Credit Card Fraud Detection

*

1st Marta Hajdina
Primijenjena Matematika
Prirodoslovno-matematički fakultet
Zagreb,Hrvatska

2nd Hrvoje Čosković
Računarstvo i Matematika
Prirodoslovno-matematički fakultet
Zagreb,Hrvatska

3rd Glorija Maloča
Matematika i informatika
Prirodoslovno-matematicki fakultet
Zagreb,Hrvatska

4th Matej Duvanjk
Računarstvo i Matematika
Prirodoslovno-matematicki fakultet
Zagreb, Hrvatska

***Abstract*—U ovom projektu rješavamo problem Detekcije prevara kod korištenja kreditnih kartica različitim metodama strojnog učenja. Glavna poteškoća koja se javlja u tom problemu je jaka nebalansiranost seta podataka, što je česta pojava u problemima strojnog učenja.**

***Index Terms*—SVM, Logistička regresija, RandomForest, NaiveBayes**

I. Uvod

Upotreba kreditnih kartica jako je raširena u današnjem svijetu,te je moderni život bez njih gotovo nezamisliv. S većom raširenošću dolazi i prevare su češće, te smo se htjeli pozabaviti ovim problemom koji je jednako bitan i obične ljude kao i financijskim instituciama. Ovaj problem je zapravo problem klasifikacije, pa ćemo u radu koristi najpopularnije klasifikacijske metode,kao što su Naivni Bayes, Logistička regresija, Support Vector Machine i Random Forest.Njihovu uspješnost ćemo evaluirati pomoću mjera Recall, Precision i F1, te Konfuzijske matrice.

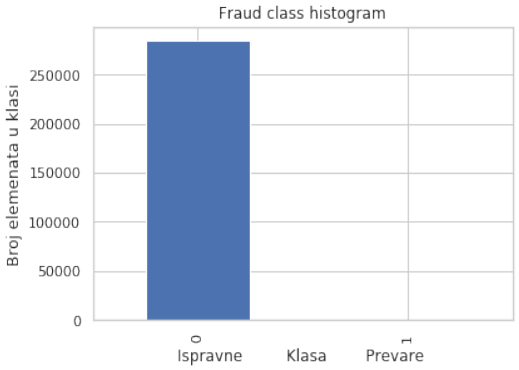
II. Opis skupa podataka

Temeljni problem sa kojim se nosimo u ovom projektu je nebalansiranim skup podatka. Iz prikaza na Slici 1. vidimo da klasi prevara pripadaju samo 492 instance, dok klasi ispravnih transakcija 284315 instance.

Skup podataka se sastoji od 30 značajki, od koji vrijeme transakcije i iznos nisu transformirani PCA algoritmom, dok je značenje ostalih 28 vektora značajki potpuno nepoznato, jer je na njima primijenjen PCA. Kako na značajkama vremena i iznosa transakcije nije proveden PCA, mi smo ih normalizirali.

U projektu ćemo se sa nebalansiranim skupom podataka nositi na više načina možemo ih podijeliti u 2 grupe, a to su Resample način i Podešavanje težina klasa. Podešavanje težina klasa utječe na algoritam koji treniramo te više penaliziramo kod Cost funkcije (koju nam je cilj minimizirati),krivu klasifikaciju instance iz klase sa manje primjera,nego krivu klasifikaciju instance iz klase sa više primjera.

Fig. 1. Instance u različitim klasama

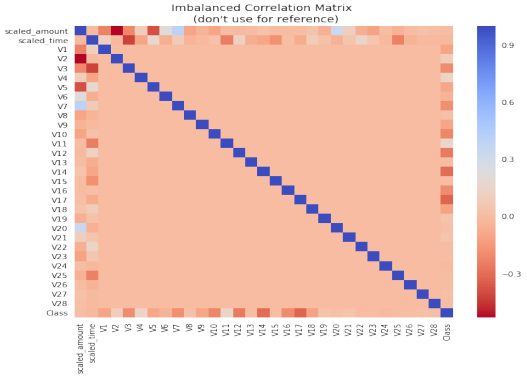


Resample metode zapravo mijenjaju skup podatka na kojima treniramo naše metode.

A. Korelacije između svojstava podataka

Korelacija među značajkama V1-V28 je 0 zbog provedenog PCA, prikaz na slici 2. Također uočavamo da PCA nije proveden na značajkama vrijeme i iznosa transakcije, pa između njih postoji korelacija.

Fig. 2. Korelacijska matrica



III. Opis korištenih metoda

Kao što je ranije spomenuto za problem klasifikacije koristili smo metode Logističke regresije, Naivnog Bayesa, Random Foresta i Support Vector Machina. Dok smo za rješavanje problema nebalansiranosti skupa podataka koristili različite resampling metode i podešavanje težina klasa.
Važno je napomenuti da je najprije izvorni set podataka podijeljen, tako da se zadrži nebalansiranost podataka i podskupovima, na set za treniranje i set za testiranje u omjeru 70:30. Nakon toga je na setu za testiranje provedena pojedina metoda za balansiranje skupa, te su zatim na balansiranom skupu trenirane različite klasifikacijske metode.

A. Logistička regresija

Logistička regresija je klasifikacijska metoda, koja se koristi za mnogo različitih problema kao što su klasifikacija email spamova, transkacijske prevare, klasifikacija malignih tumora, itd...
Logistička regresija mijenja svoj output koristeći logističku sigmoid funkciju, te vraća vjerojatnost pripadanja klasi.

$$h_\theta(x) = \frac{1}{1 - e^{\theta' x}}$$

Funkcija koja mjeri grešku kod logističke regresije je:

$$J(\theta) = -\frac{1}{m}\sum_{i=1}^{n}(y^{(i)} \cdot log(h_\theta(x_i)) + (1 - y^{(i)}) \cdot log(1 - h_\theta(x_i)))$$

Zbog regularizacije, koja sprječava overfitting, dodajemo regularizacijski faktor,a zbog nebalansiranosti podataka dodajemo različite težine klasama.

$$J(\theta) = -\tfrac{1}{m}\sum_{i=1}^{n}(w_0 y^{(i)} \cdot log(h_\theta(x_i)) + w_1(1 - y^{(i)}) \cdot log(1 - h_\theta(x_i))) + C\sum_{j=1}^{m}\theta_j^2$$

U našem projektu smo w_0, w_1 i C koristili kao hiperparametare i mijenjali njihove vrijednosti tako da vrijednost recall mjere bude što veća, ali da se ujedno i mjera presition previše ne smanji.
Prije nego što krenemo koristiti metodu logističke regresije, moramo razdvojiti naš skup podatka na skup podataka za treniranje i skup podataka za testiranje i to smo učinili u omjeru 0.7:0.3.

- Prvi primjer koji smo koristili za logističku regresiju je da smo pomoću funkcije GridSearchCV optimizirali hiperparametre $w0$,$w1$ i C iz gornje jednadžbe.Gridsearch radi tako da podjeli naš skup za treniranje na koliko god dijelova želimo (mi smo ga podjelili samo na 2 jer imamo relativno puno podataka u skupu), te trenira metoda sa nekim od parametera koje smo prethodno specificirali te na preostalom djelu skupa za treniranje računa vrijednost koju smo sami tražili da bude najveća te na kraju dostavi paramtere koji najviše povečavaju traženu vrijednost npr. recall,precision, f1,rocaucscore... Unutar GridSearcha birali smo između sljedećih paramtera za C i w_0,w_1. Vrijednost parametra C smo birali između vrijednosti $[0.05, 0.1, 0.15, 0.2, 0.25, 0.3, 0.4]$, dok smo parametar težina klasa birali između sljedećih parametara $[(0:.1, 1:.9), (0:.01, 1:.99), 'balanced', (0:.3, 1:.7), (0:.2, 1:.99)]$ Provedbom Gridsearcha dobivamo da je Vrijednsost parametara $C = 0.25$, dok je omjer težina klasa $w_0 = 0.1$ i $w_1 = 0.9$.Nakon što smo dobili najbolje tražene parametre za algoritam koristeći GridSearchCV,koji nam daju najbolju vrijednost $f1$,ponovno istreniramo Logističku regresiju sa tim paramterima na cijelom skupu za treniranje i zatim mjerimo recall, precision, F1 te konfuzijsku matricu.

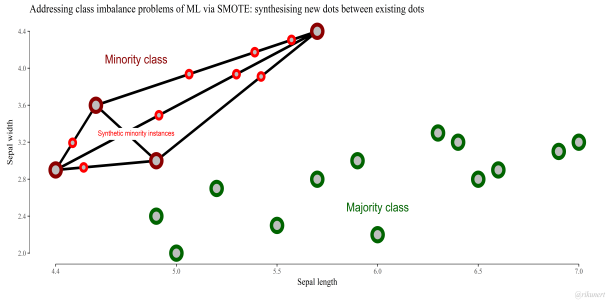
Fig. 3. Konfuzijska matrica za Logističku regresiju uz optimiziranje težine klasa

Dok su vrijednosti Recalla je 0.826,Precision je 0.805 te F1=0.815. Bitnije nam je imati što viši recall zbog toga što recall prema definiciji je sljedeče $recall = \frac{TP}{pravipozitivni} = \frac{TP}{TP+FN}$
što zapravo mjera koliko je od sveukupno pozitivnih naš algoritam njih točno klasificirao te nam je to važnost da bi znali otprilike točnost našega algoritma.

Veličina F1 nam je također bitna jer ne želimo da bude niti premali Precision iz razloga da ne bi previše transakcija bilo klasificirane kao prevare, što bi dovelo do bez potrebnog optuživanja kljenata.

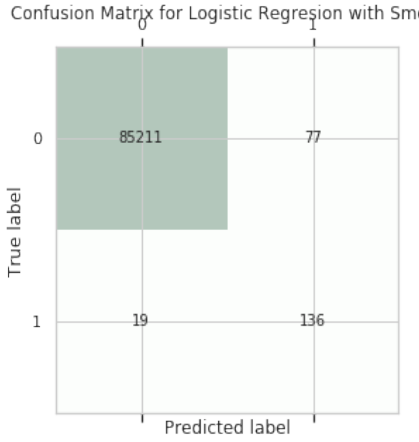
- Sljedeći primjer u kojem smo koristili logističku regresiju je bio kada smo koristili uz korištenje algoritma za sintetičko genriranje slučajeva iz manje klase uz pomoć algoritma SMOTE.Na nekakvoj primarnoj razini SMOTE radi tako da vuče linije između točaka klase s manje elemenata.Zatim negdje na dužini između 2 točke dodaje nove elemente. Kao što je prikazano na slici ispod.

Fig. 4. Kako SMOTE generira nove instance



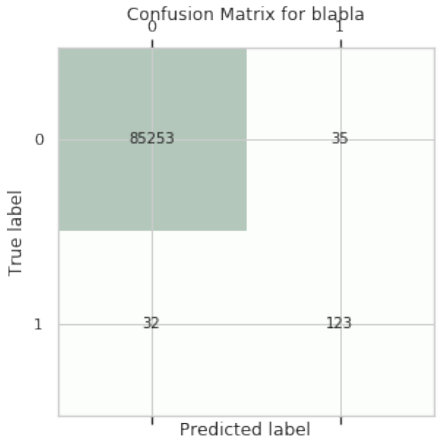
Smote koji smo koristili implementiran je u API-u za Python imbalanced.learn i ima različite parametre,jedan od njih mi smo uzeli kao hiperparmetar koji smo optimizirali unutar Grid Searcha,a to je odnos između klasa i koliko će algoritam generirati novih elemenata. Naime paramater sampelingstrategy ukoliko primi vrijednost tipa float, tumači ju kao omjer $\alpha_{os} = \frac{N_M}{N_{rm}}$ gdje je N_M broj instanci u većinskoj klasi,dok je N_{rm} broj instanci u manjinskoj klasi nakon obavljenje resampling metode, također uz te paramtere opet smo optimizirali i paramterar C u cost-funkciji logističke regresije.Vrijednosti paramtara sampelingstrategy između kojih smo birali je sljedeča :$[0.075, 0.085, 0.01, 0.03, 0.1, 0.3, 0.5, 0.8]$,dok za parametar C logističke regresije imamo vrijednosti $[0.01, 0.04, 0.06, 0.1, 0.3, 0.5, 0.9]$ Također SMOTE alogritam smo provodili unutar GridSearcha na podacima na kojima bi on trenirao algoritam sa određenim paramterima,ali i ne na onima na kojima bi testirao da bi dobili što bolju procjenu za veličine koje tražime jer kada bi prije to učinili te nakon toga dijelili na djelove gdje treniramo i testiramo algoritam, došlo bi do mogučnosti lažnog optimizma. Rezulatiti koje smo dobili su sljedeči: Vrijednosti Recall-a:0.877,Precision:0.638,F1:0.739.
- Zadnji primjer kod kojeg smo koristili Logističku regresiju je ja da smo za balansiranje klasa koristili RandomUnderSampling koji nam omogućava da većinsku klasu smanjimo tako da slučajnim odabiremo bez ponavljanja članove klase koja sadržava valjane transakcije,te smanjimo njihov broj.Također koristimo

Fig. 5. Konfuzijska matrica Logističke regresije sa SMOTE algoritmom



funkciju RandomUnderSampling iz imbalanced.learn koja sadržava parametar samplingstrategy koji ako primi vrijednost tipa float tumači ju kao omjer $\alpha = N_{rM}/N_m$ gdje je N_{rM} broj instanci u klasi valjanih transkacija nakon provedene RUS-meteode,N_m broj instanci u klasi prevara. Također opet optimiziramo parametar C i parametre samplingstrategy. Isprobavamo za parametar C različitih vrijednosti $[0.0005, 0.001, 0.05, 0.1, 0.3, 0.5, 1, 3, 5]$, i za parametar samplingstrategy $[0.005, 0.01, 0.015, 0.09, 0.1, 0.2]$,te GridSearchCV pronalazi najbolje vrijednosti za koja povečava F1 vrijednost.

Fig. 6. Konfuzijaska matrica Logističke regresije nad RUS podacima



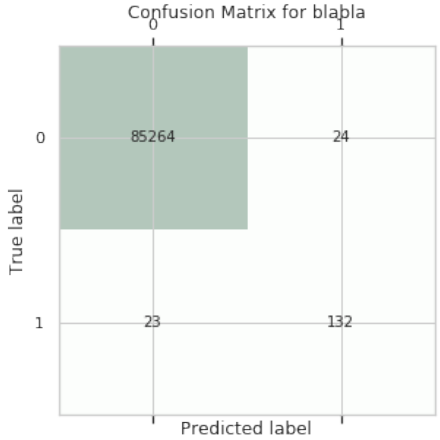
B. *Support Vector Machine*

Cilj algoritma Support Vector Machine je naći ravninu koja razvaja primjere iz različitih klasa sa maksimalnom marginom.Pošto naš problem najvjerojatnije nije linearno separabilan moramo koristiti varijantu SVM-a sa soft marginom,te koristiti olabavljeni uvjet za optimalnu hiper-ravninu tako da uključimo dodatan izraz
$y_i(x_i \cdot w) \geq 1 - \varepsilon_i$ te treba minimizirati $w'w + C\sum_{i=1}^{m} \varepsilon_i$

gdje je C regularizacijski parametar koji kontrolira razmjenu između maksimiziranja margine i minimizacije training greške.Za malene vrijednosti C utječu na naglašavnje margine te može dovesti do podnaučenosti, visoki C utječe tako da bolje uči konkretne podatke te može dovesti do prenaučenosti modela,njega smatramo hiper-parametrom SVM-a i učimo ga preko GridSearch-a nalazimo koji će najbolje koristiti našem skupu podataka.

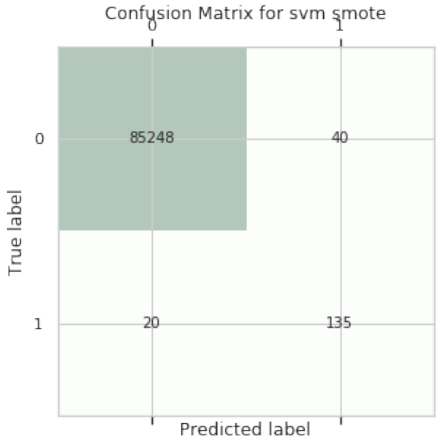
- Prvi primjer koji smo koristili za SVM je da smo pomoću funkcije GridSearchCV optimizirali hiperparametre $w0$,$w1$ i C gdje su w_0,w_1 težine klasa zbog ne balansiranog dataseta,dok je C gore objašnjeni hiperparametar SVM-a. Vrijednost parametra $C = 0.01$ smo birali između vrijednosti $[0.05, 0.1, 0.15, 0.2, 0.25, 0.3, 0.4]$, dok smo parametar težina klasa birali između sljedećih parametara $[(0 : .1, 1 : .9), (0 : .01, 1 : .99), 'balanced', (0 : .3, 1 : .7), (0 : .2, 1 : .8)]$ Provedbom Gridsearcha dobivamo da je Vrijednsost parametara $C = 0.25$, dok je omjer težina klasa $w_0 = 0.1$ i $w_1 = 0.9$.Nakon što smo dobili najbolje tražene parametre za algoritam koristeći GridSearchCV,koji nam daju najbolju vrijednost $f1$,ponovno istreniramo SVM sa tim paramterima na cijelom skupu za treniranje i zatim mjerimo recall, precision, F1 te konfuzijsku matricu.

Fig. 7. Konfuzijska matrica SVM sa traženjem težinskih klasa



- Drugi primjer u kojem smo koristili SVM je kada smo ga koristili uz Smote , te smo opet samplingstrategy uzeli kao hiperparmetar koji smo optimizirali unutar Grid Searcha,a to je odnos između klasa i koliko će algoritam generirati novih elemenata. Naime paramater sampelingstrategy ukoliko primi vrijednost tipa float, tumači ju kao omjer $\alpha_{os} = \frac{N_M}{N_{rm}}$ gdje je N_M broj instanci u većinskoj klasi,dok je N_{rm} broj instanci u manjinskoj klasi nakon obavljenje resampling metode, također uz te paramtere opet smo optimizirali i paramterar C u SVM-a.Vrijednosti parametara sampelingstrategy između kojih smo birali je sljedeča :$[0.075, 0.085, 0.01, 0.03, 0.1, 0.3, 0.5, 0.8]$,dok za parametar C logističke regresije imamo vrijednosti $[0.01, 0.04, 0.06, 0.1, 0.3, 0.5, 0.9]$ Također SMOTE alogritam smo provodili unutar GridSearcha na podacima na kojima bi on trenirao algoritam sa određenim parametrima,ali i ne na onima na kojima bi testirao da bi dobili što bolju procjenu za veličine koje tražimo jer kada bi prije to učinili te nakon toga dijelili na dijelove gdje treniramo i testiramo algoritam, došlo bi do mogućnosti lažnog optimizma. Rezulatiti koje smo dobili su sljedeći: Vrijednosti

Fig. 8. Konfuzijska matrica SVM sa SMOTE algoritmom

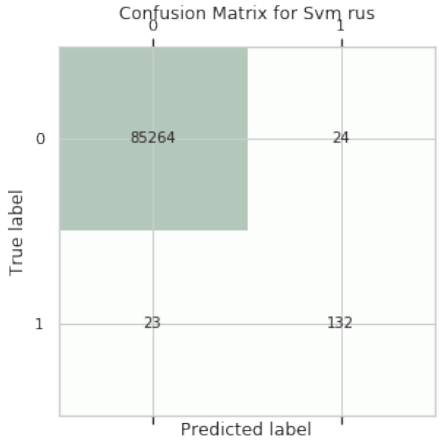


Recall-a:0.871,Precision:0.771,F1:0.818.

- Sada prije treniranja SVM-a zovemo RUS(RandomUnderSampling) za balansiranje klasa koji nam omogućava da većinsku klasu smanjimo tako da slučajnim odabiremo bez ponavljanja članove klase koja sadržava valjane transakcije,te smanjimo njihov broj.Ponovno koristimo funkciju RandomUnderSampling iz imbalanced.learn koja sadržava parametar samplingstrategy koji ako primi vrijednost tipa float tumači ju kao omjer $\alpha = N_{rM}/N_m$ gdje je N_{rM} broj instanci u klasi valjanih transkacija nakon provedene RUS-meteode,N_m broj instanci u klasi prevara. Također opet optimiziramo parametar C i parametre samplingstrategy. Isprobavamo za parametar C različitih vrijednosti $[0.0005, 0.001, 0.05, 0.1, 0.3, 0.5, 1, 3, 5]$, i za parametar

samplingstrategy $[0.005, 0.01, 0.015, 0.09, 0.1, 0.2]$,te GridSearchCV pronalazi najbolje vrijednosti za koja povečava F1 vrijednost.

Fig. 9. Konfuzijaska matrica SVM nad RUS podacima



C. Random Forest

Random Forest je metoda nadziranog učenja koja se koristi pri rješavanju klasifikacijskih, regresijskih i općenitih problema kod kojih se koriste stabla odlučivanja (eng. *decision trees*). Metoda generira ansambl stabla odlučivanja i uprosječuje njihova predviđanja (Fig. 10).

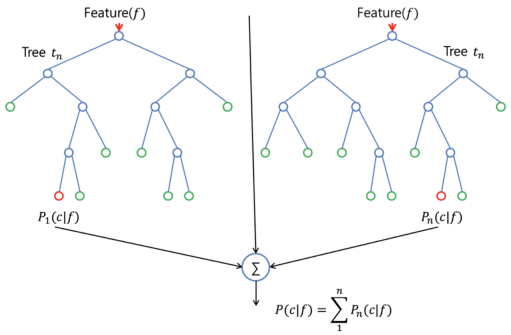


Fig. 10. Stabla odlučivanja u ansamblu

Svako stablo je izgrađeno na slučajnom podskupu (s ponavljanjem) skupa za učenje i prilikom izgradnje svakog stabla se uvijek uzima slučajni podskup značajki za izgradnju svakog pojedinog čvora. Opisat ćemo način na koji smo probali riješiti Credit card fraud detection problem pomoću Random Forest klasifikatora iz sklearn.ensemble modula.

1) Skaliranje podataka: Skaliranje podataka je proces koji pomaže algoritmu lakše uočiti i razumjeti zakonitosti u podacima. Analizom podataka ustanovili smo da vrijeme i iznos nemaju normalne distribucija. Ako malo bolje pogledamo Fig. 11, možemo zaključiti kako je ipak distribucija vremena uobičajena, jer prati uzorak dan-noć (više transakcija po danu, nego po noći), stoga vrijeme nismo skalirali.

S druge strane, iznos skaliramo pomoću RobustScalar iz sklearn.preprocessing modula. Iznos nakon transmormacije je prikazan na Fig. 12.

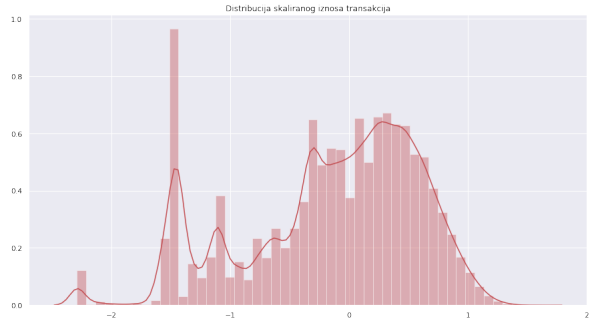


Fig. 12. Skalirani iznos

2) Evaluacija modela: Ponašanje i treniranje Random forest klasifikatora pratit ćemo pomoću matrice konfuzije (eng. *confusion matrix*) i mjera uspješnosti poput precision-a, recall-a i f_score-a. Nastavaljamo algoritmom s treniranjem i pozivanjem Random Forest klasifikatora kojem nisu podešeni hyperparametri i koji je radio na nebalansiranom datasetu iz kojeg nisu uklonjeni outlieri. Rezultati su slijedeći:

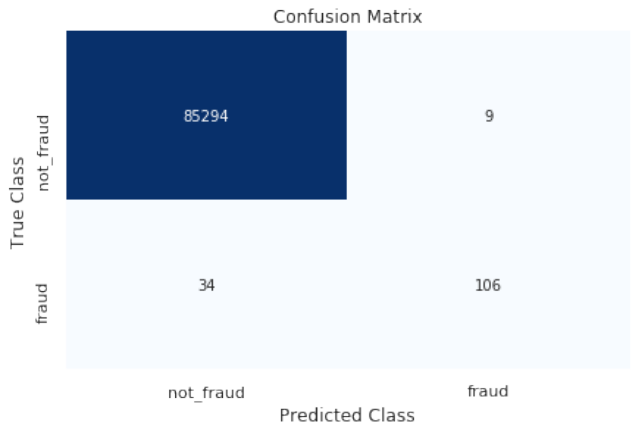


Fig. 13. $Prec = 0.921739$, $Recall = 0.757143$, $f\_score = 0.831373$

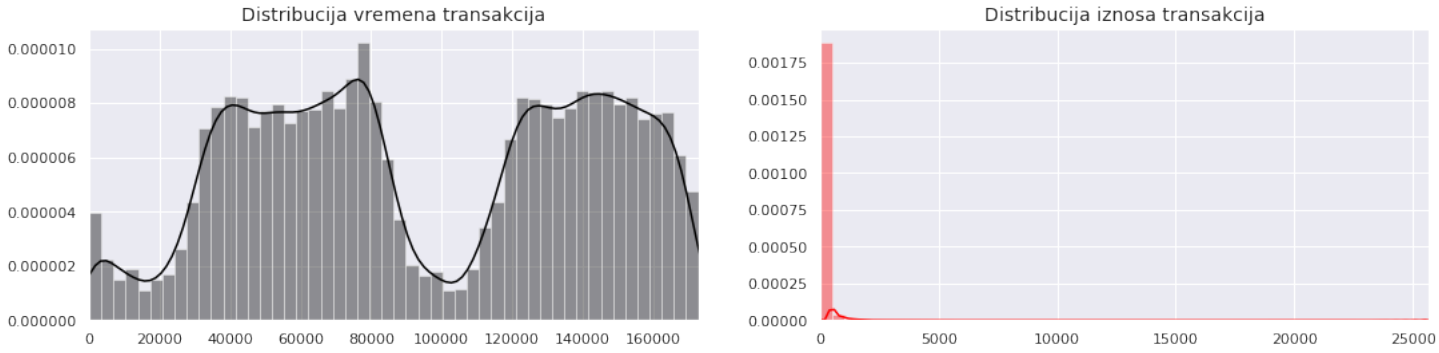


Fig. 11. Distribucija vremena i iznosa u skupu podataka

3) Sampling metode korištene nad datasetom: Zbog već navedenog problema nebalansiranosti skupa podataka, prisiljeni smo provesti oversampling i undersampling metode na datasetu s kojim radimo. Koristili smo više metoda i pratili njihov učinak, kako bi što bolje mogli odabrati novi dataset dobiven tim metodama.

Navedene metode su korištene prilikom samplinga dataseta:

SMOTE – Već opisana.

RandomUnderSampler - RUS – Smanjuje veću klasu izbacujući iz nje random podatke.

RandomOverSampler - RUS – Povećava manju klasu tako da random generira podatke.

Tomek Links – Izbacuje podatke između kojih postoji "TomekLink" https://imbalanced-learn.readthedocs.io/en/stable/under_sampling.html#tomek-links

SMOTEENN – Over sampling metoda. Kombinira under- i over- sampling pomoću SMOTE metode i Edited Nearest Neighbours.

SMOTETomek – Over sampling metoda. Kombinira under- i over- sampling pomoću SMOTE metode i TomekLinks metode.

Rezultati treninga modela na dobivenim podacima su slijedeći:

	precision	**recall**	**f_score**
SMOTE	0.848485	0.800000	0.823529
RUS	0.037679	0.885714	0.072282
ROS	0.923077	0.771429	0.840467
TomekLinks	0.921739	0.757143	0.831373
SMOTEENN	0.811594	0.800000	0.805755
SMOTETomek	0.844444	0.814286	0.829091

Zbog ovih rezultata, izabiremo podatke generirane SMOTE-Tomek metodom, zbog toga što ima najveći recall (broj određenih prevara) od SMOTE metode (RUS metodu ovdje ne razmatramo jer je ima nizak precision (broj određenih ispravnih transakcija), jednako tako dobar precision. Sva naša daljnja razmatranja temeljit će se na tom skupu podataka.

4) Optimizacija parametara klasifikatora : GridSearchCV metodom smo proveli traženje optimalnih hyperparametara za skup podataka kojim raspolažemo. dobili smo slijedeće rezultate za 346 minuta:

```
from sklearn.model_selection
    import RandomizedSearchCV

param_grid = {
    'bootstrap': [True],
    'max_depth': [80, 90, 100, 110],
    'max_features': [2, 3],
    'min_samples_leaf': [3, 4, 5],
    'min_samples_split': [8, 10, 12],
    'n_estimators': [100, 200, 300, 500]}

from sklearn.model_selection
    import GridSearchCV
grid_search = GridSearchCV(rf, param_grid,
    cv = 3, n_jobs = 8, verbose = 2)

best = grid_search.fit(X_train, y_train)

rf_opt = RandomForestClassifier(
bootstrap = True,
            max_depth = 90,
            max_features = 3,
            min_samples_leaf = 3,
            min_samples_split = 8,
            n_estimators = 100,
            n_jobs = -1 )
```

5) Matrica korelacije: Matrica korelacije je osnova razumijevanja odnosa naših podataka, ali matrica korelacije mora biti generirana na balansiranom skupu podataka, jer kao što uočavamo sa Fig. 14, na nebalansiranom skupu ne vidimo zavisnosti podataka, pa tako ne možemo zaključiti koji podaci najviše utječu na vrijednost 'Class'.

pozitivna korelacija = "ako se prva varijabla poveća, ona s kojom je u pozitivnoj korelaciji će se isto povećat"

negativna korelacija = "ako se prva varijabla smanji ona s kojom je u negativnoj korelaciji će se povećat"

Iznos korelacije je iz domene [-1,1]:

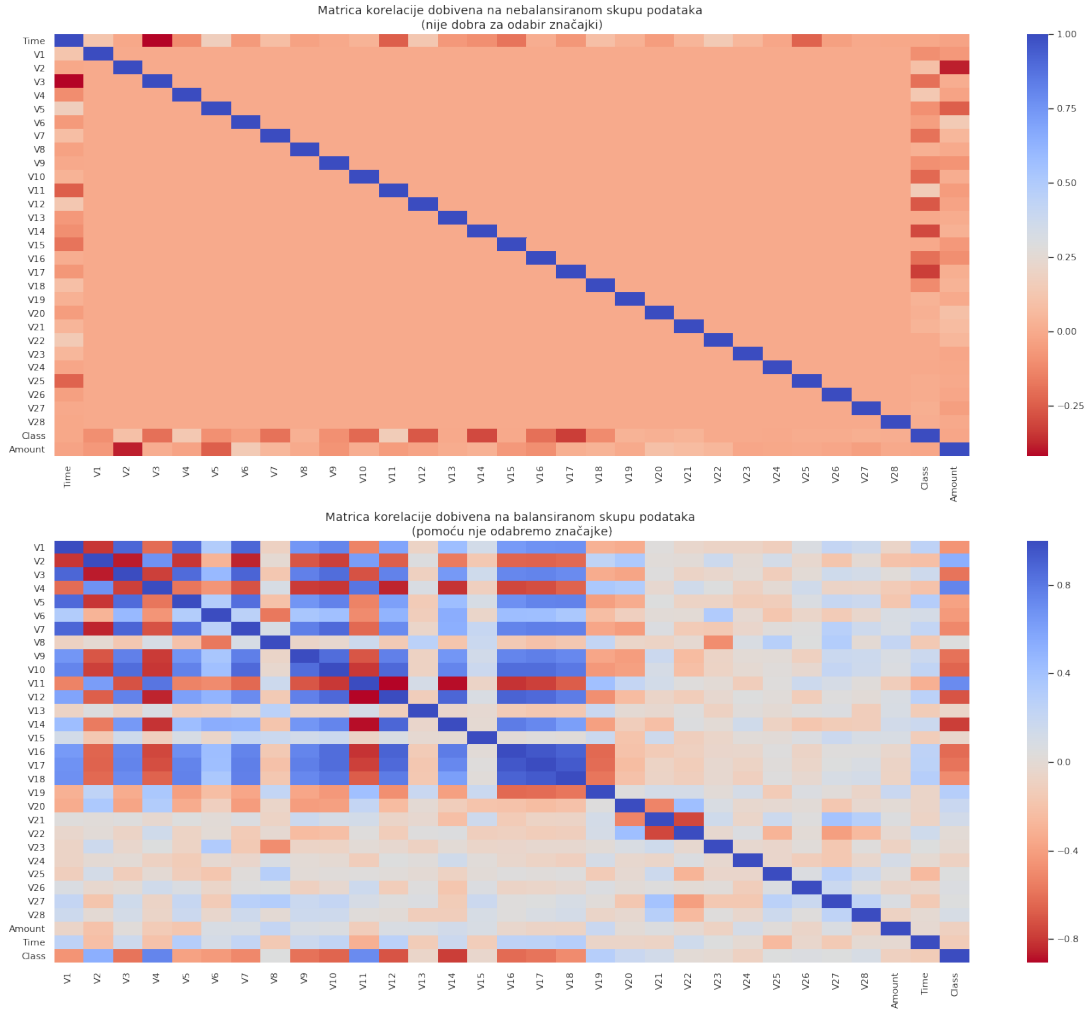


Fig. 14. Matrice korelacije na različitim skupovima podataka: gornja na orginalnom datasetu, donja na datasetu generiranom SMOTETomek metodom oversamplanja.

1.00 = savršeni pozitivni utjecaj
0 = nema utjecaja
-1.00 = savršeni negativni utjecaj

Iz dobivenih rezulata smo zaključili da su **V14, V12, V10, V16** u negativnoj korelaciji sa **Class** - to znači što su te vrijednosti manje (bliže -1) da će transakcija vrlo vjerojatno biti Fraud. Također se vidi da su **V11, V4, V2 i V19** u pozitivnoj korelaciji sa **Class** - to znači što su te vrijednosti veće (bliže 1) da će transakcija vrlo vjerojatno biti Fraud.

Sada kada znamo koje su značajke najutjecajnije na Određivanje Frauda transakcije, pokušat ćemo detektirat anomalije u njima i maknut ih iz skupa podataka.

6) Detektiranje i uklanjanje anomalija: Ovdje treba biti oprezan, može se dogoditi da uklonimo previše podataka i tako naštetimo našem modelu. Detektiranje i "čišćenje" podataka radimo pomoću Interquartile Range Method (IQR). To je metoda koja pomoću faktora **k** računa donju granicu (granicu ispod 25. centila) i gornju granicu(granicu iznad 75. centila) te miče podatke koji nisu unutar te dvije granice. Centil se računa tako da ako imamo 10000 podataka, onda će 50. centil biti srednja vrijednost 5000. i 5001. podatka.

7) Krajnji test i evaluacija modela: Na poslijetku pokrećemo training i test na pročišćenom skupu podataka, dobivajući zadnje rezultate.

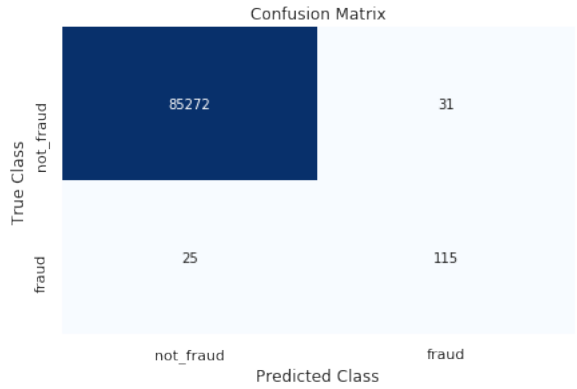


Fig. 15. $Prec = 0.787671, Recall = 0.821429, f\_score = 0.804196$

Vidimo da se Recall povećao za gotovo 7%, odnosno da je RF klasifikator nakon optimizacije parametara i treninga na pročišćenom skupu podataka prepoznao čak 9 slučajeva prevara više od početnog modela.

D. Gaussian Naive Bayes

Metoda naivnog Bayesovog klasifikatora se temelji na Bayesonom teoremu i "naivnoj" pretpostavci nezavisnosti značajki. Iako pretpostavka o nezavisnosti značajki nije uvijek prisutna u praksi, pokazalo se kako Bayesov klasifikator vrlo dobro klasificira bez obzira na nju. Jedna od glavnih prednosti ove metode je malen broj parametara. Uz pretpostavku nezavisnosti, korištenjem Bayesovog teorema dobivamo:

$$P(C_j|x) = \frac{p(x|C_j)P(C_j)}{p(x)}$$

Optimalna klasifikacijska odluka je ona koja maksimizira aposteriornu vjerojatnost, a kako je $p(x)$ konstanta, hipotezu možemo zapisati kao:

$$h(x) = argmax \quad p(x|C_k)P(C_k)$$

Kao i ranije, prvotni skup podataka je podijeljen na train i test set, u omjeru 70:30. Train set smo balansirali sa 4 različite metode, te zatim na njima trenirali klasifikacijsku metodu, pokušavajući optimizirati parametre.

- Undersampling metodom balansiramo train set koristeći $RandomUnderSampler$. Zatim treniramo metodu koristeći $GaussianNB$ mijenjajući $var\_smoothing$ parametar. Nakon svakog treniranja, testiramo metodu za određeni parametar na train setu, te mjerimo recall, precision i f1. Slika prikazuje dobivene rezultate za 15 parametara iz intervala $[10^{-20}, ..., 10^{-1}]$

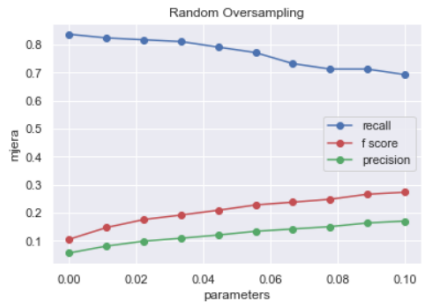


Fig. 16. GaussianNB s različitim parametrima trenran na undersampling balansiranom setu podataka

Nakon toga $GridSearch$ metodom uz pomoć $Pipeline$ optimiziramo paramtre $RandomUnderSampling$ metode, te na njoj parametre za $GaussianNB$. Imamo 5 parametara za $RandomUnderSampling$ iz $[0.02, ..., 0.08]$ i 10 parametara za $RandomUnderSampling$ iz intervala $[0.02, ..., 0.05]$. Mjera po kojoj se vrši optimizacija je recall. Za dobivene optimalne parametre testirali smo metodu na skupu za testiranje te dobili: recall=0.817, precison=0.0671 i F-score=0.124 . Kofuzijska matrica je:

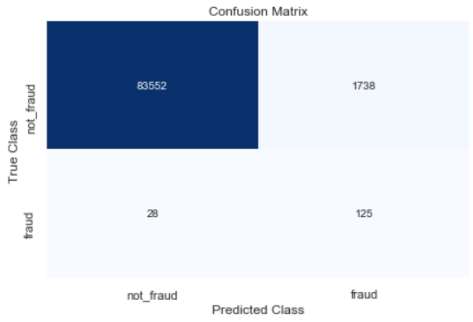


Fig. 17. Konfuzijska matrica za optimalne parametre undersampling metode i Gaussovog naivnog Bayesa

- Na potpuno isti način kao i kod undersampling metode, gledamo kako se Gaussov niavni Bayes ponaša na setu balansiranom oversampling metodom. Ovdje smo koristili $RandomOverSampler$. Prvo je Gaussov naivni Bayes testiran za parametre iz intervala $[10^{-17}, ..., 10^{-1}$, te su dobivene mjere za te parametre prikazane na slici:

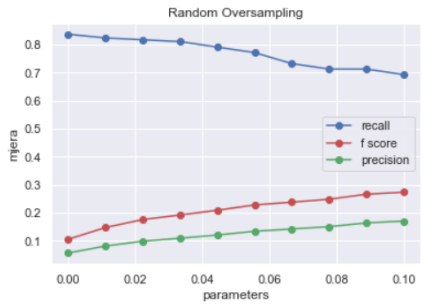


Fig. 18. GaussianNB s različitim parametrima trenran na oversampling balansiranom setu podataka

Nakon toga koristeći optimiziramo parametre za oversampling na intervalu $[0.07, ..., 0.2]$, te parametre za $GaussianNB$ na intervalu $[0.01, ..., 0.2]$. Za dobivene optimalne parametre mjere uspješnosti su: recall=0.824, precison=0.071 i F-score=0.129. Dok matrica kofuzije izgleda ovako:

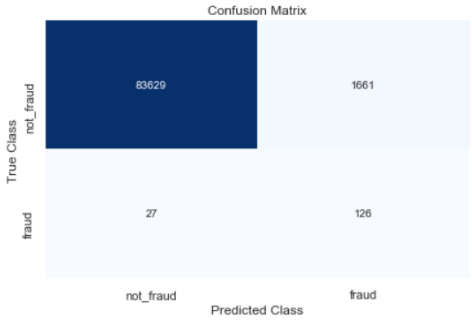


Fig. 19. Konfuzijska matrica za optimalne parametre oversampling metode i Gaussovog naivnog Bayesa

- Slijedeći isti način razmmišljanja kao i ranije, gledamo kako se $GaussianNB$ ponaša na train setu balansiranom $TomekLinks$ metodom. Ponašanje $GaussianNB$ za različite parametre prikazano je na slici:

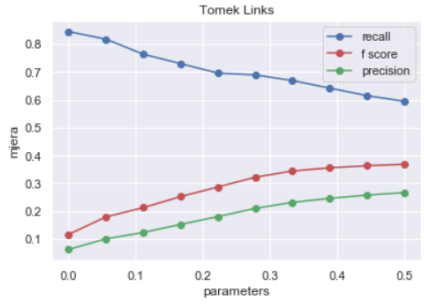


Fig. 20. GaussianNB s različitim parametrima trenran na oversampling balansiranom setu podataka

Dok smo $GridSearch$ metodom dobili da mjere uspješnosti za optimalni parametar iznose: recall=0.844, precison=0.072 i F-score=0.133. Konfuzijska matrica prikazana je na slici niže:

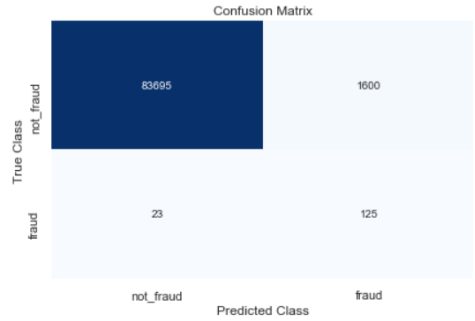


Fig. 21. Konfuzijska matrica za optimalne parametre oversampling metode i Gaussovog naivnog Bayesa

Iz rezultata možemo zaključiti kako Gaussov naivni Bayes nije najbolja opcija za klasifikaciju ovog problema. Kako je klasifikacijska metoda davala otprilike jednake rezultate prilikom treniranja na različito balansiranim setovima, zaključujemo da je jedini mogući problem u "naivnoj" pretpostavci, tj. da značajke nisu međusobno nezavisne.

REFERENCES

[1] Kaggle: https://www.kaggle.com/mlg-ulb/creditcardfraud
[2] Skripta Strojno učenje PMF
[3] Skripta Strojno učenje FER
[4] Aurélien Géron, “Hands-On Machine Learning with Scikit-Learn and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems”
[5] Nitesh V. Chawla, Kevin W. Bowyer, Lawrence O. Hall, and Y. Tagawa, “SMOTE: Synthetic Minority Over-sampling Technique” ,
[6] Sotiris Kotsiantis, Dimitris Kanellopoulos, Panayiotis Pintelas , “Handling imbalanced datasets: A review ” ,
IMBALANCED DATASETS: FROMSAMPLING TO CLASSIFIERST. Ryan Hoens and Nitesh V. Chawla