** (Especificitat). Mesura la proporció d'elements classificats correctament de la classe negativa. També es pot anomenar taxa de negatius vertaders.

$$Specificity = \frac{d}{d+b}$$

Equació 6-3. *Specificity*.

- ***Precision*** (Precisió). És la fracció d'elements rellevants de la classe positiva.

$$Precision = \frac{a}{a+b}$$

Equació 6-4. *Precision*.

- ***F1-score***. Es tracta d'una mesura que equilibra tant la *precision* com el *recall*.

$$F1\_score = \frac{2 \cdot Recall \cdot Precision}{Recall + Precision}$$

Equació 6-5. *F1-Score*.

Amb la sensitivitat i l'especificitat també es pot construir un altre indicador: la corba ROC (Figura 6-2). En un classificador binari, es pot generar un límit que indica el punt on es diferencien les dues classes. El gràfic es genera a partir dels valors de la sensitivitat i l'especificitat del model amb diferents llindars. Un model aleatori, traça una línia diagonal mentre que en un classificador perfecte, la sensitivitat i l'especificitat són 1 i es dibuixa un esglaó.

A part de la visualització, la corba ROC proporciona l'indicador Area Under the Curve. Com més gran sigui l'àrea, millor serà el model. Com que el valor màxim dels eixos és 1, l'àrea màxim també tindrà aquest valor. Tota àrea que estigui per sota de 0,5 indicarà que el model és pitjor que un d'aleatori.

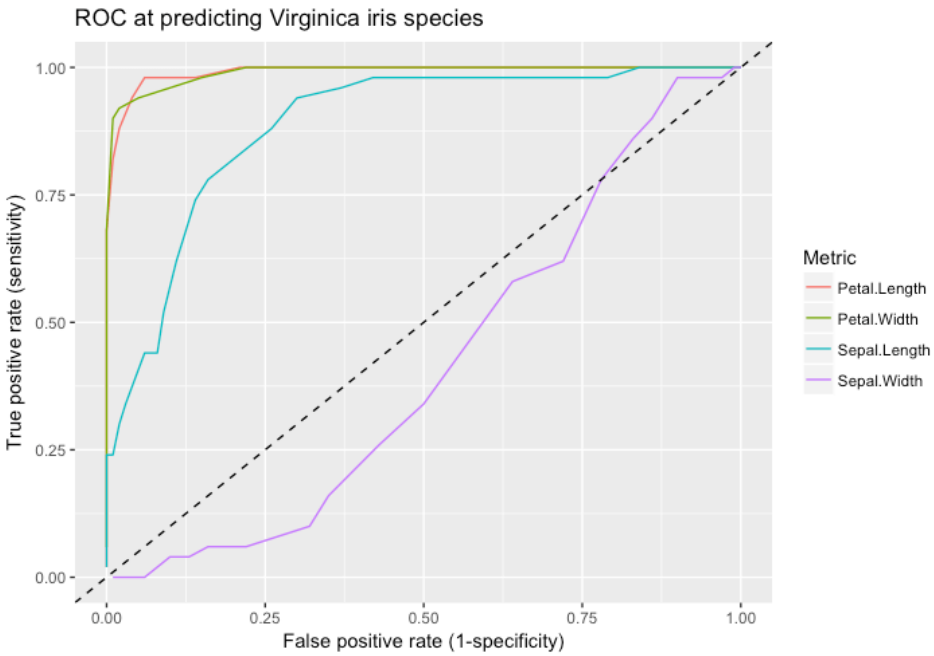

Figura 6-2. Corba ROC.

En la figura 6-2 s'observa com els models vermell i verd són millors que el turquesa perquè s'apropen més al classificador ideal. Mentre que el lila és pitjor que un classificador aleatori.

En aquest treball avaluarem totes aquestes diferents mètriques però en la que ficarem més atenció serà la *F1-Score*, ja que per detectar el *churn* dels glovers volem buscar un balanç entre la *precision* i el *recall*, i aquesta mètrica ens les equilibra trobant el seu màxim. Busquem maximitzar aquesta mètrica ja que les accions que executarem no tenen un cost associat

econòmic molt gran, però si aquestes accions impliquessin un cost molt gran llavors buscaríem maximitzar la precisió.

De maximitzar el *F1-Score*, que serà quan tant la *precision* i el *recall* siguin màximes, obtindrem el *threshold* a partir del qual decidirem si una predicció es positiva o negativa. Totes les mètriques dels resultats presentats s'obtindran a partir d'aquest *threshold*.

Recordem que disposem de dos *datasets* de dades:

- Entrenament. Conté les dades de *churn* dels couriers des del mes de desembre del 2019 fins al mes d'abril del 2020.
- Validació. Conté les dades de *churn* dels couriers del mes de maig del 2020.

Per tal d'avaluar els primers resultats dels models primer agafarem el *dataset* d'entrenament i el separarem perquè un 70% de les dades siguin del *train set* i el 30% restant del *test set*. Això ho fem per evitar l'*overfitting*, ja que entrenem l'algorisme amb les dades d'entrenament i després l'avaluem sobre les dades de test per veure com generalitza. També forcem la condició de que els sets tinguin *stratify*, per garantir que tindrem una distribució entre la classe positiva i la negativa. Un cop obtinguts els primers resultats aplicarem el model resultant sobre el *dataset* de validació per tal d'avaluar com es comporta el model sobre un conjunt de dades noves.

Cal tenir en compte que a cada model se li ha aplicat un *hyperparameter tunning* amb un *GridSearchCV* amb un *cross-validation* de per tal d'obtenir el conjunt d'hiperparametres i valors valors òptims que maximitzen la precisió del algorisme.

6.2 Resultats

6.2.1 *Decision Tree*

Per aquest algorisme hem utilitzat la classe *sklearn.tree.DecisionTreeClassifier*, on a la figura 6-3 podem veuran els seus paràmetres i valors després d´haver-li aplicat *hyperparameter tunning.*

```
clf_dt = DecisionTreeClassifier(class_weight=None, criterion='gini', max_depth=8,
                        max_features='auto', max_leaf_nodes=None,
                        min_impurity_decrease=0.0, min_impurity_split=None,
                        min_samples_leaf=5, min_samples_split=500,
                        min_weight_fraction_leaf=0.0, presort=False,
                        random_state=42, splitter='random')
```

Figura 6-3. Classe *DecisionTreeClassifier*

A la figura 6-4 veiem la gràfica de la *precision* i *recall* de les prediccions del model en el *dataset* de test (30% de les dades d'entrenament), d'on podem veure que al *threshold* 0,18 tenim la màxima *precision* i *recall* (*F1-score* màxima).

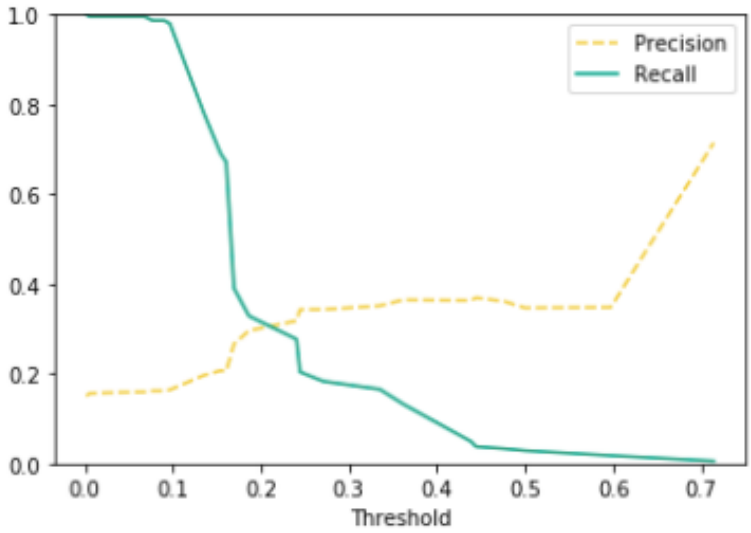

Figura 6-4. Gràfica *Precision* i *Recall* vs *Threshold (Decision Tree)*

Després d'aplicar el *threshold* seleccionat a la sortida del model obtenim la següent matriu de confusió (Taula 6-1) i les següents mètriques (Taula 6-2) sobre el *dataset* de test.

Taula 6-1. Matriu de confusió del *Decision Tree* (test *dataset*)

	Predicció	
Referència	No Churn	Churn
No Churn	4713	539
Churn	656	251

Taula 6-2. Mètriques del *Decison Tree* (test *dataset*)

Mètrica	**Churn**
Accuracy	80,59 %
Precision	29,58 %
Recall	29,58 %
ROC (AUC)	58,70 %
F1-score	29,58 %

Si ara apliquem el model sobre les dades de validació obtenim la següent matriu de confusió (Taula 6-3) i mètriques (Taula 6-4).

Taula 6-3. Matriu de confusió del *Decision Tree* (validation *dataset*)

	Predicció	
Referència	No Churn	Churn
No Churn	943	47
Churn	105	20

Taula 6-4. Mètriques del *Decison Tree* (validation *dataset*)

Mètrica	**Churn**
Accuracy	86,36 %
Precision	29,85 %
Recall	16,00 %
ROC (AUC)	55,62 %
F1-score	20,08 %

6.2.2 *Random Forest*

Per aquest algorisme hem utilitzat la classe *sklearn.ensemble.RandomForestClassifier*, on a la figura 6-5 podem veuren els seus paràmetres i valors després d'haver-li aplicat *hyperparameter tunning.*

```
clf_rf = RandomForestClassifier(bootstrap=False, class_weight=None, criterion='gini',
                        max_depth=14, max_features='auto', max_leaf_nodes=None,
                        min_impurity_decrease=0.0, min_impurity_split=None,
                        min_samples_leaf=20, min_samples_split=550,
                        min_weight_fraction_leaf=0.0, n_estimators=20, n_jobs=-1,
                        oob_score=False, random_state=42, verbose=0,
                        warm_start=False)
```

Figura 6-5. Classe *RandomForestClassifier*

A la figura 6-6 veiem la gràfica de la *precision* i *recall* de les prediccions del model en el *dataset* de test (30% de les dades d'entrenament), d'on podem veure que al *threshold* 0,29 tenim la màxima *precision* i *recall* (*F1-score* màxima).

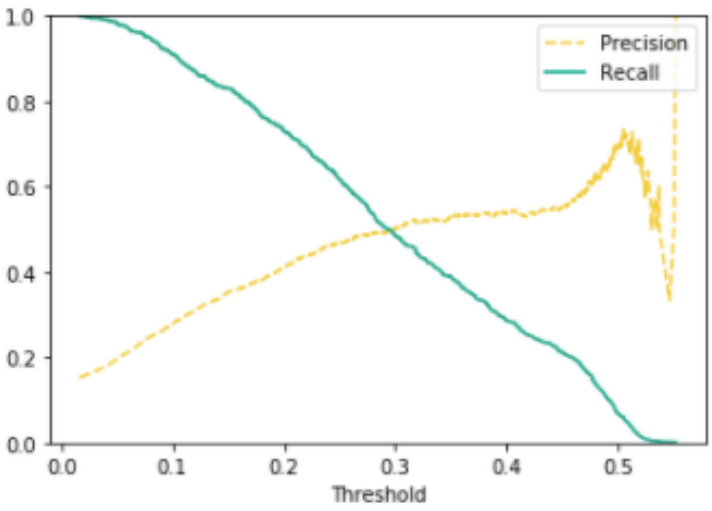

Figura 6-6. Gràfica Precision i Recall vs Threshold (Random Forest)

Després d'aplicar el *threshold* seleccionat a la sortida del model obtenim la següent matriu de confusió (Taula 6-5) i les següents mètriques (Taula 6-6) sobre el *dataset* de test.

Taula 6-5. Matriu de confusió del Random Forest (test dataset)

	Predicció	
Referència	No Churn	Churn
No Churn	4796	456
Churn	456	451

Taula 6-6. Mètriques del *Random Forest* (test *dataset*)

Mètrica	**Churn**
Accuracy	85,19 %
Precision	49,72 %
Recall	49,72 %
ROC (AUC)	70,05 %
F1-score	49,72 %

Si ara apliquem el model sobre les dades de validació obtenim la següent matriu de confusió (Taula 6-7) i mètriques (Taula 6-8).

Taula 6-7. Matriu de confusió del *Random Forest* (validation *dataset*)

	Predicció	
Referència	No Churn	Churn
No Churn	914	76
Churn	61	64

Taula 6-8. Mètriques del *Random Forest* (validation *dataset*)

Mètrica	Churn
Accuracy	87,71 %
Precision	45,71 %
Recall	51,2 %
ROC (AUC)	71,76 %
F1-score	48,30 %

6.2.3 *Gradient Boosting*

Per aquest algorisme hem utilitzat la classe *sklearn.ensemble.GradientBoostingClassifier*, on a la figura 6-7 podem veuren els seus paràmetres i valors després d´haver-li aplicat *hyperparameter tunning.*

```
clf_gbc = GradientBoostingClassifier(criterion='friedman_mse', init=None,
                           learning_rate=0.06, loss='deviance', max_depth=6,
                           max_features='sqrt', max_leaf_nodes=None,
                           min_impurity_decrease=0.0, min_impurity_split=None,
                           min_samples_leaf=20, min_samples_split=150,
                           min_weight_fraction_leaf=0.0, n_estimators=120,
                           n_iter_no_change=None, presort='auto',
                           random_state=42, subsample=0.8, tol=0.0001,
                           validation_fraction=0.1, verbose=0,
                           warm_start=False)
```

Figura 6-7. Classe *GradientBoostingClassifier*

A la figura 6-8 veiem la gràfica de la *precision* i *recall* de les prediccions del model en el *dataset* de test (30% de les dades d'entrenament), d'on podem veure que al *threshold* 0,36 tenim la màxima *precision* i *recall* (*F1-score* màxima).

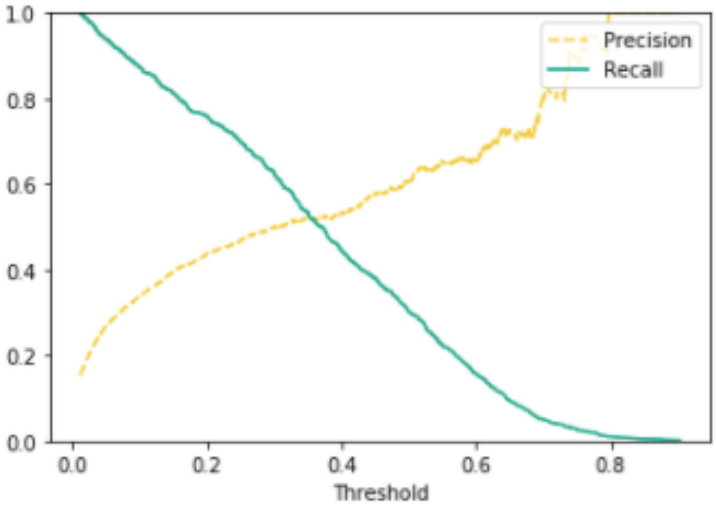

Figura 6-8. Gràfica Precision i Recall vs Threshold (Gradient Boosting)

Després d'aplicar el *threshold* seleccionat a la sortida del model obtenim la següent matriu de confusió (Taula 6-9) i les següents mètriques (Taula 6-10) sobre el *dataset* de test.

Taula 6-9. Matriu de confusió del Random Forest (test dataset)

	Predicció	
Referència	No Churn	Churn
No Churn	4816	436
Churn	436	471

Taula 6-10. Mètriques del *Random Forest* (test *dataset*)

Mètrica	**Churn**
Accuracy	85,84 %
Precision	51,92 %
Recall	51,92 %
ROC (AUC)	71,81 %
F1-score	51,92 %

Si ara apliquem el model sobre les dades de validació obtenim la següent matriu de confusió (Taula 6-11) i mètriques (Taula 6-12).

Taula 6-11. Matriu de confusió del *Random Forest* (validation *dataset*)

	Predicció	
Referència	No Churn	Churn
No Churn	922	68
Churn	69	56

Taula 6-12. Mètriques del *Random Forest* (validation *dataset*)

Mètrica	Churn
Accuracy	87,71 %
Precision	45,16 %
Recall	44,8 %
ROC (AUC)	68,96 %
F1-score	44,97 %

6.2.4 *Extreme Gradient Boosting* (XGBoost)

Per aquest algorisme hem utilitzat la classe *sklearn.xgboost.XGBClassifier*, on a la figura 6-9 podem veuren els seus paràmetres i valors després d´haver-li aplicat *hyperparameter tunning.*

```
clf_xgb = XGBClassifier(base_score=0.5, booster='gbtree', colsample_bylevel=1,
              colsample_bynode=1, colsample_bytree=0.8, eval_metric='auc',
              gamma=0, learning_rate=0.1, max_delta_step=0, max_depth=4,
              min_child_weight=1, missing=None, n_estimators=60, n_jobs=1,
              nthread=4, objective='binary:logistic', random_state=0,
              reg_alpha=1e-06, reg_lambda=1, scale_pos_weight=1, seed=42,
              silent=None, subsample=0.8, verbosity=1)
```

Figura 6-9. Classe XGBClassifier

A la figura 6-10 veiem la gràfica de la *precision* i *recall* de les prediccions del model en el *dataset* de test (30% de les dades d'entrenament), d'on podem veure que al *threshold* 0,36 tenim la màxima *precision* i *recall* (*F1-score* màxima).

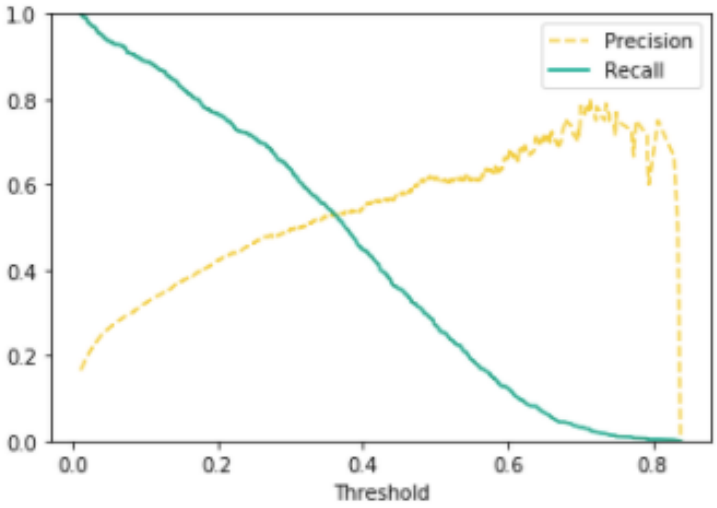

Figura 6-10. Gràfica Precision i Recall vs Threshold (XGBoost)

Després d'aplicar el *threshold* seleccionat a la sortida del model obtenim la següent matriu de confusió (Taula 6-13) i les següents mètriques (Taula 6-14) sobre el *dataset* de test.

Taula 6-13. Matriu de confusió del *XGBoost* (test dataset)

	Predicció	
Referència	No Churn	Churn
No Churn	4826	426
Churn	426	481

Taula 6-14. Mètriques del *XGBoost* (test *dataset*)

Mètrica	**Churn**
Accuracy	86,16 %
Precision	53,03 %
Recall	53,03 %
ROC (AUC)	72,46 %
F1-score	53,03 %

Si ara apliquem el model sobre les dades de validació obtenim la següent matriu de confusió (Taula 6-15) i mètriques (Taula 6-16).

Taula 6-15. Matriu de confusió del *XGBoost* (validation *dataset*)

	Predicció	
Referència	No Churn	Churn
No Churn	927	63

Churn	70	55

Taula 6-16. Mètriques del *XGBoost* (validation *dataset*)

Mètrica	Churn
Accuracy	88,71 %
Precision	46,61 %
Recall	44,00 %
ROC (AUC)	68,81 %
F1-score	45,26 %

6.2.5 Xarxes neuronals

Per aquest algorisme hem utilitzat la classe *tf.keras.Sequential*, on a la figura 6-11 podem veure la configuració que li hem aplicat. Hem definit un *input layer* de 107 neurones on els *hidden layers* són configurables, i hem dimensionat la sortida d'una neurona per indicar si ha fet *churn* o no.

```
nn_seq = Sequential()
# Input and first hidden layer
nn_seq.add(Dense(input_dim = 107, units=units_1, activation='relu', kernel_initializer='glorot_normal'))
# Second hidden layer
nn_seq.add(Dense(units=units_2, activation='relu', kernel_initializer='glorot_normal')
# Output layer
nn_seq.add(Dense(units=1, activation="sigmoid", kernel_initializer='glorot_normal'))
# Compile setting
nn_seq.compile(loss="mse", optimizer=Adam(lr=0.0001), metrics=['accuracy'])
# Fit
nn_seq.fit(X_train, y_train.values, validation_split=0.33, epochs=epochs, verbose=0, shuffle=False)
```

Figura 6-11. Classe *Sequential* de *Keras.*

Tal i com s'ha comentat en l'apartat 3.2.5, en les xarxes neuronals no hi ha cap norma específica per determinar la seva estructura, aquesta s'obté mitjançant prova i error. Per tal doncs de trobar aquesta estructura, hem fet un seguit d'experiments amb diferents estructures optimitzades per un número concret d'*epochs* per tal de no fer *overfitting* o *underfitting*, i fixant els valors del kernel_*initializer*, *activation*, *loss* i *optomitzer* als valors que veiem a la Figura 6-11. Com que les xarxes neuronals compten amb aquesta componen d'aleatorietat cada cop que es defineixen, s´ha repetit l'experiment 30 cops per el *dataset* d'entrenament i s'ha agafat la mitjana del *F1-score* per a cada una de les estructures. Aquests experiments es resumeixen a la següent taula 6-17.

Taula 6-17. Figura 6 12. Experiments estructura ANN.

Epochs	Hidden Layer 1	Hidden Layer 2	F1-score
30	10	0	49.61%
17	20	0	49.20%
11	50	0	49.54%

14	70	0	49.87%
9	90	0	49.47%
12	110	0	49.70%
7	130	0	49.57%
8	170	0	49.50%
8	200	0	49.64%
8	230	0	49.82%
8	270	0	49.85%
20	20	10	49.63%
15	50	20	49.48%
15	20	50	49.45%
15	70	20	49.61%
15	50	70	49.35%
10	110	70	49.51%
8	170	110	49.51%

De la Taula 6-17 veiem que l'estructura que ens dóna millors resultats és la que té un únic *hidden layer* de 70 nodes amb 14 *epochs* amb un *F1-score* del 49,87%. S´han escollit 14 epochs perquè tal i com veiem a la figura 6-1 a partir d'aquest número veiem com l'*accuracy* i la *loss* del *dataset* de test s'estabilitza i l'ANN comença a fer *overfitting* sobre les dades d'entrenament.

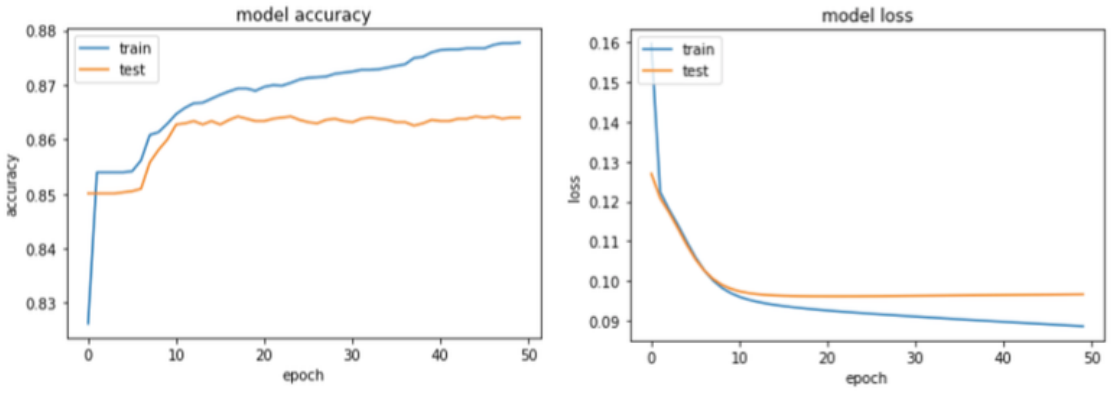

Figura 6-12. Accuracy i loss.

A la figura 6-13 veiem la gràfica de la *precision* i *recall* de les prediccions del model en el *dataset* de test (30% de les dades d'entrenament), d'on podem veure que al *threshold* 0,33 tenim la màxima *precision* i *recall* (*F1-score* màxima).

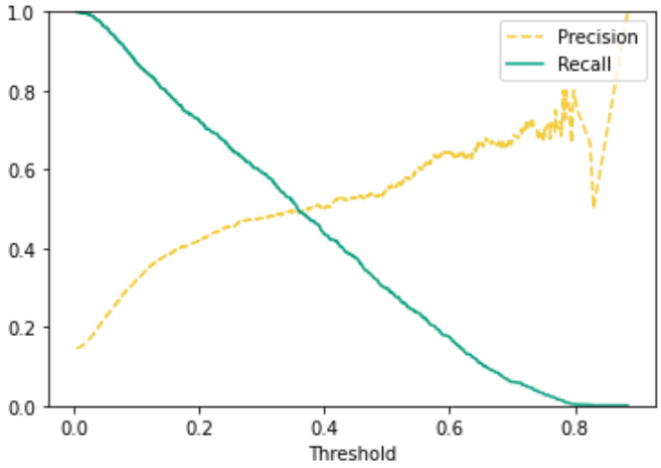

Figura 6-13. Gràfica Precision i Recall vs Threshold (ANN).

Després d'aplicar el *threshold* seleccionat a la sortida del model obtenim la següent matriu de confusió (Taula 6-18) i les següents mètriques (Taula 6-19) sobre el *dataset* de test.

Taula 6-18. Matriu de confusió de la ANN (test dataset)

	Predicció	
Referència	No Churn	Churn
No Churn	4792	460
Churn	460	447

Taula 6-19. Mètriques de la ANN (test *dataset*)

Mètrica	**Churn**
Accuracy	85,06 %
Precision	49,23 %
Recall	49,23 %
ROC (AUC)	70,26 %
F1-score	49,23 %

Si ara apliquem el model sobre les dades de validació obtenim la següent matriu de confusió (Taula 6-20) i mètriques (Taula 6-21).

Taula 6-20. Matriu de confusió de la ANN (validation *dataset*)

	Predicció	
Referència	No Churn	Churn
No Churn	925	65
Churn	71	54

Taula 6-21. Mètriques de la ANN (validation *dataset*)

Mètrica	Churn
Accuracy	87,80 %
Precision	45,37 %
Recall	43,30 %
ROC (AUC)	68,31 %
F1-score	44,26 %

7 Anàlisi i discussió

L'anàlisi i discussió dels resultats es farà a partir de la taula 7.1 que mostra les mètriques aconseguides pels diferents algorismes sobre les dades de validació comentades en l'anterior capítol.

Taula 7-1. Mètriques algorismes.

Mètrica	Decision Tree	Random Forest	Grandient Boosting	XGboost	ANN
Accuracy	86,36 %	87,71 %	87,73 %	88,71 %	87,80 %
Precision	29,85 %	45,71 %	45,16 %	46,61 %	45,37 %
Recall	16,00 %	51,2 %	44,8 %	44,00 %	43,20 %
ROC (AUC)	55,62 %	71,76 %	68,96 %	68,81 %	68,31 %
F1-score	20,08 %	48,30 %	44,97 %	45,26 %	44,26 %

Si ens fixem en les diferents *accuracies* dels models, veiem que totes tenen valors similars. Però aquesta es una mètrica enganyosa per el problema que volem resoldre, ja que mesura quants encerts en total ha encertat l'algorisme, però per naturalesa les nostres dades no estan balancejades ja que tenim molt més glovers que no fan *churn* que glovers que en fan. Així doncs, l'accuracy té un pes molt més gran en la classe majoritària (no churn) que no en la minoritària (churn), però per resoldre el problema plantejat ens interessa molt més que l'algorisme identifiqui correctament aquells glovers que si que fan *churn*, es per això que en problemes no balancejats ens fixerem en la *precision*, *recall* i el F1-score.

L'algorisme que pitjor es comporta és el *Decision Tree*, ja que tot i tenir una *accuracy* similar als altres models (mètrica enganyosa) té una *precision* i *recall* considerablemt més baixa. Això es deu ja que es el model més senzill escollit i les seves capacitats són limitades, tal i com ja s'ha comentat en l'apartat 3.2.5 aquests models tendeixen a fer *overfitting* ja que no generalitzen bé.

L'ANN no ha destacat per obtenir uns resultats àmpliament superiors a la resta de models, tot i que a priori es podia pensar que era el que millor funcionaria degut al seu potencial. Tal i com s'ha explicat a l'apartat 3.2.1 el poc volum de dades i dimensions que presenta aquest problema fan que el seu rendiment sigui comparable al dels altres algorismes tradicionals estudiats.

Com s'ha comentat, tots els algorismes excloent el *Decison Trees* tenen un comportament similar i podrien ser candidats per ser escollits com a model per posar en producció. Però dels resultats cal destacar que sorprenentment el que té un millor comportament es el *Random Forest*, això es podria deure a que per tal de reduir la dimensionalitat de les dades s'ha aplicat un *step forward feature selection* amb un *Random Forest* com a model. Llavors podríem concloure que com que les dades s'han agafat sobreponderant l'algorisme de *Random Forest, per això* aquest ha obtingut uns millors resultats.

Com a última conclusió cal ressaltar els bons resultats aconseguits tenint en compte la finestra temporal de 5 dies que s'ha escollit, ja que amb molts pocs dies podem arribar a tenir un nivell de fiabilitat relativament alt per tal d'impactar aquells glovers amb risc de *churn* i així reactivar-los.

8 Conclusions

8.1 Assoliment dels objectius

A la introducció del treball es definien un conjunt d'objectius a assolir. A continuació, es valora el seu assoliment:

1. **Definició del glover *churn*.** S'ha definit glover *churn* a la regió d'EEMEA com aquells glovers que passats 5 dies després d´haver fet la seva primera ordre (CFO) no hagin repartit més.

2. **Extracció i estructuració les dades històriques dels glovers**. Pel que fa a l'extracció s'han definit diferents blocs agrupats amb característiques de comuna similitud per tal d'englobar-les en diferents queries d'SQL. Respecte l'estructuració, s'ha dividit la informació de les features entra la primera ordre, la última ordre i la mitjana del que ha passat entre totes les ordres entre la primera i la última

3. **Aplicació de preprocessament de dades**. S'ha aplicat un procés per gestionar duplicats, *missing values*, *categorical encoding* i correlacions. Finalment s'ha efectuat un *feature selection* al dataset mitjançant un SFFS.

4. **Implementació i comparativa de diferents algorismes d'aprenentatge automàtic**. S'ha implementat i comparat un decision tree, un random forest, un gradient boosting, un *XGBoost* i addicionalment una xarxa neuronal.

8.2 Cost

En aquest apartat es descriuran les hores dedicades per la realització del projecte. Per tal de dur-lo a terme s'han realitzat un seguit de tasques, on a continuació es descriuen:

- **Estudi**. Es la fase on s'ha hagut d'aprendre tots aquells coneixements necessaris per a la realització del projecte. Que inclouen la consolidació del llenguatge Python, així com les seves corresponents llibreries i l'entorn Anaconda - Navigator (Jupyter). També s'ha profunditzat en coneixements d'SQL per l'extracció de dades i finalment en tot el referent al aprenentatge automàtic desconegut amb anterioritat.

- **Disseny**. Aquestes hores s'han dedicat a analitzar els problemes i a buscar solucions. En aquesta fase s´ha definit el *glover churn*, l'estructuració de les dades i el preprocessament de dades i algorismes a implementar.

- **Implementació**. Consisteix en el desenvolupament del codi que ha materialitzat les idees del disseny.

- **Simulacions**. Inclou totes les diferents simulacions que s'han realitzat, la recollida de resultats i el seu posterior anàlisi.

- **Documentació**. Es la fase d'elaboració de la memòria.

La primera i l'última fase del projecte, estudi i documentació, són les que han suposat menys cost temporal sumant un 20% entre les dues. Tanmateix, la fase de disseny ha comportat un major cost temporal que les anteriors ja que condiciona l'èxit del treball posterior. La implementació és, com en qualsevol projecte d'aquest estil, la part que necessita més temps per ser completada. En aquest cas, suposa gairebé la meitat de tot el treball. Un cop fet el codi, les simulacions tampoc són immediates. S'han de fer moltes proves amb diferents variacions i el temps d'entrenament és força alt, sobretot si s'ha de repetir moltes vegades. Per últim, també ha estat destacable el volum de feina que ha suposat l'escrit del treball.

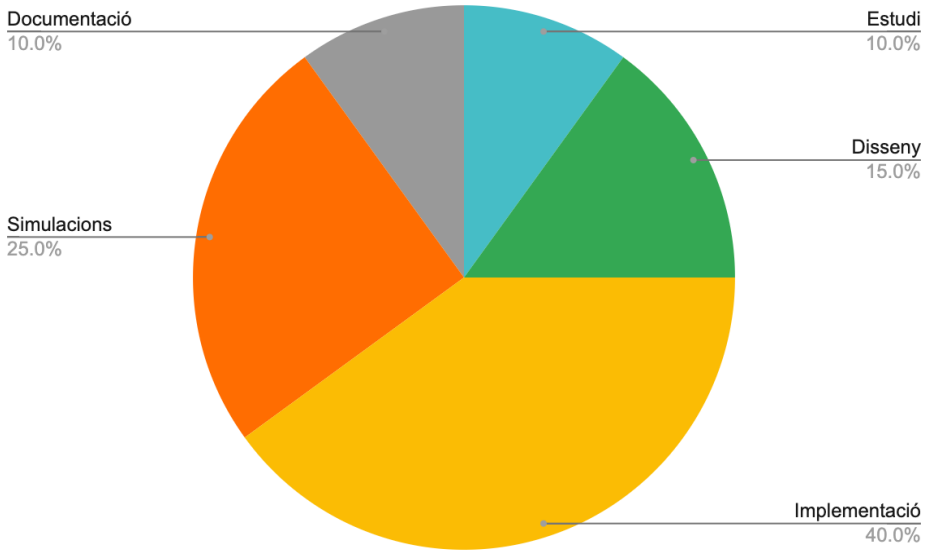

Figura 8-1. Cost temporal del projecte.

8.3 Lliçons apreses

Es el moment de concloure el treball amb les lliçons que s'han pogut extreure. Primer de tot, ha estat difícil aconseguir les dades, ja que les dades estaven localitzades a multitud de taules que es desconeixen del DWH de Glovo i les nocions d'SQL no eren les adequades, amb aquest treball hem obtingut una visió clara de l'estructura del DWH així com una millora considerable de les nostres nocions d'SQL.

També s'ha après a com resoldre un problema de negoci aplicant un procés de vessant tecnològica, en aquest cas, hem aplicat un procés de *Data Science* punta a punta, des de la definició de la variable objectiu, passant per l'extracció de dades i preprocessament fins a arribar a la implantació de models.

Pel que fa als algorismes d'aprenentatge automàtic, hem vist que el rendiment de les xarxes neuronals amb poques dades es equiparable al dels algorismes d'aprenentatge automàtic tradicionals. També s'ha après a llegir i entendre les mètriques dels resultats correctament en problemes de dades no balancejades. Sobre aquesta temàtica també s'ha après a realitzar i aplicar *hyperparameter tunning* sobre els models per tal d'obtenir els millors paràmetres

possibles, també destacar la importància del *cross-validation* a l'hora d'avaluar els models ja que sinó podríem arribar a conclusions errònies.

El fet de predir el *churn* dels glovers a part de tenir un impacte molt gran en els beneficis de la companyia també té un impacte molt gran en les operacions, ja que es dediquen molts recursos d'agents i CRM per intentar resoldre aquest problema. Mitjançant els models implementats es podrà segmentar millor a la flota de glovers, i amb aquesta segmentació es podran dedicar els recursos necessaris per impactar a aquells que tinguin un risc de *churn* considerable.

8.4 Línies de futur

En aquest treball s´han obert moltes vies d'estudi: definició del glover *churn*, zona geogràfica, preprocessament, algorismes, etc. Tots aquests punts poden servir de punt de partida de nous treballs.

Per començar podria resultar interessant canviar la definició del *glover churn*, es podria ampliar la finestra temporal a més dies o fins a mesos per tal de resoldre el problema del *late churn*. També es podria canviar la zona geogràfica, i enlloc de treballar en models centrats en regions es podrien centrar en països o fins i tot plantejar un model global.

Una altre línia de futur seria la formulació del problema en si mateix, es podria estructurar el problema com a seqüencial, on vectors d'esdeveniments es podrien generar per exemple cada dia, i utilitzar-los per predir el *churn*. Això es podria implementar mitjançant models dissenyats per a dades seqüencials com la *Recurrent Neural Network* (RRN) o el *Hidden Markov Model* (HHM). Aquesta formulació alternativa o d'altres poden oferir informació sobre com els senyals ajuden a predir el *churn* o a entendre millor als glovers.

Una altre línia podria ser aplicar mètodes per tal de balancejar les dades i comparar-ne els resultats entre afegir aquestes dades sintètiques o duplicades i les originals. També seria bona idea provar diferents mètodes de *feature selection*, es podria per exemple per a cada model a estudiar aplicar-li un feature selection sobre ell i després comparar-ne els resultats.

El *Reinforcement Learning* també semblaria una bona opció a implementar, ja que és un tipus d'algorisme que aprèn quan fa bé les coses i, en aquest cas, quan prediu millor el *churn*.

Per últim, una altra línia podria ser el desplegament del model en un servidor acompanyat d'una lògica de *backend* enfocada a comunicacions de CRM, que impactarien als diferents glovers en risc de *churn*.

9 Bibliografia

[1] Mundo | Historias - https://www.elmundo.es/papel/historias/
[2] La Vanguardia | Economía - https://www.lavanguardia.com/economia
[3] NG DATA | *Resources* - https://www.ngdata.com/
[4] *Optimove* | *Learning center* - https://www.optimove.com/resources/learning-center/
[5] *Altexsoft* | *Business* - https://www.altexsoft.com/blog/business
[6] *Nelipatel* | Blog - https://neilpatel.com/blog/
[7] Guilherme Dinis Chaliane Junior. *Churn analysis in a music streaming service*. Master's Thesis at KTH Information and Communication Technology, 2017.
[8] Yizhe Ge, Shan He, Jingyue Xiong and D. E. Brown. *Customer churn analysis for a software-as-a-service company*. SIEDS, Charlottesville, VA, 2017, pp. 106-111, doi: 10.1109/SIEDS.2017.7937698.
[9] *KD nuggets* | *Categorical features* - https://www.kdnuggets.com/
[10] *KD nuggets* | *Step forward feature selection* - https://www.kdnuggets.com/
[11] *Medium* | *Decision Trees* - https://www.medium.com/
[12] *DataCamp* | *Random Forest Classifiers* - https://www.datacamp.com/
[13] *Analytics Vidhya* | GBM - https://www.analyticsvidhya.com/
[14] *Medium*| *Gradient Boosting Machines* - https://www.medium.com/
[15] Chen, T., & Guestrin, C. (2016, August). Xgboost: A scalable tree boosting system. In Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining (pp. 785--794). ACM.
[16] *Towards Data Science* | *XGBoost* - https://towardsdatascience.com/
[17] Aggarwal, Charu C.. *Neural Networks and Deep Learning*. Cham: Springer, 2018.
[18] *Data Science Blog | ANN* - https://www.analyticsvidhya.com/
[19] *Towards Data Science*| *Introduction to Deep Learning* - https://towardsdatascience.com/
[20] *Machine Learning Mastery* | *Deep Learning* - https://machinelearningmastery.com/