SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU

FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA OSIJEK

Sveučilišni studij

OTKRIVANJE PRIJEVARA S KREDITNIM KARTICAMA

Projektni zadatak

Tomislav Barbarić

Osijek, 2023.

Sadržaj

1. UVOD

Cilj projektnog zadatka je usporediti dostupne modele za klasifikaciju prijevara kreditnim karticama. Projekt je pisan jezikom Python dok je dataset preuzet sa Kaggle repozitorija. Prijevare kreditnim karticama sve su češća pojava razvojem tehnologija. Načina prevare ima mnogo, od krađe kreditnih kartica do prikupljanja podatak raznim phishing metodama. Bankama je prioritet predvidjeti takve transakcije i na vrijeme ih spriječiti. Pri tome koriste razne metode poput stvaranja profila korisnika, računanja ocjene prijevare i umjetnu inteligenciju za predviđanje prijevara. U ovome projektu usporedit će se četiri modela koja konceptualno prikazuju način detekcije prijevara.

2. OPIS DATASETA I PREDOBRADA PODATAKA

Za treniranje modela potrebno je imati određene podatke sa što većim brojem instanci. Dataset korišten u ovome radu sastoji se od 2840000 bankovnih transakcija i 31 značajke. Dostupan je na Kaggle repozitoriju.

Tablica 1. Prikaz podataka Dataseta

Time	V1	V2	…	V28	Amount	Class
0	-1.359807	-0.072781	…	-0.021053	149.62	0
1	-0.966272	-0.185226	…	0.061458	123.50	0
2	-1.158233	0.877737	…	0.215153	69.99	0

Tablica 1. prikazuje podatke i značajke koji se nalaze u datasetu. Značajka Time predstavlja vrijeme u sekundama koje je proteklo između transakcija. Pošto ima jako veliki raspon i ne utječe na krajnji rezultat briše se iz dataseta. Značajke V1 do V28 nemaju specifično ime zbog anonimnosti te predstavljaju parametre koje banka prikuplja pri svakoj transakciji. Amount predstavlja količinu novaca koja je korištena pri transakciji. Raspon mu je između 0 i 25000 te se zbog toga mora skalirati. Class zmačajka predstavlja klasu odnosno ako je vrijednost 0 transakcija je regularna, a ako je vrijednost 1 transakcija je prevara.

2.1. Skaliranje značajke Amount

Kako bi se podatci skalirali na vrijednosti koje približno odgovaraju značajkama V1 do V28 koristi se RobustScaler koji je dostupan u Sklearn biblioteci. Na slici 2.1 vidljiv je isječak koda koji skalira podatke Amount. Podatci su skalirani na vrijednosti između -0.307413 i 358.683155.

```
from sklearn.preprocessing import RobustScaler

rob_scaler = RobustScaler()
data['scaled_amount'] = rob_scaler.fit_transform(data['Amount'].values.reshape(-1,1))
```

Slika 2.1 Skaliranje podataka

Ova metoda koristi medijan i interkvartilni raspon kako bi odredila skaliranu vrijednost. Formula kojom računa novu vrijednost podatka je:

$$X_{scale} = \frac{X_i - X_{med}}{X_{75} - X_{25}}$$

Nakon uklanjanja Time značajke i skaliranja Amount značajke podatci izgledaju kao što je predstavljeno u tablici 2.

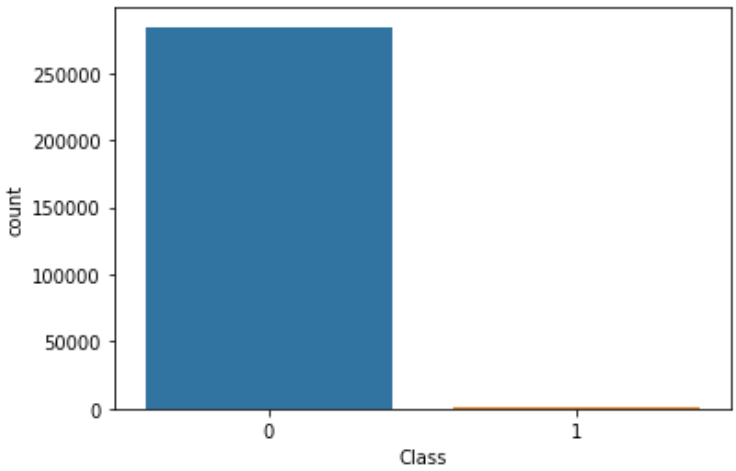
Tablica 2. Konačni podatci

V1	V2	…	V28	Class	scaled_amount
-1.359807	- 0.072781	…	-0.021053	0	1.783274
-1.358354	-1.340163	…	0.014724	0	-0.269825
-0.966272	-0.185226	…	-0.059752	0	4.983721

3. BALANSIRANJE PODATAKA

Prije treniranja modela potrebno je provjeriti balansiranost klasa. Ako se modeli istreniraju sa nebalansiranim skupom podataka model će biti pristran na klasu koja ima više instanci. Analizom dataseta utvrđeno je da je samo 0.2% podatak klase 1 odnosno transakcija prevare. Na slici 3.1 i 3.2. su grafički vidljivi problemi nebalansiranosti.

Slika 3.1 Postotni prikaz nebalansiranosti

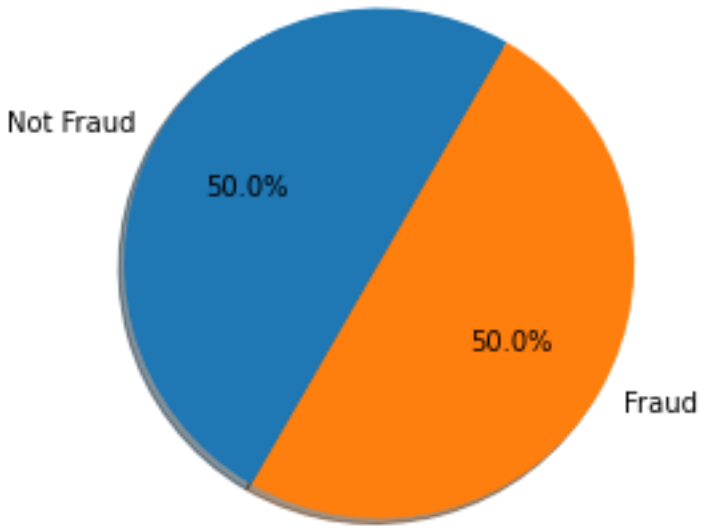


Slika 3.2 Prikaz nebalansiranosti stupčastim grafom

Postoje dva pristupa rješavanju problema balansiranosti, a to su undersampling i oversampling. Undersampling je metoda kojom se broj instanci dominante klase nastoji izjednačiti sa brojem instanci manjinske klase. Ovresampling radi suprotno tako što broj instanci manjinske klase nastoji izjednačiti sa brojem instanci dominante klase. Za ovaj dataset bolje je koristiti oversampling jer je postotak manjinske klase iznimno mal te se ovim načinom ne gube korisne informacije za treniranje modela. U ovome projektu korištena je SMOTE analiza za balansiranje podataka.

SMOTE analiza za balansiranje odabire jednu instancu manjinske klase nasumično. Zatim za K susjeda izračunava udaljenost i množi ju sa nasumičnim brojem između 0 i 1. Novi podatak stavlja se na dobivenu poziciju. Analiza se ponavlja dok se ne izjednači broj klasa.

Prije izvođenja analize podatci su podijeljeni na train i test grupe. Balansiranje se provodi samo na train podatcima kako model ne bi bio pristran. Na slici 3.3 vidljivi su krajnji rezultati SMOTE analize odnosno jednak broj instanci obje klase. Ukupan broj train podatak se uduplao zbog analize.



Slika 3.3 Balansiranost train podataka

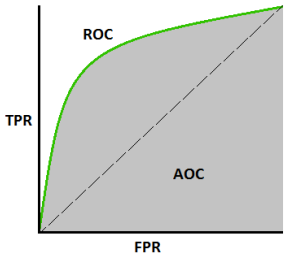
4. IZRADA I EVALUACIJA MODELA

Modeli korišteni u ovome projektu su dostupni u Sklearn biblioteci. Odabrani su modeli za logističku regresiju, Decision Tree klasifikator, Random Forest klasifikator i XGBoost klasifikator koji je dostupan u xgboost biblioteci. Kako bi se olakšala izrada i evaluacija modela napisane su funkcije kojima se predaje instanca modela i train i test podatci. Na slici 4.1 vidljiv je isječak koda za te dvije funkcije.

```
def model(classifier,x_train,y_train,x_test,y_test):

    classifier.fit(x_train,y_train)
    prediction = classifier.predict(x_test)
    cv = RepeatedStratifiedKFold(n_splits = 10,n_repeats = 3,random_state = 1)
    print("Cross Validation Score : ",'{0:.2%}'.format(cross_val_score(classifier,x_train,y_train,cv = cv,scoring = 'roc_auc').mean()))
    print("ROC_AUC Score : ",'{0:.2%}'.format(roc_auc_score(y_test,prediction)))
    plot_roc_curve(classifier, x_test,y_test)
    plt.title('ROC_AUC_Plot')
    plt.show()

def model_evaluation(classifier,x_test,y_test):


    cm = confusion_matrix(y_test,classifier.predict(x_test))
    names = ['True Neg','False Pos','False Neg','True Pos']
    counts = [value for value in cm.flatten()]
    percentages = ['{0:.2%}'.format(value) for value in cm.flatten()/np.sum(cm)]
    labels = [f'{v1}\n{v2}\n{v3}' for v1, v2, v3 in zip(names,counts,percentages)]
    labels = np.asarray(labels).reshape(2,2)
    sns.heatmap(cm,annot = labels,cmap = 'Blues',fmt ='')


    print(classification_report(y_test,classifier.predict(x_test)))
```

Slika 4.1 Funkcije za treniranje i evaluaciju modela

Pošto je korištena SMOTE analiza modeli se evaluiraju pomoću Cross Validation Score-a i ROC-AUC Score-a. Cross validation se koristi kako bi se procijenila sposobnost modela na neviđenim podatcima odnosno generalne mogućnosti modela na podatcima koji se nisu koristili za treniranje. ROC-AUC govori koliko je model precizan. Na slici 4.2 vidljiv je primjer ROC-AUC krivulje. Što je veća površina ispod krivulje to je model precizniji.

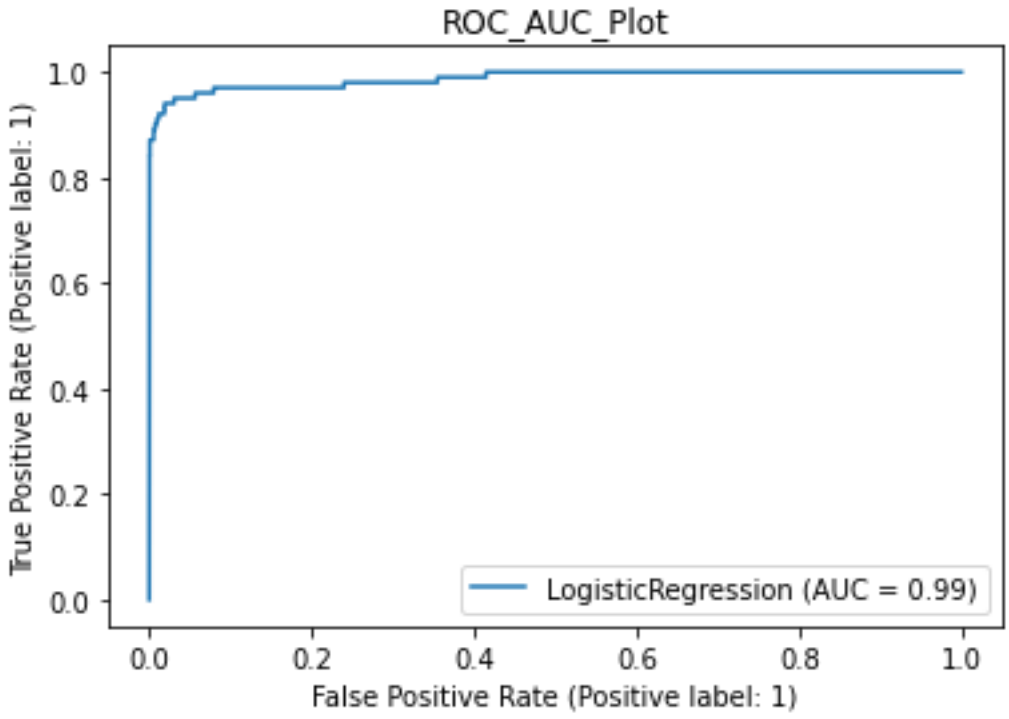


Slika 4.2 Primjer ROC-AUC krivulje

4.1. Logistička regresija

Logistička regresija je algoritam koji služi za računanje vjerojatnosti binarnih klasa. Analiziranjem veza između varijabli određuje se vjerojatnost za klasu što predstavlja izlaz iz modela. Određeni prag se postavlja za pojedinu klasu.

Istrenirani model dobio je cross validation score od 98.84% dok je ROC-AUC score bio 95.81%. Na slici 4.3 vidljiva je ROC-AUC krivulja za model.

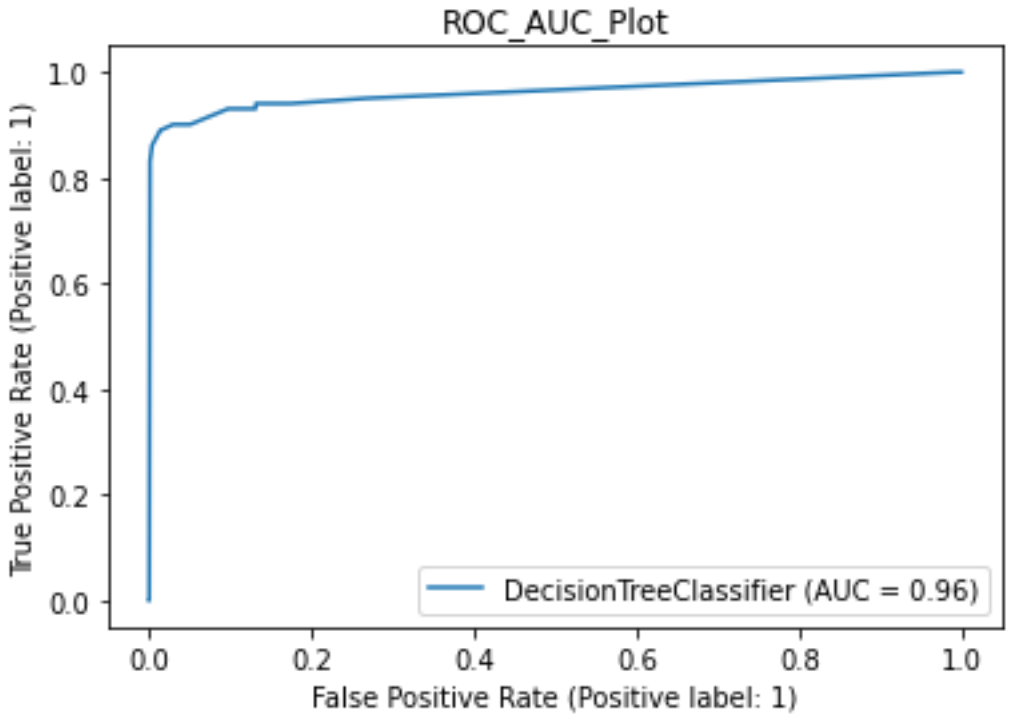


Slika 4.3 ROC-AUC krivulja za model logističke regresije

4.2. Klasifikator stabla odluke

Klasifikator satabla odluke radi tako da konstruira model odluka i njihovih mogućih posljedica u obliku stabla. Svaki unutarnji čvor predstavlja test značajke dok listovi predstavljaju oznake klasa. Da bi napravio predviđanje za novu instancu, algoritam slijedi put odluke u stablu, počevši od korijena i završavajući u čvoru lista, na temelju vrijednosti značajki. Oznaka klase dosegnutog lisnog čvora je predviđanje za novu instancu.

Istrenirani model dobio je cross validation score od 98.00% dok je ROC-AUC score bio 93.16%. Na slici 4.4 vidljiva je ROC-AUC krivulja za model.

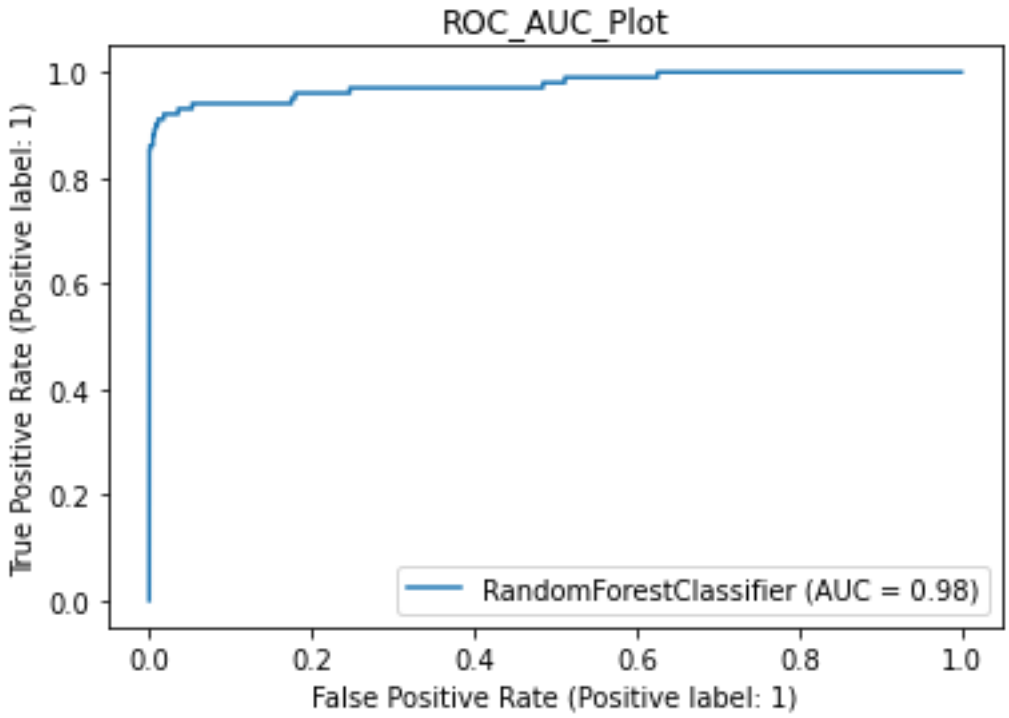


Slika 4.4 ROC-AUC krivulja za model klasifikatora stabla odluke

4.3. Klasifikator nasumične šume

Klasifikator nasumične šume radi tako da za vrijeme treninga konstruira veliki broj stabala odluke i ispisuje klasu pojedinačnog stabla. Ključna ideja iza nasumičnih šuma je ukrasiti stabla, tj. napraviti stabla koja su što neovisnija, tako da prosjek procjene bude bolji od procjene bilo kojeg pojedinačnog stabla. Da bi se to postiglo algoritam pri svakom grananju stabla odabire nasumičan podskup značajki za grananje. Ovo čini stabla raznolikima i smanjuje overfitting.

Istrenirani model dobio je cross validation score od 98.71% dok je ROC-AUC score bio 93.81%. Na slici 4.5 vidljiva je ROC-AUC krivulja za model.

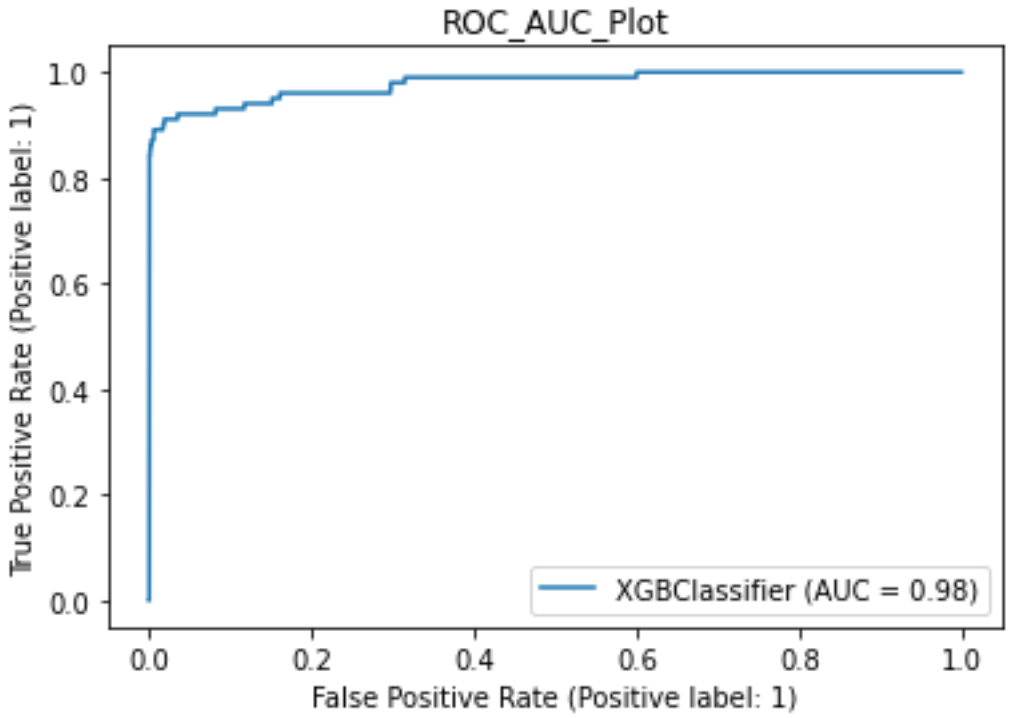


Slika 4.5 ROC-AUC krivulja za model klasifikatora nasumične šume

4.4. XGBoost klasifikator

XGBoost je algoritam za povećanje gradijenta za stabla odluke. Implementacija je frameworka za povećanje gradijenta koja je skalabilna, brza i točna. Algoritam radi stvaranjem skupa stabala odluke kako bi analizirao podatke za obuku i korištenjem postupka optimizacije spuštanja gradijenta kako bi se smanjila funkcija gubitka i poboljšala točnost predviđanja.

Istrenirani model dobio je cross validation score od 99.86% dok je ROC-AUC score bio 93.96%. Na slici 4.6 vidljiva je ROC-AUC krivulja za model.



Slika 4.6 ROC-AUC krivulja za model XGBoost klasifikatora

5. ZAKLJUČAK

Klasifikacija se sve više primjenjuje u svakodnevnom životu kako bi se obradile velike količine podataka u malo vremena što čovjek ručno ne može napraviti. U radu su pokazani samo neki modeli kojima se postiže prepoznavanje prijevara kreditnim karticama. S obzirom na rezultate evaluacije modeli XGBoost klasifikatora i klasifikatora nasumične šume pokazali su da imaju najveću preciznost određivanja prevare. Ostali modeli se mogu poboljšati podešavanjem parametara što je različito za svaki dataset.