Problemele datelor este deosebirea descrierei de realitate si distribuirea neuniforma intre clasele negative si pozitive in 6 miliaone de randuri . Scopul acestei analize este de a rezolva aceste neajunsuri, alegand un model corespunzator de machine-learning si sa curatim datelele.

Care tranzactii sunt fraudulente?

Din 5 tipuri de tranzactii, frauda apare numai in doua din ele : TRANSFER unde banii is trimisi catra utilizator/fraudator in cash. De notat ca numarul transferurilor fraudulente TRANSFER aproape este egal cu numarul de incasari CASH\_OUT. Conform datasetului frauda este comisa cand mai intai transferand bani spre un account care apoi ii scoate.

Ce determina ca isFlaggedFraud sa fie setat sau nu?

Se intelege ca originile la isFlaggedFraud nu este clar, in contrast cu descrierea propusa. Toate 16 setari a acestei variabile nu coreleaza cu explicatia variabilei. Data este descrisa ca fiind setata cand se face o incercare de a TRANSFERa un ‘amount’ mai mare de 200 000. In realitate, isFlaggedFraud poate sa ramana nesetat chiar daca se indeplineste conditia de mai sus. Poate oare setarea acestul flag sa fie dependenta de faptul ca oldBalanceOrig = newBalnceOrig, cand newBalanceOrig face update doar daca tranzactia trece, dar isFlaggedFraud va fi setat inainte sa treaca tranzactia.

Poate oare isFlagegdFraud sa fie setat bazandu-se pe faptul ca utilizatorul face tranzactii de mai multe ori? De notat ca duplicatele numelor utilizatorilor nu exista acolo unde isFlaggedFraud este setat, dar exista in tranzactiile unde variabila nu este setata. Deducem ca cei care au isFlaggedFraud setat au facut tranzactie o singura data. Foarte putini destinatari a tranzactiilor care au isFlaggedFraud au mai mult de o tranzactie.

De asemenea, se observa ca isFlaggedFraud setat apare la toate de step, prin urmare nu se accentueaza vreo corelatie dintre step sau oricare alta valoare din dataset care ar influenta aceasta variabila.

Concluzie: Chiar daca isFraud este mereu setat cand isFlaggedFraud este setat, dar deoarece isFlaggedFraud este setat doar 16 ori intr-un mod aleator, putem trata aceasta variabila ca una insignificanta si sa o stergem din dataset fara a pierde informatie.

Se stie ca CASH\_IN implica plata de comerciant (merchant) a carui nume are prefixul M. Dar, datasetul nu are comercianti care fac CASH\_IN tranzactii catre utilizatori.

In realitate nu sunt comercianti intre conturi de initiatori. Comerciantii sunt doar in account destinatari pentru toate PAYMENTS.

Concluzie: Intre accounturi din nameOrig si nameDest, pentru toate tranzactiile comerciantii cu prefixul M apare intr-un mod neasteptat.

Exista conturi comune pentru TRANSFER si CASH\_OUT frauduloase?

Din descrierea datei, comiterea fraudei implica mai intai efectuarea transferului la un cont (fraudulent) care apoi face CASH\_OUT. CASH\_OUT implică tranzacții cu un comerciant care plătește cash. Prin proces din 2 etape, contul fraudulent va fi ambele, destinatarul in TRANSFER si initiatorul in CASH\_OUT. Oricum, data arata ca nu sunt asa conturi comune prin tranzactiile frauduloase. Prin urmare, modus-operandi a datei nu este exacta.

Concluzie: Nici nameOrig, nici nameDest nu codifica contul comerciantilor in modul corespunzator, vom exclude aceste features pentru ca sunt inutile.

=====================================================================================

Data cleaning

Am dedus ca frauda apare doar in TRANSFER si CASH\_OUT. Deci noi punem doar data corespunzatoare in X.

Data are multe tranzactii cu balanta 0 la contul de destinatie inainte si dupa ce un amount diferit de 0 a fost trimis. Rata acestor tranzactii, unde 0 posibil denota lipsa valorii, este mai mare la cele frauduloase (50%) comparate cu cele adevarate (0.06%).

Deoarece balanta contului destinatar fiind 0 este un indicator puternic a fraudei, nu imputam soldul (balanta) contului (inainte de efectuarea tranzactiei) cu o statistica sau dintr-o distributie cu o ajustare ulterioara a sumei tranzactionate. Procedand astfel, ar masca acest indicator de frauda si ar face ca trazactiile frauduloase sa para reale. In schimb, inlocuim valoarea 0 cu -1, ceea ce va fi mai util pentru un algoritm adecvat de invatare automata care detecteaza frauda.

Data de asemenea are multe tranzactii cu soldul 0 in contul de origine, inainte si dupa ce o valoare nonzero a fost tranzactionata. In acest caz rata acestor tranzactii este mai mica in cele fraudulente (0.3%) comparativ cu cele reale (47%). Din aceleasi motive ca in cazul descris mai sus, in loc sa imputam o valoare numerica noi inlocuim valoare de 0 cu valoarea null.

Feature-engineering:

Deoarece soldurile-zero ajuta la diferentierea intre tranzactii fraudulente si reale, noi mai cream 2 coloane care vor inregistra errorile in conturile originare (initiante) si destinatare pentru fiecare tranzactie. Se va adeveri a fi foarte efeciente aceste coloane pentru algoritmul ML.

Vizualizarea Datei:

Cea mai buna metoda de a confirma ca dat contine suficienta info pentru ca algoritmul ML sa faca predictii corecte, este de a vizualiza direct diferentele dintre tranzactiile fraudulente si reale.

Dispersia in timp

Din grafic se poate observa ca tranzactiile fraudulente is distribuite mai omogen comparat cu cele reale. CASH\_OUT is mai multe decat TRANSFER in tranzactii reale, in contrast cu distributia balansata intre ele in tranzactiile fraudulente.

Dispersia prin cantitate

Prezenta fraudei in tranzactii poate fi distinsa si din graficul pentru coloana amount, dar noua coloana errorBalanceDest este mai efectiva la efectuarea distinctiei.

Separarea tranzactiilor reale de frauduloase

Graficul 3D distinge cel mai bine intre data frauda si non-frauda folosind coloana bazata pe eroare. Este clar ca coloana step este inefectiva in separarea fraudelor, din cauza naturii dungelor din graficul cu tranzactiile reale vs timp.

Amprentele tranzactiilor reale si frauduloase

Dovezi complete încorporate în setul de date ale diferenței dintre tranzacțiile frauduloase și autentice sunt obținute prin examinarea corelațiilor lor respective în hărțile termice de mai jos.

ML pentru detectarea fraudei in data deformata

Acum stiind ca data contine coloane care fac ca tranzactiile fraudulente sa fie detectabile, urmatorul obstacul pentru modelul ML este disbalanta inalta a datei.

Selectarea metricii: Deoarece data este destul de deformata, se va folosi AUPRC (aria under the precision-recall curve) in favoarea AUROC (area unde the receiver operating characteristic). Asta e din cauza ca AUPRC este mai sensibil la diferentele dintre algoritmi si parametrii setarii, decat AUROC.

Selectarea algoritmului ML: Prima abordarea a datei dezechilibrate este de a echilibra prin excluderea clasei majoritare inainte de aplicare a algoritmului ML. Dezavantajul undersampling-ului este ca modelul antrenat astfel nu va arata o performanta buna intr-o data deformata din lumea reala, deoarece aproape toata data a fost stearsa. O abordare mai buna poate fi oversampling-ul clasei minoritare, prin synthetic minority oversampling technique (SMOTE) continuta in biblioteca imblearn. Cel mai bun rezultat se obtine de algoritmul ML bazat pe copacii de decizii. Asa algoritmi nu doar se descurca cu lipsele valorilor din data, dar ajuta la accelerare rpin procesarea in paralel. Printre acesti algoritmi, extreme gradient-boosted (XGBoost) intrece random-forest.

Cele mai importante coloane in modelul ML

errorBalanceOrig este cea mai relevanta coloana pentru model. Coloanele sunt ordonate bazandu-se pe numarul de exemple afectate de split-urile pe acele coloane.

Vizualizarea modelului ML

Cum si s-a asteptat, din copacul de decizii vizualizat, coloana errorBalanceOrig este root node, deoarece este destul de semnificativ pentru model.

Bias-variance tradeoff

Modelul antrenat are un grad de bias si este putin underfit. Aceasta este indicat de nivelare in AUPRC, cat marimea setului de antrenare este marit in curba cross-validation. Cea mai buna metoda de imbunatatire a performantei modelului este de a mari parametrul max\_depth a XGBClassifier, prin costul timpului necesar pentru invatarea modelului.

//////Alti parametri a clasificatorului care pot fi ajustate pentru a corecta efectul de underfitting include micsorarea min\_child\_weight si micsorarea reg\_lambda.

Nu lucreaza

Concluzie:

Am analizat data pentru a vedea care coloane pot fi sterse. Graficile arata dovezi care confirma ca aceste coloane pot fi sterse sau pot fi adaugate noi. Pentru a opera cu data deformata, alegem o metrica corespunzatoare si algoritmul ML bazat pe copacuri de decizie care lucreaza cel mai bine cu clasele inalt dezechilibrate. Astfel, se poate aplica acest algorimul in problemele din viata reala.