Curs SAS ML - Analiza retenției clienților

(Customer Churn)

Introducere

Sectorul bancar operează într-un mediu competitiv și dinamic, unde clienții au la dispoziție multiple opțiuni, iar retenția acestora devine o provocare majoră pentru succesul pe termen lung. Menținerea unui portofoliu stabil de clienți este esențială pentru creșterea sustenabilă a instituțiilor financiare, iar analiza retenției clienților joacă un rol central în această strategie. Unul dintre principalii factori care influențează retenția este rata de renunțare a clienților (customer churn), un indicator care reflectă procentul celor care încetează colaborarea cu banca într-un anumit interval de timp. Această renunțare poate fi voluntară, atunci când clienții aleg să plece din motive personale, sau involuntară, în cazul în care relația se încheie din cauze externe, precum neplata obligațiilor financiare sau modificările în politicile instituției.

Analiza retenției clienților oferă informații valoroase despre factorii care determină plecarea acestora și ajută instituțiile bancare să identifice segmentele de clienți cu risc ridicat de renunțare. Prin monitorizarea indicatorilor cheie, cum ar fi frecvența tranzacțiilor, tipul serviciilor utilizate și interacțiunile cu banca, se pot dezvolta strategii proactive de fidelizare. În acest context, tehnologiile emergente și analiza avansată a datelor oferă oportunități semnificative pentru optimizarea retenției.

Algoritmii de învățare automată (Machine Learning) joacă un rol crucial în anticiparea comportamentului clienților, permițând identificarea unor tipare și tendințe care nu pot fi detectate prin metodele tradiționale. Modelele predictive de churn pot analiza variabile precum datele demografice, istoricul tranzacțiilor, frecvența utilizării serviciilor financiare și interacțiunile anterioare cu banca, generând astfel scoruri de probabilitate pentru fiecare client. Pe baza acestor analize, băncile pot lua măsuri personalizate pentru a preveni plecarea clienților valoroși, fie prin oferte speciale, fie prin îmbunătățirea experienței lor bancare.

Exemple concrete de aplicare a acestor metode includ bănci care detectează din timp clienții predispuși să își închidă conturile și le propun soluții personalizate, aplicații de mobile banking care adaptează notificările și recomandările în funcție de comportamentul utilizatorilor sau strategii de marketing direcționate către clienții inactivi. Astfel, analiza retenției clienților și utilizarea tehnologiilor de Machine Learning nu doar că îmbunătățesc capacitatea băncilor de a menține clienții loiali, ci contribuie și la optimizarea deciziilor de afaceri, sporind competitivitatea instituțiilor financiare pe termen lung.

Metodele tradiționale de analiză a retenției clienților prezintă multiple limitări, bazându-se pe reguli rigide care nu reușesc să surprindă toți factorii ce influențează decizia de renunțare. Aceste abordări pot duce la predicții inexacte și la măsuri de retenție ineficiente. Pentru a îmbunătăți acuratețea identificării clienților cu risc ridicat de renunțare, se propune utilizarea unui model de Machine Learning. Acest model va analiza un volum mare de date și va identifica tipare complexe, oferind predicții mai precise și reducând atât numărul alertelor false, cât și pierderile de clienți. Implementarea acestui sistem va permite companiei să adopte strategii proactive de retenție, optimizând astfel relația cu clienții și performanța afacerii.

Setul de date analizat conține 14 variabile, dintre care 11 sunt relevante pentru problema noastră.

**Variabilele setului de date și descrierile acestora**

|  |  |
| --- | --- |
| Conținut | Detalii |
| RowNumber | Indexul rândului – nesemnificativ |
| CustomerId | Identificator unic al clientului |
| Surname | Numele de familie |
| CreditScore | Scorul de credit al clientului, calculat pe baza istoricului său financiar |
| Geography | Țara sau regiunea de reședință a clientului, indicând piața geografică a băncii |
| Gender | Genul clientului, care poate fi masculin sau feminin |
| Age | Vârsta clientului |
| Tenure | Numărul de ani sau luni de când clientul este client al băncii |
| Balance | Suma de bani disponibilă în conturile clientului la un moment dat |
| NumOfProducts | Numărul de produse bancare utilizate de client |
| HasCrCard | Indică dacă clientul deține un card de credit (1) sau nu (0) |
| IsActiveMember | Indică dacă clientul este activ (1) sau nu (0) |
| EstimatedSalary | Salariul anual estimat al clientului |
| Exited | Variabila țintă, indicând dacă clientul a părăsit banca (1) sau nu (0) |

Ne propunem să realizăm un model de clasificare binară (0 – client care nu a părăsit banca (Exited=0), 1 – client care a părăsit banca(Exited=1)) utilizând cele 10 de variabile independente (se mai numesc și regresori/predictori).

# Probleme de clasificare în Machine Learning. Învățarea supervizată

De cele mai multe ori, problemele din sfera Machine Learning se împart în două mari categorii: învățarea supervizată (Supervised Learning) și nesupervizată (Unsupervised Learning). În cele ce urmează ne vom concentra pe descrierea metodologiei de lucru necesare primului tip de probleme.

Învățarea supervizată (Supervised Learning)

Scop: un model matematic este antrenat pe un set de date care include instanțe/observații etichetate de către un specialist. Scopul principal este ca algoritmul să găsească șabloane (tendințe) corecte în **setul de date de antrenament** pentru a reuși să clasifice instanțe care nu apar în acesta cu eticheta corespunzătoare. Altfel spus, algoritmul să reușească o generalizare a variabilei țintă (de multe ori, în industrie, denumită flag).

Modelele de Machine Learning (ML) sunt diverse, fiecare având un specific diferit și o strategie internă de implementare diferită. Printre principalele tipuri de modele putem enumera: regresia logistică, tehnici bazate pe vectori suport (Support Vector Machine – SVM), tehnici bazate pe arbori de decizie (Decision Trees) și tehnici de ansamblu (Ensemble Learning – Random Forest, Light Gradient Boosting Machine Classifier – LGBMClassifier etc.), acestea fiind doar câteva exemple de modele utilizate în practică.

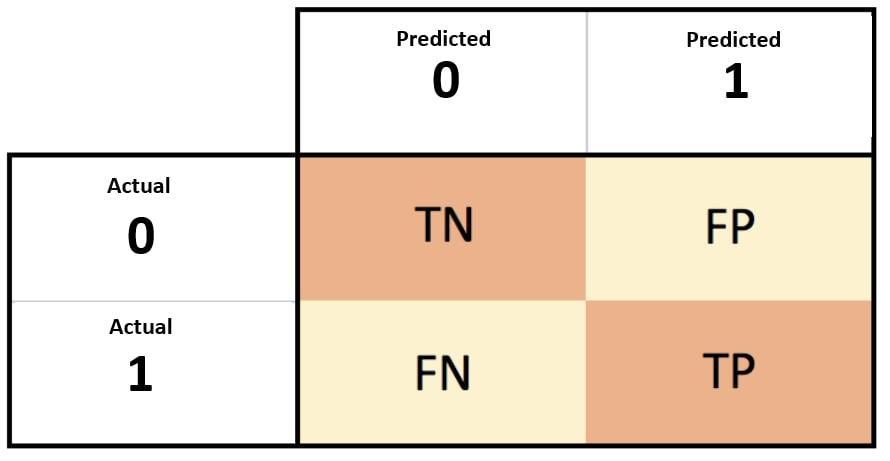
Problemele din sfera învățării supervizate se împart în 2 categorii:

* De*regresie***:** predicția unui rezultat de tip numeric (prezicerea prețului caselor în anii următori, evoluția prețului în criptomonede, alte exemple din econometrie)
* De *clasificare***:** predicția unui rezultat discret, adică etichetarea unor instanțe în categorii. Când în setul de date variabila target prezintă 2 categorii, problema se numește de clasificare binară (Binary Classification). Când există mai multe variante, lucrăm cu probleme de clasificare multiclasă (Multiclass Classification).

În cadrul învățării supervizate, setul de date inițial este împărțit/divizat (split) în 2 seturi de date individuale: setul de antrenament și setul de testare. În mod obligatoriu, observațiile din setul de antrenament (training set) sunt distincte de cele din setul de testare (test set). În practică, setul de date inițial este divizat în 3: setul de antrenament, de validare (validation set) și de test, fiecare conținând instanțe unice. Setul de validare este un instrument adjuvant în măsurarea indicatorilor de performanță pentru modelele de clasificare, în timp ce setul de testare este păstrat pentru a aplica modelul antrenat pe instanțe care nu au fost incluse nici în timpul antrenării, dar nici în timpul verificării.

Împărțirea setului de date în antrenament și test (train-test split) se realizează, de obicei, după o schemă standard, depinzând și de volumul de date disponibil data scientist-ului. De obicei, schema train-test split (volum%train-volum%test ) se realizează după procentele 70%-30% sau 80%-20%, existând variații de la aceste procente (se stabilesc empiric).

După antrenarea modelelor pe setul de antrenament are loc etapa de testare. În cazul problemelor de clasificare sunt folosiți mai mulți indicatori statistici care descriu capacitățile de generalizare a unui model. Un instrument foarte util în descrierea puterii de clasificare a modelelor este reprezentat de matricea de confuzie (Confusion Matrix). Forma generală a unei matrice de confuzie pentru problemele de clasificare binară este:



* Actual: reprezintă eticheta prestabilită de specialist pentru instanțe. Altfel spus, este de fapt eticheta reală a instanțelor din setul de test.
* Predicted: ceea ce returnează modelul ML, adică eticheta/flag-ul pe care modelul de ML o/îl consideră.
* TN: True Negative – adevărat negativ, numărul de instanțe care descrie faptul că modelul ML prezice corect instanțele etichetate cu 0.
* FP: False Pozitive – fals pozitiv, numărul de instanțe care sunt clasificate de modelul ML cu etichetă 1, în realitate ele fiind negative (0).
* FN: False Negative – fals negativ, instanțele clasificate de model cu 0, deși, în realitate, ele sunt pozitive (1).
* TP: True Positive – adevărat pozitiv, observațiile care sunt corect clasificate de modelul ML cu 1.

Pe baza tuturor acestor indicatori, putem descrie următoarele metrici de succes pentru un model ML:

* + 1. **Acuratețea (Accuracy)**: este măsura generală a performanței unui model și reprezintă proporția de predicții corecte (atât pozitive cât și negative) față de totalul cazurilor. Se calculează folosind formula:

* + 1. **Precizia (Precision)**: măsura acurateței predicțiilor pozitive. Reflectă proporția de cazuri etichetate ca pozitive de model care sunt de fapt pozitive. Formula este:
    2. **Senzitivitate (Recall)**: măsura capacității modelului de a identifica corect toate cazurile relevante (pozitive). Este important în scenarii unde ratarea cazurilor pozitive este critică (cum este și în cazul problemei noastre). Se calculează astfel:

* + 1. **Scorul F1 (F1-Score)**: este media armonică între precizie și senzitivitate. Oferă un echilibru între precision și recall și este util când avem o distribuție neregulată a claselor. Formula scorului F1 este:

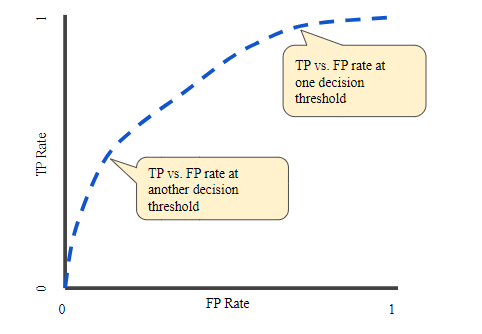
Modelele de ML nu oferă, de fapt, o clasă anume pentru instanțele pe care le clasifică, ci oferă o probabilitate de apartenență la una dintre clase. Aplicând un prag (threshold), obținem clasa corespunzătoare observației clasificate. De exemplu, dacă avem o probabilitate de 0.4 (40% șanse de apartenență la una dintre clase) cu un threshold prestabilit de 0.5 (50%) pentru o observație din setul de test, atunci acesteia i se va oferi categoria 0 (0.4 < 0.5). Dacă probabilitatea depășește acest prag, atunci observația va fi încadrată în categoria 1.

Threshold-ul de probabilitate este, de cele mai multe ori, stabilit de către Data Scientist în urma deciziilor de business și a performanțelor modelului. Se pot încerca mai multe variante pentru prag și se va realiza monitorizarea rezultatelor obținute.

În funcție de decizia de business, se dorește maximizarea unui anumit indicator din cei prezentați anterior. De exemplu, în cazul problemei noastre, se dorește obținerea unui Recall cât mai mare, dar și ca precizia să fie cât mai mare. Matematic, observând formula preciziei, FP trebuie să fie cât mai mic, iar în cazul Recall-ului, FN trebuie să fie minimizat.

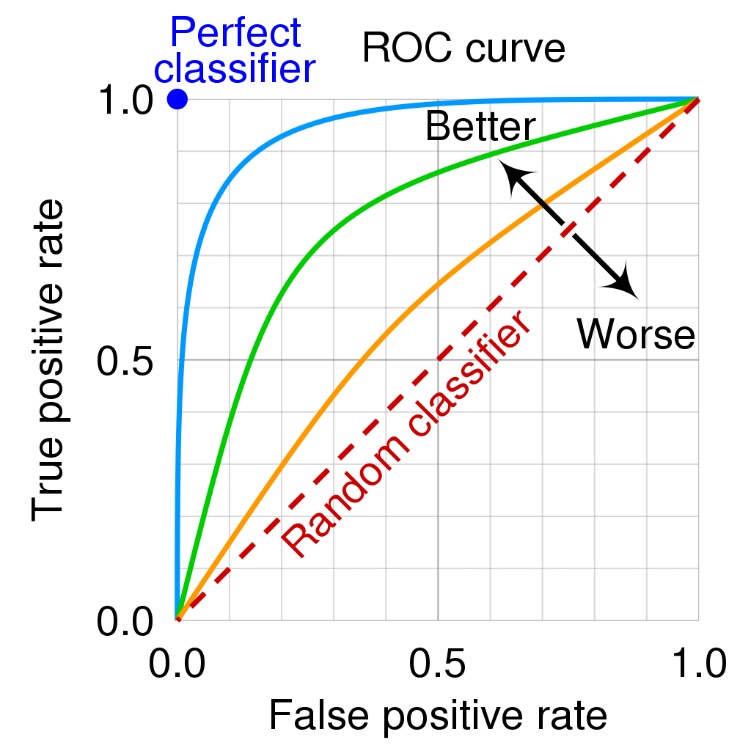
În practică, acest obiectiv este, adeseori, greu de atins. Între Precision și Recall există un compromis, denumit Precision-Recall Tradeoff, care exprimă faptul că, atunci când ne dorim atingerea unui nivel cât mai mare pentru un indicator, există posibilitatea ca celălalt să aibă o valoare nedorită. Astfel, ne dorim atingerea unui threhsold care să fie convenabil atât pentru optimizarea valorii Recall, dar și pentru a obține o precizie adecvată.

Un instrument util pentru a stabili pragul de decizie este reprezentat de curba ROC (Receiver Operating Characteristic).



Pe curba ROC putem observa procentele True Positive (TPR) și False Positive (FPR). De obicei, threshold-ul se alege astfel încât ambii indicatori să aibă valori optime (TPR cât mai mare și FPR cât mai mic).

Aria de sub curba ROC (AUC – Area Under the ROC Curve) este, de asemenea, un indicator de succes pentru modelele de clasificare. Cu cât valoarea acesteia este mai mare, cu atât performanța modelului este mai bună. Un clasificator perfect are AUC = 1 (100%), în timp ce clasificatorii slabi au valori AUC, de obicei, sub 0.6.



# Implementarea de modele ML de clasificare binară în limbajul SAS

## Citirea datelor

\*Pas 1 - Crearea setului de date;

proc import datafile='/home/anca.andreescu/Curs5/Churn.csv'

out=work.churn

dbms=csv

replace;

run;

/\*Pas 2- Eliminarea anumitor variabile care nu sunt de interes\*/

data work.churn;

set work.churn;

drop RowNumber CustomerId Surname;

run;

## Etapa de curățare a datelor – tratarea valorilor lipsă

În funcție de tipul variabilei ce conține valori lipsă, avem următoarele strategii de tratare a acestora:

1. Pentru variabilele numerice înlocuim, de obicei, cu media/mediana variabilei în cauză. Atenție, media este un indicator statistic sensibil la valori extreme!
2. Pentru variabilele categoriale/nominale folosim preponderent valoarea modală (cea mai întâlnită valoare din seria variabilei respective).

Pentru a verifica dacă în setul de date încărcat există valori lipsă, folosim atât procedura FREQ (pentru a verifica lipsa valorilor în rândul variabilelor categoriale), cât și procedura MEANS (pentru variabilele numerice).

\*Pas 3 - încărcăm variabilele numerice și categoriale în liste separate;

proc sql noprint;

select name into :var\_categ separated by ' ' from dictionary.columns

where libname='WORK' and memname='CHURN' and type='char';

select name into :var\_num separated by ' ' from dictionary.columns

where libname='WORK' and memname='CHURN' and type='num';

quit;

Observăm că în codul anterior a fost folosită tabela dictionary. Sistemul SAS generează în timpul rulării, și menține pe parcursul unei sesiuni SAS, informații importante despre bibliotecile, seturile de date, cataloagele, indexurile, macrocomenzile, opțiunile de sistem, titluri etc. într-o colecție de tabele disponibile doar pentru citire, numite tabele dicționar. Cu toate că sunt numite tabele, tabelele dicționar nu sunt tabele reale. Informațiile sunt generate automat în timpul rulării și conținutul este disponibil odată ce o sesiune SAS este începută. Tabelele dicționar și conținutul acestora permit accesarea și monitorizarea cu ușurință a activităților unei sesiuni SAS, prin intermediul interogărilor. În acest exemplu, declarațiile where sunt folosite pentru a selecta variabilele de tip caracter (type='char') și, respectiv, numeric (type='num') din setul de date specificat prin memname și care este stocat în biblioteca descrisă prin libmane.

Pentru vizualizarea rezultatelor utilizăm comenzile:

/\*Pas 4 - Afisarea informatiilor despre valorile lipsa\*/

proc freq data=work.churn;

tables &var\_categ / missing;

run;

proc means data=work.churn n nmiss;

var &var\_num;

run;

/\*Pas5 - Tabele de frecvente pentru variabila Exited\*/

PROC FREQ DATA=work.churn;

TABLES Exited /nocum nopercent;

RUN;

Nu există valori lipsă în rândul variabilelor categoriale.

Variabilele numerice Tenure, Balance și EstimatedSalary au valori lipsă.

1. Înlocuim valorile lipsă din coloanele Balance și EstimatedSalary cu media folosind procedura stdize. Rolul acestei proceduri este să standardizeze una sau mai multe variabile numerice dintr-un set de date SAS. Pentru standardizare se pot folosi o serie de metode bine-cunoscute precum media, mediana, abaterea standard sau intervalul.

/\*Pas 6.1- Vizualizarea distributiilor variabilelor cu valori lipsa\*/

proc univariate data=work.churn;

var Balance EstimatedSalary Tenure;

histogram;

run;

/\*Pas 6.2 - Înlocuirea valorilor lipsă din coloanele Balance si Estimated cu media\*/

proc stdize data=work.churn out=work.churn

method=mean reponly;

var Balance EstimatedSalary;

run;

1. Înlocuim valorilor lipsă din coloana Tenure cu mediana folosind aceeași procedură:

/\*Pas 7- Înlocuirea valorilor lipsă din coloana Tenure cu mediana\*/

proc stdize data=work.churn out=work.churn

method=median reponly;

var Tenure;

run;

/\*Pas 8- Reluam vizualizarea variabilelor numerice\*/

proc means data=work.churn n nmiss;

var &var\_num;

run;

Observăm că nu mai avem valori lipsă.

## Analiza exploratorie a datelor

Analiza exploratorie a datelor (EDA) ajuta la identificarea pattern-urilor și a relațiilor dintre variabile înainte de a construi modelele predictive. Permite descoperirea variabilelor importante care influențează rata de părăsire, detectarea valorilor extreme sau a imbalansării claselor și înțelegerea distribuției datelor. Prin analiza vizuală si statistică a datelor putem lua decizii privind preprocesarea datelor, selecția caracteristicilor și interpretarea modelelor de predicție.

/\*Pas 9 - Analiza exploratorie a datelor\*/

/\* Vizualizarea distributiei scorului de credit \*/

proc sgplot data=work.churn;

/\* Histograma pentru CreditScore \*/

histogram CreditScore / binstart=0 binwidth=10 showbins;

/\* Curba de densitate KDE (Kernel Density Estimation) \*/

density CreditScore / type=kernel;

/\* Titlu si etichete pentru axe \*/

title "Distributia Scorului de Credit";

xaxis label="Scor de Credit";

yaxis label="Frecventa";

/\* Format pentru axa X (optional) \*/

format CreditScore comma12.0;

run;

/\* Vizualizarea distribuției vârstei (Age)\*/

proc sgplot data=work.churn;

/\* Histogramă pentru Age \*/

histogram Age / binstart=0 binwidth=5 transparency=0.3 fillattrs=(color=blue) showbins;

/\* Curba de densitate KDE \*/

density Age / type=kernel lineattrs=(color=red thickness=2 pattern=solid);

/\* Linie de referință pentru vârsta medie (calculată anterior) \*/

refline 40 / axis=x lineattrs=(color=green pattern=shortdash) label="Varsta medie";

/\* Titlu și etichete pentru axe \*/

title "Distributia varstei clientilor";

xaxis label="Varsta (ani)" grid;

yaxis label="Frecventa" grid;

run;

/\* Vizualizarea distributiei soldului (Balance) \*/

proc sgplot data=work.churn;

/\* Histograma pentru Balance \*/

histogram Balance / binstart=0 binwidth=10000 transparency=0.3 fillattrs=(color=orange) showbins;

/\* Curba de densitate KDE \*/

density Balance / type=kernel lineattrs=(color=red thickness=2 pattern=solid);

/\* Linie de referinta pentru soldul mediu (calculata anterior) \*/

refline 80000 / axis=x lineattrs=(color=green pattern=shortdash) label="Sold Mediu";

/\* Titlu si etichete pentru axe \*/

title "Distributia Soldului Clientilor (Balance)";

xaxis label="Sold (Balance)" grid;

yaxis label="Frecventa" grid;

run;

/\* Vizualizarea distributiei pentru NumOfProducts si Exited \*/

/\* Cream tabelul de frecventa \*/

proc freq data=work.churn;

tables NumOfProducts\*Exited / out=FreqData noprint;

run;

/\* Cream graficul combinat \*/

proc sgplot data=FreqData;

vbar NumOfProducts / response=Count group=Exited groupdisplay=cluster datalabel;

xaxis label="Number of Products";

yaxis label="Count";

title "Distributia NumOfProducts si Exited combinate";

run;

/\* Rata de parasire a bancii in functie de geografie \*/

proc sgplot data=work.churn;

vbar Geography / group=Exited groupdisplay=cluster stat=freq;

title "Rata de Parasire a Bancii in Functie de Geografie";

xaxis label="Geografie";

yaxis label="Numar de Clienti";

run;

/\* Rata de parasire a bancii in functie de gen \*/

proc sgplot data=work.churn;

vbar Gender / group=Exited groupdisplay=cluster stat=freq;

title "Rata de Parasire a Bancii in Functie de Gen";

xaxis label="Gen";

yaxis label="Numar de Clienti";

run;

/\*Pas 10 - Feature Creation\*/

data work.churn;

set work.churn;

BalanceToSalaryRatio = Balance / EstimatedSalary;

BalanceToProductRatio = Balance / NumOfProducts;

TenureToAge = Tenure / Age;

run;

## Crearea de variabile noi (Feature creation)

Crearea de variabile noi, în general, permite modelelor predictive să capteze relații mai subtile în date, care nu sunt evidente în variabilele originale. Pot îmbunătăți capacitatea modelului de a face predicții mai precise și de a identifica clienții cu risc ridicat de a părăsi banca. Aceste variabile adiționale vor fi ulterior incluse în modelele predictive pentru a evalua impactul lor asupra preciziei clasificării.

/\*Pas 10 - Feature Creation\*/

data work.churn;

set work.churn;

BalanceToSalaryRatio = Balance / EstimatedSalary;

BalanceToProductRatio = Balance / NumOfProducts;

TenureToAge = Tenure / Age;

run;

**Raportul dintre sold și salariu**

Variabila ne permite sa identificăm dacă acei clienți care au un sold mare în raport cu venitul lor sunt mai predispuși să părăsească banca. De exemplu dacă un client are un sold foarte mare dar un venit scăzut acest lucru ar putea indica o reținere în utilizarea fondurilor sau o posibilă instabilitate financiară.

**BalanceToProductRatio**

Aceasta variabilă poate indica dacă un client cu un sold mare utilizează un număr redus de produse bancare, ceea ce poate sugera o lipsă de implicare în cadrul băncii. Astfel de clienți ar putea fi mai predispuși sa părăsească banca, iar pe baza unui astfel de indicator se pot dezvolta diverse strategii de retenție.

**TenureToAge**

Evidențiază situația în care clienții care sunt clienți fideli de la o vârstă fragedă sunt mai puțin predispuși să părăsească banca. De asemenea, poate ajuta la identificarea grupurilor de clienți care au avut relații scurte cu banca, dar care au deja o vârstă mai înaintată, ceea ce ar putea indica un risc mai mare de abandon.

## Transformarea variabilelor non-numerice în variabile numerice

În general, algoritmii de Machine Learning sunt capabili să lucreze doar pe date de tip numeric. Există, însă, multe situații în care variabilele din seturile de date de intrare sunt de tip non-numeric (categorial, nominal). Vom transforma corespunzător aceste variabile pentru a reprezenta input pentru algoritmii de ML.

/\* Pas 11 - Encodare pentru variabila Gender \*/

data work.churn;

set work.churn;

/\* Crearea variabilei intermediare pentru Gender \*/

if Gender = 'Male' then Gender\_encoded = 1;

else if Gender = 'Female' then Gender\_encoded = 0;

else Gender\_encoded = .; /\* Tratarea valorilor lipsa sau invalide \*/

run;

/\* Pas 12 - Encodare pentru variabila Geography \*/

data work.churn;

set work.churn;

/\* Crearea variabilei intermediare pentru Geography \*/

length Geography\_encoded 8; /\* Asiguram ca este o variabila numerica \*/

if Geography = 'France' then Geography\_encoded = 0;

else if Geography = 'Spain' then Geography\_encoded = 1;

else if Geography = 'Germany' then Geography\_encoded = 2;

else Geography\_encoded = .; /\* Tratarea valorilor necunoscute \*/

run;

/\* Verificarea rezultatului \*/

proc print data=work.churn(obs=10);

run;

/\* Pas 13 - Stergem coloanele initiale si le redenumim pe cele codificate\*/

data work.churn;

set work.churn;

drop Gender Geography;

rename Gender\_encoded=Gender Geography\_encoded=Geography;

run;

proc print data=work.churn(obs=10);

run;

## Calculul matricei de corelație în SAS

Matricea de corelație este un instrument util pentru a măsura în ce măsură variabilele din setul de date sunt corelate între ele. Coeficientul Pearson măsoară relația liniară dintre două variabile, iar coeficientul Spearman măsoară relația monotonă, adică dacă o creștere într-o variabila este asociata cu o creștere sau descreștere în cealaltă, indiferent dacă relația este liniară. Valorile coeficienților variază între -1 si 1, unde:

* valoare apropiata de 1 indica o corelație pozitivă puternică;
* valoare apropiata de -1 indică o corelație negativă puternică;
* valoare apropiata de 0 indică o corelație slabă sau inexistentă.

Analiza corelațiilor ajută la identificarea variabilelor care au cel mai mare impact asupra părăsirii băncii de către client. Sunt utile pentru selecția variabilelor în modelele predictive și pentru înțelegerea factorilor care influențează comportamentul clienților. Variabilele cu corelații mai mari pot fi utilizate în modele pentru a îmbunătăți predicția variabilei dependente, iar strategiile bancare pot fi ajustate în funcție de rezultate.

În SAS ne propunem să calculăm corelația dintre numerice și variabila țintă, ***Exited***. Utilizăm procedura ***corr***:

/\*Pas 14 - Matricea de corelatie Spearman\*/

/\* Calcularea corelațiilor Spearman între variabilele numerice și Exited \*/

proc corr data=work.churn spearman outp=correlation\_matrix nosimple;

var CreditScore Age Tenure Balance NumOfProducts HasCrCard

IsActiveMember EstimatedSalary BalanceToSalaryRatio

BalanceToProductRatio TenureToAge;

with Exited;

run;

/\* Vizualizarea primei părți a matricei de corelație \*/

proc print data=correlation\_matrix(obs=20);

run;

## Oversampling

În problemele de Machine Learning, distribuția variabilei dependente joacă un rol foarte important în ceea ce privește construirea modelelor. Există două tipologii de probleme, în funcție de această distribuție:

1. probleme balansate, în care numărul de instanțe din categoriile de clasificat sunt relativ apropiate;
2. probleme imbalansate, când există un dezechilibru puternic în categoriile de clasificat.

Analizăm ca structură setul de date inițial. Vom analiza grafic distribuția variabilei Exited pentru a studia imbalansarea celor două clase.

/\* Pas 15 - Descriere inainte de oversampling\*/  
proc contents data=work.churn;  
run;  
  
/\* Verificăm distribuția inainte de oversampling \*/  
proc freq data=work.churn;  
 tables Exited Gender Geography / nocum;

run;

Se poate observa din distribuția datelor că există o majoritate a instanțelor care aparțin clasei 0, în timp ce un număr semnificativ mai redus de clienți care au părăsit banca a fost raportat (7963 vs 2037). Acest fenomen de imbalansare a claselor este des întâlnit în problemele de clasificare și poate influența performanța modelelor, determinându-le să favorizeze clasa majoritară. Există două tehnici de a rezolva acest fenomen, și anume ***oversampling*** (adăugarea de instanțe sintetice în setul de date, create prin metode statistice specifice pentru a echilibra cele două clase) și ***undersampling***, procedeul complementar primei abordări.

Atunci când distribuția claselor nu este echilibrată, modelul rezultat poate avea o tendință de a favoriza categoria predominantă. Acest lucru se întâmplă deoarece, în timpul antrenării, algoritmul întâlnește mai des exemple din această clasă, ceea ce face mai dificilă recunoașterea și învățarea caracteristicilor specifice cazurilor mai rare. În practică, astfel de situații sunt des întâlnite și pot afecta acuratețea predicțiilor.

Pentru a rezolva imbalansarea am folosit un algoritm scris în Python, deoarece aplicarea unui algoritm de imbalansare în SAS nu a dus la rezultate favorabile pentru setul nostru de date (găsiți la final, în anexă, codul SAS cu care am încercat inițial).

SMOTE (Synthetic Minority Over-sampling Technique) este o tehnică utilizată pentru a rezolva problema seturilor de date dezechilibrate. Aceasta creează exemple sintetice pentru clasa minoritară, ajutând modelele de machine learning să învețe mai bine tiparele din datele rare. În loc să copieze pur și simplu datele existente, SMOTE generează noi exemple prin interpolarea dintre un punct de date existent și cei mai apropiați vecini ai săi. Practic, selectează un punct din clasa minoritară, identifică vecinii cei mai apropiați (de obicei cinci) și creează puncte noi între acesta și vecinii săi. Astfel, modelul devine mai capabil să recunoască și să clasifice corect exemplele din clasa minoritară.

SMOTETomek este o tehnică care combină două metode puternice: SMOTE și Tomek Links. După ce SMOTE creează exemple sintetice pentru clasa minoritară, Tomek Links intervine pentru a curăța datele. Aceasta identifică perechile de puncte de date care sunt foarte apropiate și aparțin unor clase diferite, numite „Tomek links”. Astfel de puncte creează ambiguitate pentru model și sunt considerate zgomot. SMOTETomek elimină aceste perechi pentru a îmbunătăți claritatea granițelor dintre clase. Această combinație ajută atât la echilibrarea claselor, cât și la curățarea setului de date, rezultând un model de clasificare mai robust și mai precis.

from imblearn.combine import SMOTETomek

smk = SMOTETomek(random\_state=42)

X\_oversampled, y\_oversampled = smk.fit\_resample(X, y)

X\_oversampled['Exited'] = y\_oversampled.copy()

df\_oversampled = X\_oversampled.copy()

df\_oversampled.to\_csv('oversampled\_dataset.csv')

/\* Pas 16.1 - Importul setului de date pe care am aplicat oversampling in Python\*/

proc import datafile='/home/anca.andreescu/Curs5/oversampled.csv'

out=work.churn\_oversampled

dbms=csv

replace;

run;

Variabilele index și VAR1 apar în setul de date din cauza procesării în Python. index este generat automat atunci când fișierul este salvat, iar VAR1 este adăugată de SMOTETomek pentru a indica dacă o observație provine din setul original sau a fost sintetizată. Deoarece aceste variabile nu sunt relevante pentru analiza noastră, le vom elimina.

/\* Pas 16.2 – Eliminarea variabilelor adaugate in oversampling\*/

DATA work.churn\_oversampled;

SET work.churn\_oversampled;

DROP index Var1; /\* Eliminăm coloanele nedorite \*/

RUN;

PROC PRINT DATA=work.churn\_oversampled (OBS=5);

RUN;

## Scalare și train-test split

Folosim **scalarea Min-Max** în Machine Learning pentru a aduce variabilele numerice într-un interval comun (de obicei între **0 și 1**), îmbunătățind astfel performanța modelelor. Se aplică pentru a îmbunătăți performanța algoritmilor de învățare automată, a accelera convergența modelelor iterative și a preveni influența disproporționată a variabilelor cu valori mari. Aceasta este esențială pentru algoritmi precum regresia logistică care sunt sensibili la diferențele de magnitudine dintre caracteristici.

Formula utilizată este:

/\* Pas 18 -Aplicăm scalarea Min-Max \*/

proc stdize data=work.churn out=work.churn method=range;

var \_numeric\_;

run;

proc print data=work.churn(obs=10);

run;

/\*Distributia de frecventa a variabilei Exited\*/

proc sgplot data=work.churn;

vbar Exited / datalabel;

title "Distributia variabilei Exited";

run;

Împărțirea setului de date în set de antrenament (train set) și set de testare (test set) este un pas esențial în procesul de modelare, permițând evaluarea corectă a performanței algoritmului. Vom aloca 70% dintre observații pentru antrenare și 30% pentru testare, utilizând o tehnică statistică numită eșantionare. În SAS, pentru a realiza această împărțire, folosim procedura PROC SURVEYSELECT, aplicând metoda de eșantionare simplă aleatoare (Simple Random Sample – SRS). Setarea parametrului SEED asigură reproductibilitatea rezultatelor, iar utilizarea opțiunii RATE permite specificarea proporției de 70% pentru setul de antrenament.

Prin opțiunea OUTALL includem toate observațiile din setul de date de intrare specificat în DATA în setul de date de ieșire din declarația OUT. În mod implicit, setul de date de ieșire include numai acele observații selectate pentru eșantion. Dacă specificăm opțiunea OUTALL, setul de date de ieșire va include toate observațiile din setul de date de intrare și, de asemenea, va conține o variabilă care indică starea de selecție a fiecărei observații într-o variabilă numită SELECTED. Variabila SELECTED este egală cu 1 pentru o observație care este selectată pentru eșantion și cu 0 pentru o observație care nu este selectată.

/\*Pas 19 - train-test split\*/

/\* Definim proportia de impartire in antrenament-test \*/

%let train\_prop = 0.7;

/\* Impartim setul de date \*/

proc surveyselect data=work.churn out=work.train outall

method=srs /\* Simple Random Sample \*/

rate=&train\_prop

seed=12345;

run;

data work.train work.test;

set work.train;

if selected then output work.train;

else output work.test;

run;

În continuare, selectăm automat toate variabilele din setul de date CHURN, exceptând variabila țintă (Exited), și le stocăm într-o macrovariabilă &indepVars. Acest pas este necesar pentru a simplifica procesul de modelare, permițând utilizarea listei de variabile independente fără a le introduce manual

/\*Pas 20 - plasam variabilele independente intr-o macrovariabila\*/

proc sql noprint;

select name into :indepVars separated by ' '

from dictionary.columns

where libname='WORK' and memname='CHURN' and name ne 'Exited';

quit;

%put &indepVars;

## Antrenarea modelelor de Machine Learning

1. Modelul de regresie logistică (Logistic Regression – LR)

Reprezintă unul dintre modele de bază din categoria estimatorilor de clasificare, rezultatul generat de acesta fiind o probabilitate ca o instanță să aparțină la una dintre clasele specificate în problemă, adică un rezultat în intervalul (0,1)[[1]](#footnote-1). Vom afișa, de asemenea, și curba ROC pentru acest model folosind opțiunea plots = roc.

Proc logistic construiește un model de regresie logistică pentru a prezice probabilitatea ca un client să părăsească serviciul (Exited=1). Opțiunea descending indică faptul că modelul prezice probabilitatea pentru valoarea 1 a variabilei țintă. Se salvează modelul logistic pentru utilizare ulterioară în setul de date logitmodel. Predicțiile sunt generate pentru setul de test cu ajutorul opțiunii score, iar rezultatele sunt salvate în test\_pred. De asemenea, sunt generate date pentru analiza curbei ROC, utile pentru evaluarea performanței modelului.

/\*Pas 21 - Regresie logistica caz undersampled \*/

proc logistic data=work.train descending outmodel=work.logitmodel;

model Exited(event='1') = &indepVars;

score data=work.test out = test\_pred outroc=vroc;

run;

După ce modelul produce probabilitățile (P\_1 pentru clasa 1), acestea sunt transformate în predicții binare folosind două praguri diferite:

* pragul de 0.5 este un standard obișnuit, care presupune o clasificare echilibrată între clase;
* pragul de 0.3 este ajustat empiric, pentru a optimiza metrici de performanță (precum recall-ul sau precizia), mai ales dacă datele sunt dezechilibrate.

În problemele cu dezechilibru între clase (de exemplu, când doar un mic procent din clienți părăsesc serviciul), un prag de 0.5 poate să nu fie optim. Ajustarea pragului permite maximizarea performanței modelului în funcție de prioritățile business-ului (ex: reducerea falselor negative). În acest caz se creează două variabile de clasificare:

* pred\_class folosește un prag standard de 0.5 (dacă probabilitatea P\_1 este mai mare de 0.5, clientul este prezis ca "va părăsi").
* pred\_class\_2 folosește un prag ajustat empiric de 0.3 pentru a optimiza performanța modelului.

/\* Pas 22 - Cream predictii pe baza unor praguri (stabilite in mod empiric)\*/

data work.test\_pred;

set work.test\_pred;

pred\_class = (P\_1 > 0.5);

pred\_class\_2 = (P\_1 > 0.3);

run;

Vom utiliza proc freq pentru a genera matricele de confuzie, care compară predicțiile modelului cu valorile reale ale variabilei Exited. Matricea de confuzie arată câte cazuri au fost prezise corect și câte au fost eronate (adevărate pozitive, adevărate negative, false pozitive și false negative).

/\* Pas 23 - Matricea de confuzie pentru pragul 1\*/

title 'Matrice de confuzie pentru modelul de regresie logistica prag 1';

proc freq data=work.test\_pred;

tables Exited \* pred\_class / nocol;

run;

/\* Pas 24 - Matricea de confuzie pentru pragul 2\*/

title 'Matrice de confuzie pentru modelul de regresie logistica prag 2';

proc freq data=work.test\_pred;

tables Exited \* pred\_class\_2 / nocol;

run;

În modelele de clasificare binară, modelul nu returnează direct o clasă (0 sau 1), ci o probabilitate care indică cât de probabil este ca o observație să aparțină clasei pozitive (Exited = 1, în cazul nostru).

De exemplu, pentru un client, modelul poate prezice o probabilitate de 0.72 că acesta va părăsi serviciul. Pentru a transforma această probabilitate într-o clasă binară (0 sau 1), trebuie să stabilim un prag de decizie (threshold). Presupunem că alegem pragul standard de 0.5. Dacă probabilitatea prezisă este mai mare decât 0.5, clientul este clasificat ca 1 (va părăsi serviciul). Dacă este mai mică sau egală cu 0.5, clientul este clasificat ca 0 (nu va părăsi serviciul).

De obicei, se pornește de la un prag standard pentru clasificare binară, ce reprezintă o presupunere că modelul este la fel de încrezător în ambele clase, adică tratăm costurile pentru erori de clasificare (false positive și false negative) ca fiind egale. Acesta funcționează bine pentru seturi de date echilibrate (când avem un număr similar de cazuri pozitive și negative), deci când costurile pentru erori sunt similare (ex: greșeala de a clasifica un client loial ca fiind predispus la părăsire are același impact ca și inversul).

Urmează apoi selectarea, în mod empiric, a unui prag mai mic, spre exemplu 0.3. Dacă probabilitatea prezisă este mai mare decât 0.3, clientul este clasificat ca 1. Acest prag mai mic crește sensibilitatea (recall-ul) modelului, adică detectează mai mulți clienți care ar putea părăsi serviciul. În multe probleme de business, nu toate erorile de clasificare sunt la fel de costisitoare. În cazul retenției clienților, este mai costisitor să ratezi un client care chiar va pleca (fals negativ) decât să obții alertă pentru un client care ar fi rămas oricum (fals pozitiv). Ajustarea pragului către 0.3 crește numărul de clienți identificați ca fiind „în pericol,” ceea ce este de dorit pentru strategii de retenție.

1. Modelul Random Forest

Random Forest este un algoritm de învățare automată bazat pe arbori de decizie, utilizat atât pentru clasificare, cât și pentru regresie. Funcționează prin combinarea mai multor arbori de decizie (o „pădure” de arbori) pentru a obține predicții mai precise și mai stabile. Fiecare arbore este antrenat pe un eșantion diferit din datele inițiale (bootstrap sampling), iar decizia finală se ia prin vot majoritar (pentru clasificare) sau prin media predicțiilor (pentru regresie).

• Sunt utilizati mai mulți arbori de decizie pentru a îmbunătăți precizia și a reduce fenomernul de supra-adaptare.

• Fiecare arbore este antrenat pe un subset aleatoriu de date, ceea ce asigură diversitate și robustețe.

• La fiecare împărțire a unui nod, este luat în considerare doar un subset aleatoriu de variabile, ceea ce îmbunătățește generalizarea.

• Fiind o combinatie de modele, Random Forest nu este sensibil la anomalii și funcționează bine chiar și cu date lipsa sau care introduc zgomot.

Modelul Random Forest este antrenat pentru a prezice dacă un client a părăsit serviciul (Exited), folosind un set de date de antrenament. Cu 500 de arbori de decizie, o adâncime maximă de 10 și frunze de cel puțin 15 observații, modelul utilizează variabilele independente definite în &indepVars. Predicțiile sunt salvate în setul de date scored, iar modelul antrenat este stocat în fișierul rf.bin,

/\* Random Forest caz undersampled \*/

/\* Pas 25 - Construire si antrenare RF \*/

proc hpforest data=work.train seed=123 maxtrees=500 maxdepth=10 leafsize=15;

target Exited/ level=binary;

input &indepVars;

score out = scored;

save file = '/home/anca.andreescu/Curs5/rf.bin';

run;

Modelul Random Forest salvat în fișierul rf.bin este utilizat pentru a genera predicții pe setul de date work.test. Valorile reale ale variabilei Exited sunt păstrate pentru identificare, iar rezultatele predicțiilor sunt stocate în setul de date work.test\_pred\_rf.

/\* Pas 26 - Generare predictii \*/

proc hp4score data=work.test;

id Exited;

score file='/home/anca.andreescu/Curs5/rf.bin'

out=work.test\_pred\_rf;

run;

Modelul Random Forest aplicat pe setul de test generează probabilități pentru clasa Exited=1. Pe baza acestor probabilități, se construiesc două seturi de predicții: pred\_rf\_05, unde un client este considerat ieșit dacă probabilitatea depășește 0.5, și pred\_rf\_03, unde pragul este mai scăzut, la 0.3. Rezultatele sunt salvate în setul de date predictii\_rf.

/\* Pas 27 - Creare predictii si analize pe baza pragurilor stabilite empiric \*/

data predictii\_rf;

set work.test\_pred\_rf;

pred\_rf\_05 = (P\_Exited1 > 0.5);

pred\_rf\_03 = (P\_Exited1 > 0.3);

run;

Se construiesc matrice de confuzie pentru evaluarea performanței modelului Random Forest pentru cele două praguri de decizie diferite.

/\* Pas 28 - Matricea de confuzie pentru pragul 1\*/

title 'Matrice de confuzie pentru modelul arbore de decizie, threshold 0.5';

proc freq data=predictii\_rf;

tables Exited \* pred\_rf\_05 / nocol;

run;

/\* Pas 29 - Matricea de confuzie pentru pragul 2\*/

title 'Matrice de confuzie pentru modelul arbore de decizie, threshold 0.3';

proc freq data=predictii\_rf;

tables Exited \* pred\_rf\_03 / nocol;

run;

Adăugăm predicțiile modelului Random Forest în setul de date work.test prin merge. Apoi, prin PROC LOGISTIC, se evaluează performanța modelului folosind curba ROC, unde probabilitatea estimată P\_Exited1 este utilizată ca predictor.

/\* Pas 30 - Adăugarea predicțiilor la setul de date de test\*/

data work.test;

merge predictii\_rf;

run;

/\* Pas 31 - Generarea curbei ROC\*/

proc logistic data=work.test;

model Exited(event='1') = P\_Exited1 / nofit;

roc 'rf' pred=P\_Exited1;

ods select ROCcurve;

run;

## Utilizarea modelelor de ML pe datele balansate

Aplicăm cei doi algoritmi de ML, de data aceasta pe datele balansate. Vom relua pașii urmați pe datele originale, pornind de la pasul în care facem scalarea Min-Max.

/\*Varianta 2 - Dupa oversampling\*/

/\* Pas 32 - Aplicăm scalarea Min-Max \*/

proc stdize data=work.churn\_oversampled out=work.churn\_oversampled method=range;

var \_numeric\_;

run;

proc print data=work.churn\_oversampled(obs=10);

run;

/\*Distributia de frecventa a variabilei Exited\*/

proc sgplot data=work.churn\_oversampled;

vbar Exited / datalabel;

title "Distribution of Exited Variable";

run;

/\*Pas 33 - train-test split -- oversampling\*/

/\* Definim proportia de impartire in antrenament-test \*/

%let train\_prop = 0.7;

/\* Impartim setul de date \*/

proc surveyselect data=work.churn\_oversampled out=work.train outall

method=srs /\* Simple Random Sample \*/

rate=&train\_prop

seed=12345;

run;

data work.train work.test;

set work.train;

if selected then output work.train;

else output work.test;

run;

/\*Pas 34 - plasam variabilele independente intr-o macrovariabila\*/

proc sql noprint;

select name into :indepVars separated by ' '

from dictionary.columns

where libname='WORK' and memname='CHURN\_OVERSAMPLED' and name ne 'Exited';

quit;

%put &indepVars;

/\*Pas 35 - Regresie logistica caz oversampled \*/

proc logistic data=work.train descending outmodel=work.logitmodel;

model Exited(event='1') = &indepVars;

score data=work.test out = test\_pred outroc=vroc;

run;

/\* Pas 36 - Cream predictii pe baza unor praguri (stabilite in mod empiric)\*/

data work.test\_pred;

set work.test\_pred;

pred\_class = (P\_1 > 0.5);

pred\_class\_2 = (P\_1 > 0.302);

run;

/\* Pas 37 - Matricea de confuzie pentru pragul 1\*/

title1 'Matrice de confuzie pentru modelul de regresie logistica prag 1';

proc freq data=work.test\_pred;

tables Exited \* pred\_class / nocol;

run;

/\* Pas 38 - Matricea de confuzie pentru pragul 2\*/

title2 'Matrice de confuzie pentru modelul de regresie logistica prag 2';

proc freq data=work.test\_pred;

tables Exited \* pred\_class\_2 / nocol;

run;

/\* Random Forest caz oversampled \*/

/\* Pas 39 - Construire si antrenare RF \*/

proc hpforest data=work.train seed=123 maxtrees=500 maxdepth=10 leafsize=15;

target Exited/ level=binary;

input &indepVars;

score out = scored;

save file = '/home/anca.andreescu/Curs5/rf.bin'; \*aici sa puneti dvs calea unde salvati cand lucrati

run;

/\* Pas 40 - Generare predictii \*/

proc hp4score data=work.test;

id Exited;

score file='/home/anca.andreescu/Curs5/rf.bin'

out=work.test\_pred\_rf;

run;

/\* Pas 41 - Creare predictii si analize pe baza pragurilor stabilite empiric \*/

data predictii\_rf;

set work.test\_pred\_rf;

pred\_rf\_05 = (P\_Exited1 > 0.5);

pred\_rf\_03 = (P\_Exited1 > 0.3);

run;

/\* Pas 42 - Matricea de confuzie pentru pragul 1\*/

title 'Matrice de confuzie pentru modelul arbore de decizie, threshold 0.5';

proc freq data=predictii\_rf;

tables Exited \* pred\_rf\_05 / nocol;

run;

/\* Pas 43 - Matricea de confuzie pentru pragul 2\*/

title 'Matrice de confuzie pentru modelul arbore de decizie, threshold 0.3';

proc freq data=predictii\_rf;

tables Exited \* pred\_rf\_03 / nocol;

run;

/\* Pas 44 - Adăugarea predicțiilor la setul de date de test\*/

data work.test;

merge predictii\_rf;

run;

/\* Pas 45 - Generarea curbei ROC\*/

proc logistic data=work.test;

model Exited(event='1') = P\_Exited1 / nofit;

roc 'rf' pred=P\_Exited1;

ods select ROCcurve;

run;

**Notă**

Datele de intrare au fost adaptate după setul de date Bank Customer Churn Prediction preluat de pe Kaggle și disponibil la:

https://www.kaggle.com/datasets/shubhammeshram579/bank-customer-churn-prediction/data

1. https://towardsdatascience.com/logistic-regression-detailed-overview-46c4da4303bc [↑](#footnote-ref-1)