Matematički fakultet,
Univerzitet u Beogradu

Fakultetski projekat

Predviđanje bankrota kompanija

Staša Tonić 157/2019
Natalija Lazić 230/2018
Ognjen Đuković 8/2018

Maj 2023.

Sadržaj

1 Uvod

Globalna ekonomska kriza 2008. godine pokrenula je mnoge razgovore na temu stabilnosti tržišta i poslovanja kompanija, kao i alata koji se mogu upotrebiti da se isti predvide.

Modeli koji mogu predvideti bankrot od velike su važnosti ne samo za pojedinačne kompanije, već i za globalnu ekonomiju. Oni mogu pomoći investitorima i drugim zainteresovanim stranama da procene finansijski rizik pri ulaganju u određenu kompaniju, analitičarima da uoče kompanije koje su u finansijskoj opasnosti, samim kompanijama da uoče probleme u svom poslovanju i na vreme preduzmu potrebne mere za prevenciju bankrota, poput smanjenja troškova, promene načina poslovanja, povećanja prihoda i slično. Korišćenje ovih modela takođe može pomoći pri donošenju bolje informisanih odluka kada su u pitanju kupovina i prodaja akcija, izbor dobavljača ili kupaca, kao i odobravanje kredita. Ideja iza njih je posmatranje i analiziranje KPI-jeva (Key Performance Indictors) koji predstavljaju metrike koje kompanije koriste kako bi procenile svoj uspeh u ispunjavanju određenih poslovnih ciljeva i objektiva. Neki od čestih KPI-jeva za procenu finansijski performansi su profitabilnost, rast prihoda i odnos duga i kapitala. Njihova analiza može ukazati na određene poteškoće u poslovanju i predvideti da li kompanija pokazuje simptome bankrotstva ili ne. Uočavanje poteškoća nadalje pomaže kompaniji da donese odgovarajuće odluke.

Kada kompanija bankrotira, postoje dva moguća scenarija - može pokušati reorganizaciju putem unapred pripremljenog plana reorganizacije (UPPR) ili otići pod stečaj, što znači da se automutoski povlači sa Beogradske berze. Stečaj može biti prijavljen od strane same kompanije ili nekog od njenih poverioca. Odlazak pod stečaj podrazumeva da je poslovni model te kompanije došao u situaciju gde više nije održiv i nije u stanju da izmiri svoje dugove. Kompanija se likvidira i sud postavlja stečajnog upravnika koji pokreće formalni proces u okviru kog se imovina te kompanije prodaje da bi se namirili njeni poverioci. Nadalje se resursi kompanije raspoređuju u između poverioca u skladu sa zakonom. Prodaja imovine može se odnositi na prodaju zaliha, opreme, nekretnina i drugih imovinskih predmeta. Nakon što su svi dugovi isplaćeni, preostala sredstva se distribuiraju vlasnicima kompanije ili drugim povezanim stranama i tada se likvidacija kompanije završava. Međutim, može se desiti da sredstva od prodaje imovine nisu dovoljna da se izmire sve finansijske obaveze. Ukoliko se to desi, poverioci će biti izmireni prema prioritizaciji koju propisuje zakon - uobičajeno je da se prvo izmire dugovi prema poveriocima koji imaju obezbeđenje na imovini kompanije (npr. hipoteku), zatim se izmiruju porezi i doprinosi, a na kraju se izmiruju neplaćeni dugovi. Ukoliko prihod od prodaje imovine ne pokrije sva dugovanja, preostali dugovi će ostati neizmireni. Odlazak pod stečaj predstavlja veoma komplikovan i dug proces koji može trajati nekoliko meseci ili, u zavisnosti od veličine i složenosti kompanije, čak i godina.

2 Opis baze i priprema podataka za obradu

Baza podataka koju ćemo koristiti za analizu sastoji se od podataka prikupljenih iz *Taiwan Economic Journal*-a za godine 1999 - 2009. Bankrot kompanije definisan je na osnovu poslovnih propisa tajvanske berze.

Skup podataka se sastoji od zavisne promenljive koja nosi informaciju o tome da li je kompanija bankrotirala i 95 prediktora. Zavisna promenljiva je binarnog tipa, gde 1 predstavlja bankrot, dok 0 označava da je kompanija nastavila da postoji. Prediktori obuhvataju različite informacije o poslovanju kompanije poput povraćaja sredstava, bruto dobiti, operativnih i neto prihoda i rashoda, tokova novca, dugovanja itd. Baza se sastoji od 6819 opservacija.

Baza podataka ima stroge zahteve za kompanije uključene u njenu analizu. Prvo, sve kompanije moraju imati na raspolaganju najmanje tri godine potpunih javnih informacija pre nego što dožive finansijsku krizu. Pored toga, zbog potreba poređenja, skup podataka uključuje samo kompanije sa značajnim brojem sličnih kompanija u industriji.

Skup podataka čine proizvodne kompanije, sa 346 firmi u industrijskom i elektronskom sektoru, te 39 uslužnih preduzeća i 93 kompanije iz drugih sektora. Zbog neravnomerne raspodele kompanija u ovim kategorijama, podaci su veoma nebalansirani, što predstavlja izazov za tačna predviđanja prilikom primene našeg modela na širu populaciju kompanija.

Svi podaci unutar baze su potpuni i skalirani između 0 i 1 korišćenjem formule:

$$x_{skalirano} = \frac{x - x_{min}}{x_{max} - x_{min}}$$

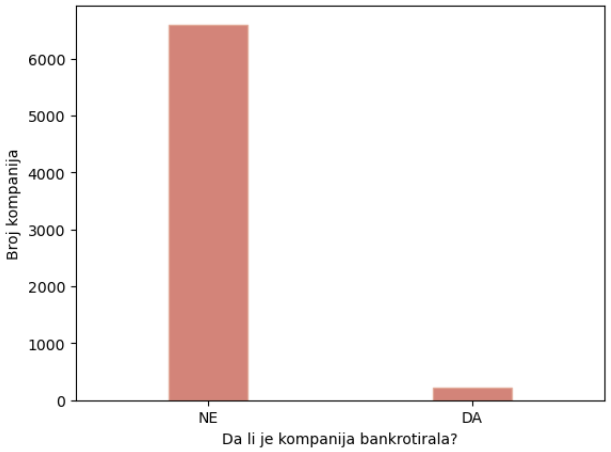
U okviru projekta predstavićemo različite algoritme klasifikacije na skupu podataka u cilju predviđanja bankrota kompanija sa zadovoljavajućom tačnošću mnogo pre stvarnog događaja. Valja napomenuti da je za ovakvu vrstu problema, pored tačnosti predviđanja, neizostavan faktor i isplativost. Kao što se može zaključiti, mnogo je isplativije predvideti da će kompanija bankrotirati i da ta pretpostavka ne bude tačna, nego obrnuto.

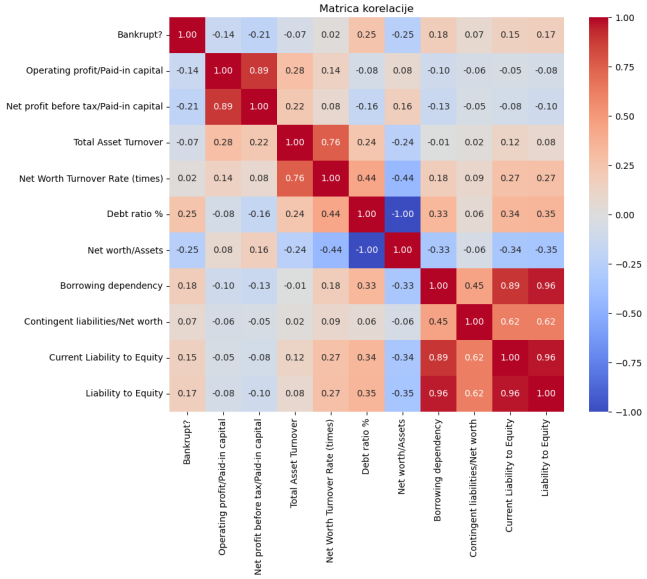
2.1 Grafički prikaz podataka

Skup podataka sadrži tri kategoričke promenljive, pri čemu je kategorička promenljiva *Net Income Flag*, koja predstavlja oznaku neto prihoda, za sve kompanije jednaka 1. Ovo ukazuje da je neto prihod za poslednje dve godine bio negativan u okviru svake kompanije, te je možemo isključiti iz dalje analize.

Kategorička promenljiva *Liability-Assets Flag* daje informaciju o tome da li ukupna obaveza premašuje ukupnu imovinu, tj. da li kompanija duguje više novca od vrednosti onoga što poseduje. Ova pojava zabeležena je samo kod 6 kompanija.

Na sledećem *bar plot*-u možemo videti broj kompanija koje su bankrotirale i broj kompanija koje nisu. Primećujemo da je broj kompanija koje nisu bankrotirale znatno veći od onih koje su doživele bankrot.





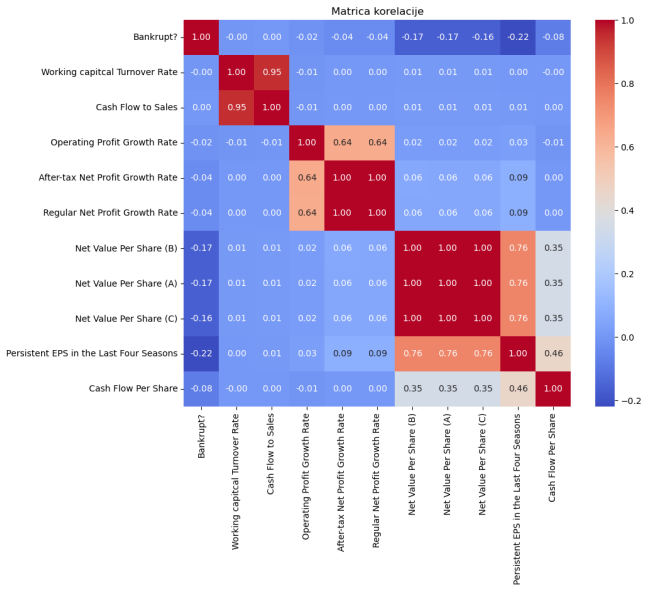
Linearnu zavisnost varijabli možemo prikazati pomoću matrice korelacije. Na slikama su izdvojene značajne veze između prediktora kao i njihova korelisanost sa zavisnom promenljivom.

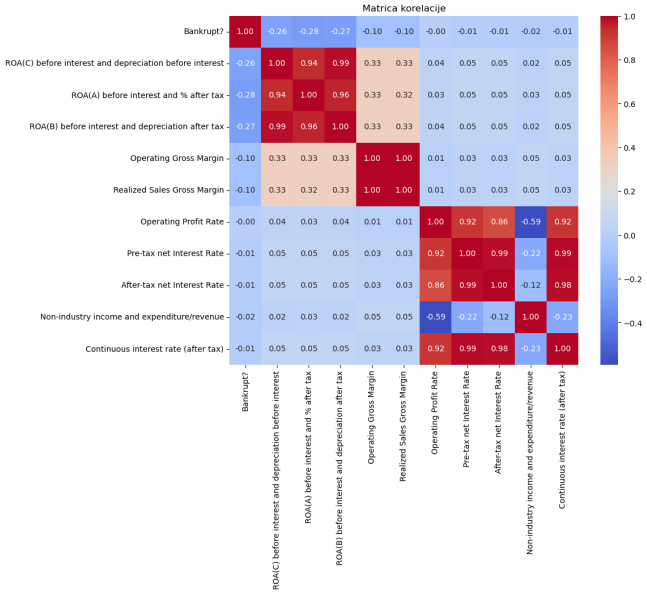
Primećujemo da su varijable *Operating profit/Paid-in capital* i *Net profit before tax/Paid-in capital* pozitivno korelisane, kao i *Borrowing dependency*, *Contingent liabilities/Net worth*, *Current Liability to Equity* i *Liability to Equity*.

Promenljive *Debt ratio %* i *Net worth/Assets* su negativno korelisane, sa visokom stopom korelacije.

Takođe, promećujemo da su promenljive *Net Value Per Share (B)*, *Net Value Per Share (A)* i *Net Value Per Share (C)* visoko pozitivno korelisane.

Analogno primećujemo veze i između ostalih prediktora. Valja napomenuti da korelacija pojedinačnih prediktora sa zavisnom promenljivom postoji, ali se ne može smatrati statistički značajnom.





2.2 Metode balansiranja podataka

Neuravnoteženost u bazi podataka odnosi se na situaciju kada postoje značajne razlike u broju primeraka različitih klasa u skupu podataka. U ovom slučaju, klasa kompanija koje su banrkotirale ima znatno manje primeraka od klase kompanija koje nisu bankrotirale, što može imati ozbiljne posledice za analizu podataka i performanse modela mašinskog učenja. Zbog ove devijacije, modeli će se više fokusirati na učenje kako klasifikovati veću klasu.

Gorepomenuti problem možemo demostrirati na primeru modela *k najbližih suseda (kNN)*. Očekujemo da će ovaj model predvideti da skoro nijedna kompanija neće bankrotirati.

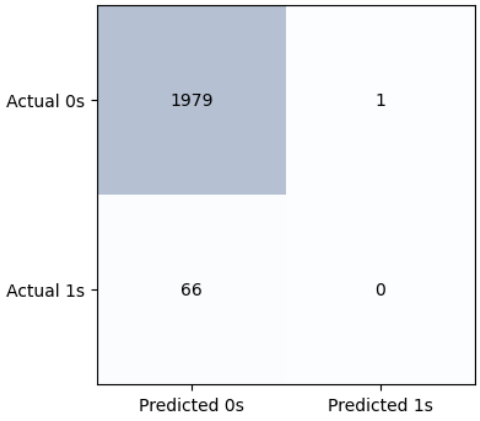
Pre konsturisanja modela, potrebno je podeliti podatke na test i trening skupove:

```
X = df.iloc[:,1:96]
y = df.iloc[:,0]
train_X, test_X, train_y, test_y = train_test_split(X, y, test_size=0.3,
    random_state=420, stratify=y)
```

Kao što smo već pomenuli, podaci unutar baze su unapred skalirani. Sada možemo primeniti *kNN* algoritam za proizvoljno k, npr. $k = 10$ i konstruisati naš model na sledeći način:

```
model = KNeighborsClassifier(n_neighbors=10)
model.fit(train_X, train_y);
pred_y = model.predict(test_X)
```

Grafički prikaz matrice konfuzije daje nam uvid u broj ispravno i pogrešno klasificiranih instanci za svaku klasu.



Slika 1: Matrica konfuzije

Matrica konfuzije ovog modela nam govori da naš klasifikator nije dobar jer pravi veliki broj *skupljih* grešaka. Dakle, mnogo je veći broj kompanija koje su bankrotirale, premda je naš model predvideo da će njihov način poslovanja biti održiv, nego obrnuto. Kao što smo već napomenuli, ovaj tip grešaka je puno manje isplativ od predviđanja bankrota unutar kompanija koje će nastaviti da postoje.

Postoje različite strategije koje se mogu primeniti za rešavanje problema neuravnoteženosti podataka. Neke od mogućih strategija su:

- *Biranje dobre funkcije evaluacije* - U slučaju neuravnotežene baze podataka, model koji pretežno predviđa dominantnu klasu može imati visoku vrednost funkcije *accuracy*, iako je prilično neuspešan prilikom predviđanja manje zastupljenih klasa. Stoga, prilikom rada sa neuravnoteženim podacima, za funkciju evaluacije treba odabrati neku od funkcija *Precision*, *Recall*, *F1 Score* i slično.

- *Reuzorkovanje (Resampling)* - Metodološki pristup koji se koristi za rešavanje problema neuravnoteženih podataka tako što manipuliše veličinom uzorka podataka u cilju poboljšanja performansi modela mašinskog učenja. Osnovni tipovi reuzorkovanja su *undersampling* (podselekcija) i *oversampling* (prekomerna selekcija).

2.2.1 Undersampling

Podselekcija, tj.*undersampling* je tehnika koja podrazumeva smanjenje broja uzoraka zastupljenije klase u cilju ravnoteže između klasa. Na primer, neka klasu 1 čini 1000 uzoraka, a klasu 2 čini 100 uzoraka. Prilikom korišćenja ove metode, iz klase 1 uzimamo 100 uzoraka, te je nakon primene *undersampling*-a broj uzoraka u obe klase izbalansiran.

Potrebno je napomenuti da ova tehnika može dovesti do preterane prilagođenosti, tj. *overfitting*-a modela, kao i gubitka informacija iz podataka. Zbog svega navedenog, *undersampling* se koristi samo u slučaju kada je to neophodno i kada se očekuje da broj uzoraka neće uticati na performanse modela.

U nastavku ćemo pokazati primenu *undersampling*-a na realnim podacima. Koristićemo ovu tehniku da rešimo problem neuravnoteženih podataka, te napraviti model korišćenjem *kNN* algoritma na sređenim podacima.

Pre konstrukcije modela, potrebno je da ranije definisane trening skupove podelimo tako što ćemo razdvojiti kompanije unutar njih na osnovu toga da li su bankrotirale ili ne. Segregaciju vršimo na sledeći način:

```
train_X_jesu_bankrotirali = train_X[train_y == 1]
train_X_nisu_bankrotirali = train_X[train_y == 0]
train_y_jesu_bankrotirali = train_y[train_y == 1]
train_y_nisu_bankrotirali = train_y[train_y == 0]
```

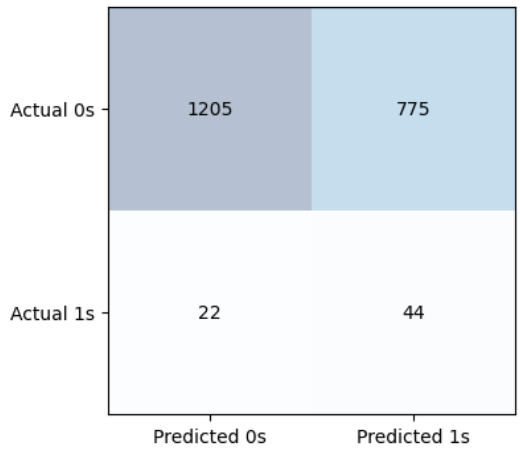
Pravimo novi trening skup koji će se sastojati iz izbalansiranih (uravnoteženih) podataka:

```
n = len(train_y[train_y == 1])
N = len(train_y[train_y == 0])
N_list = list(np.arange(1, N))
idx = sample(N_list,n)
train_X_nisu_bankrotirali_us = train_X_nisu_bankrotirali.iloc[idx,]
train_y_nisu_bankrotirali_us = train_y_nisu_bankrotirali.iloc[idx]
train_X_us = pd.concat([train_X_jesu_bankrotirali,
   train_X_nisu_bankrotirali_us])
train_y_us = pd.concat([train_y_jesu_bankrotirali,
   train_y_nisu_bankrotirali_us])
```

Sada možemo primeniti *kNN* algoritam. Uzećemo da je $k = 15$, kao vrednost broja k za koju smo dobili najbolje predviđanje. Konstruišemo naš model i njegovu matricu konfuzije kao:

```
model = KNeighborsClassifier(n_neighbors=15)
model.fit(train_X_us, train_y_us);
pred_y = model.predict(test_X)
```

Matrica konfuzije novog modela ukazuje na primetno bolje performanse istog, što možemo zaključiti na osnovu njenog grafičkog prikaza:



Slika 2: Matrica konfuzije za *undersampling* model

Primećujemo da se broj grešaka prve vrste smanjio, što možemo navesti kao primaran cilj ove popravke modela. Takođe, možemo videti da se broj grešaka druge vrste, takozvanih *lažnih uzbuna*, znatno povećao. Međutim, kao što smo ranije napomenuli, veći broj grešaka druge vrste je i dalje isplativiji u kontekstu predikcije ishoda o budućnosti kompanija.

Kao mere kvaliteta u cilju procene performansi našeg modela koristimo funkcije evaluacije *F1 Score*, *AUC ROC* i *Precision*. Njihovom primenom dobijamo sledeće rezultate:

```
mere_kvaliteta(test_y, pred_y)
```

F1 Score : 0.43

AUC ROC : 0.63

Precision : 0.05

F1 Score od 0.43 govori da naš model ima određeni nivo efikasnosti prilikom identifikacije pozitivnih instanci dok minimizira lažno pozitivne i lažno negativne instance.

Vrednost *AUC ROC*-a je 0.64, što implicira da model ima umerenu diskriminatornu moć u razlikovanju pozitivnih i negativnih instanci. Ova vrednost sugeriše da model funkcioniše bolje od slučajnog nagađanja, ali da nije veoma precizan u svojim predviđanjima.

Preciznost modela od 0.05 ukazuje na to da model ima veoma nisku sposobnost da ispravno identifikuje pozitivne instance. Dakle, od svih slučajeva koje je model predvideo kao pozitivne, samo 5% njih je stvarno pozitivno, dok je preostalih 95% lažno pozitivno.

U slučaju kada ne želimo da se odreknemo podataka, možemo napraviti više uravnoteženih trening skupova. Sastojaće se od kompanija koje su bankrotirale, dok će kompanije koje nisu bankrotirale biti jednako raspodeljene svakom od njih.

Za početak, određujemo broj trening skupova koje ćemo koristiti za pravljenje modela:

```
broj_trening_skupova = N//n
```

Ukupan broj trening skupova biće 29. Konstruišemo ih kao:

```
train_X_us2= {}
train_y_us2= {}
i = 0
while i < 29:
    train_X_tt = train_X_nisu_bankrotirali.iloc[i*n:(i+1)*n]
    train_y_tt = train_y_nisu_bankrotirali.iloc[i*n:(i+1)*n]

    train_X_t = pd.concat([train_X_jesu_bankrotirali, train_X_tt])
    train_y_t = pd.concat([train_y_jesu_bankrotirali, train_y_tt])

    train_X_us2[i] = train_X_t
    train_y_us2[i] = train_y_t

    i += 1
```

Ovom implementacijom odrekli smo se podataka o određenom broju kompanija koji je jednak ostatku pri deljenju $\frac{N}{n}$. Kako taj broj ne može biti previše veliki, izbacivanje kompanija iz modela ne bi trebalo značajno uticati na isti.

Sada možemo primeniti kNN algoritam za $k = 15$ na svaki od novonastalih trening skupova:

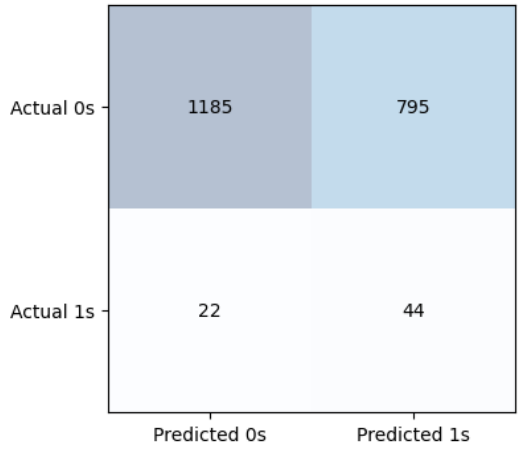
```
model = KNeighborsClassifier(n_neighbors=15)

i = 0
model.fit(train_X_us2[i], train_y_us2[i]);
pred_y_m = np.array([model.predict(test_X)])

i = 1

while i < 29:
    model.fit(train_X_us2[i], train_y_us2[i])
    pred_y_= np.array([model.predict(test_X)])
    pred_y_m = np.vstack([pred_y_m, pred_y_])
    i += 1
```

Kao konačnu procenu uzećemo ocenu koja se najčešće pojavljuje u svim modelima:

```
pred_y = najcesca_predikcija(pred_y_m[:,], broj_trening_skupova)
```

Konstruišemo matricu konfuzije modela koji čini 29 trening skupova i grafički je prikazujemo:

Kao mere kvaliteta u cilju procene performansi našeg novog modela koristimo funkcije evaluacije *F1 Score*, *AUC ROC* i *Precision*. Njihovom primenom dobijamo sledeće rezultate:



Slika 3: Matrica konfuzije za *undersampling* model

```
mere_kvaliteta(test_y, pred_y)
```

F1 Score : 0.42

AUC ROC : 0.63

Precision : 0.05

Možemo primetiti da ove vrednosti funkcija evaluacije nisu bolji od rezultata dovijenih na osnovu modela koji čini jedan trening skup. Međutim, od velikog su značaja zato što, usled ovako definisane implementacije, znamo da njihove vrednosti nisu posledica *overfitting*-a.

Ukoliko smatramo da je svaki podatak neizostavan, možemo, na primer, gorepomenuti ostatak kompanija podeliti po trening skupovima. Oni će tada imati jednu kompaniju koja nije bankrotirala više od ostalih, što neće negativno uticati na model. Međutim, primenom ovog postupka ne očekujemo bitno drugačije rezultate u odnosu na prethodnu diskusiju.

2.2.2 Oversampling

Prekomerna selekcija, tj. *oversampling* je tehnika koja podrazumeva dodavanje novih uzoraka u manje zastupljenu klasu u cilju ravnoteže između klasa. Ova tehnika se najčešće primenjuje kada je broj uzoraka bitan faktor u performansama modela.

Najjednostavniji pristup u okviru *oversampling*-a je umnožavanje postojećih uzoraka manje zastupljene klase, što se naziva *up-sampling*. Na primer, neka klasu 1 čini 1000 uzoraka, a klasu 2 čini 100 uzoraka. Prilikom korišćenja ove metode možemo udesetostručiti broj uzoraka klase 2 i time izbalansirati broj uzoraka u obe klase.

Potrebno je napomenuti da ova tehnika može dovesti do preterane prilagođenosti, tj. *overfitting*-a modela, kao i gubitka informacija iz podataka. Do gubitka informacija dolazi jer se postojeći uzorci ponavljaju bez dodavanja novih informacija. Zbog svega navedenog, često se primenjuju složenije tehnike *oversampling*-a, kao što su SMOTE (*Synthetic Minority Over-sampling Technique*) i ADASYN (*Adaptive Synthetic Sampling*) tehnike.

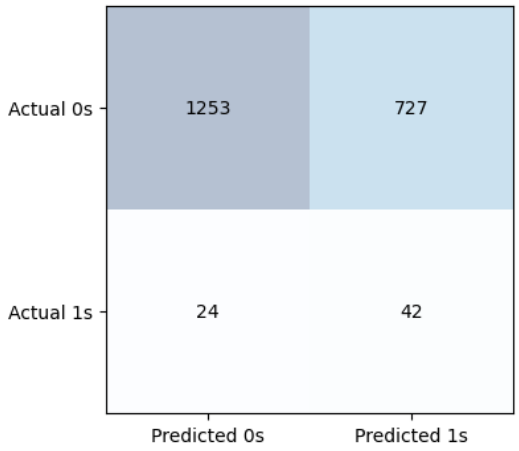
U nastavku ćemo pokazati primenu SMOTE tehnike na realnim podacima. Koristićemo ovu tehniku da rešimo problem neuravnoteženih podataka, te napraviti model korišćenjem *kNN* algoritma na sređenim podacima.

```
smote = SMOTE(random_state=42)
train_X_smote, train_y_smote = smote.fit_resample(train_X, train_y)
```

Sada možemo primeniti *kNN* algoritam. Ovako definisan trening skup sastoji od velikog broja elemenata, te je potrebno da broj k bude dovoljno veliki. Uzećemo da je $k = 150$, kao vrednost broja k za koju smo dobili dobre rezultate. Konstruišemo naš model i matricu konfuzije kao:

```
model = KNeighborsClassifier(n_neighbors=150)
model.fit(train_X_smote, train_y_smote)
pred_y = model.predict(test_X)
precision = [metrics.precision_score(test_y, pred_y)]
```

Grafički prikazujemo matricu konfuzije našeg modela:



Slika 4: Matrica konfuzije za *oversampling* model

Kao i do sad, kao mere kvaliteta koristimo funkcije evaluacije *F1 Score*, *AUC ROC* i *Precision*. Njihovom primenom dobijamo sledeće rezultate:

```
mere_kvaliteta(test_y, pred_y)
```

F1 Score : 0.44

AUC ROC : 0.63

Precision : 0.05

Možemo primetiti da matrica konfuzije ovog modela ne daje primetno bolje rezultate. Takođe, vrednosti funkcija evaluacije su zanemarljivo različite u odnosu na *undersampling* tehniku.

Dakle, možemo zaključiti da obe metode balansiranja podataka, prilikom obrade podatka iz zadate baze, imaju određeni nivo efikasnosti i u nekoj meri poboljšavaju model. Ipak, na osnovu rezultata primećujemo da predikcije koje naš model pravi i nakon ove korekcije nisu najpouzdanije.

2.3 Metode redukcije dimenzionalnosti podataka

Baza podataka koju analiziramo sastoji se od velikog broja prediktora. U okviru grafičke analize podataka, primetili smo da su mnogi od njih su međusobno korelisani. Pojava korelacije među prediktorima može dovesti do problema prilikom analize, interpretacije i vizualizacije podataka.

Kako bismo rešili problem korelacije između prediktora, možemo koristiti neku od sledećih metoda:

- *Odabir prediktora (Feature Selection)* - Pristup koji za konstrukciju modela bira najrelevantnije prediktore i izostavlja one koji su visoko međusobno korelisani. Neke od tehnika koje se mogu koristiti za implementaciju ove metode su *ANOVA*, *Ridge* i *Lasso* regresija.
- *Smanjenje broja prediktora (Feature Extraction)* - Pristup koji koristi matematičke tehnike za transformisanje originalnih prediktora u novi, kompaktniji skup prediktora. Novonastali skup će imati manju dimenziju i zadržaće što više informacija koje je inicijalni skup prediktora posedovao. Neki od algoritama za smanjenje broja prediktora su *Principal Component Analysis (PCA)*, *t-distributed Stochastic Neighbor Embedding (t-SNE)* i *Uniform Manifold Approximation and Projection for Dimension Reduction (UMAP)*.
- *Metode ansambla (Ensemble Methods)* - Pristup koji podrazumeva kombinaciju različitih algoritama za smanjenje dimenzionalnosti podataka.

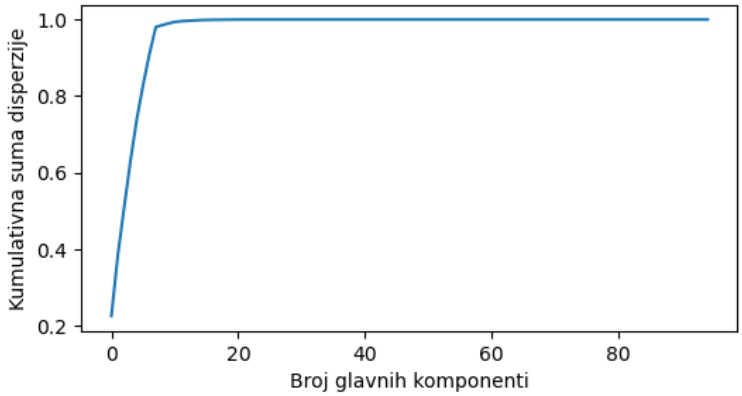
2.3.1 PCA - Analiza glavnih komponenti

Pokazaćemo primenu *Analize glavnih komponenti* na realnim podacima. Koristićemo ovu tehniku da smanjimo dimenzionalnost i rešimo problem korelacije između prediktora, te napraviti model logističke regresije na osnovu sređenih podataka.

Kao i ranije, podatke unutar baze delimo na test i trening skup. Zatim pravimo transformaciju našeg trening skupa u okviru koje njegovi prediktori postaju glavne komponente:

```
X = df.iloc[:,1:96]
y = df.iloc[:,0]
train_X, test_X, train_y, test_y = train_test_split(X, y, test_size=0.3,
    random_state=7, stratify=y)
pca = PCA().fit(train_X)
```

Nacrtaćemo grafik kumulativne sume disperzije, te na osnovu njega pokušati da zaključimo koliko infomacija prilikom redukcije zadržava svaka dimenzija i identifikujemo one dimenzije koje doprinose većoj varijabilnosti.

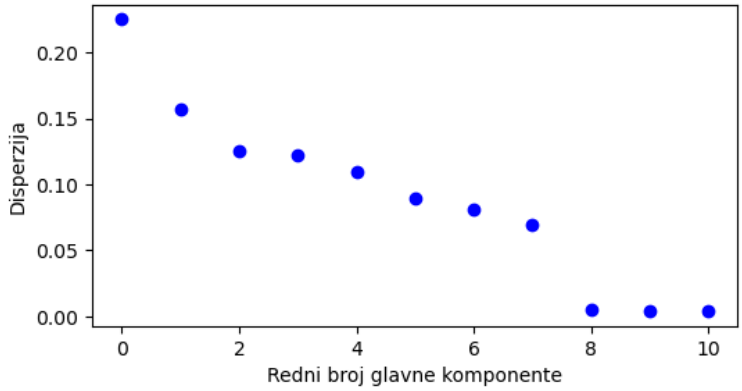


Slika 5: Grafik kumulativne sume disperzije

Cilj nam je da pronađemo dovoljan broj glavnih komponenti, tako da njihov udeo u disperziji bude 99%. To možemo uraditi na sledeći način:

```
pca = PCA(n_components=0.99).fit(train_X)
n = pca.n_components_
```

Dakle, dovoljan broj glavnih komponenti čije disperzije čine 99% ukupne disperzije je 11. Nacrtaćemo sada grafik udela u disperziji svake od glavnih komponenata.

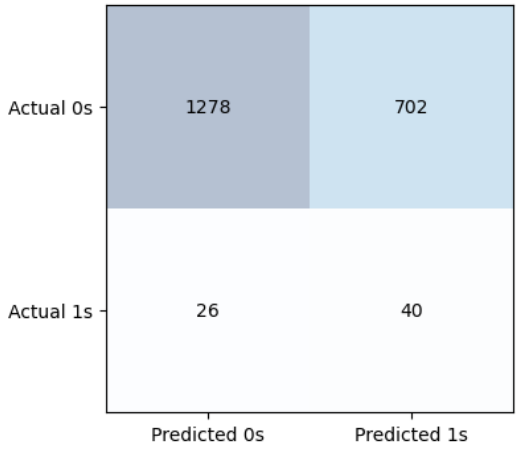


Slika 6: Grafik udela svake glavne komponente u disperziji

Na ovaj način konstruisali smo matricu transformacije dimenzije 95×11. Nju ćemo iskorisiti za transformaciju našeg trening i test skupa, te na novodobijeni trening skup primeniti logističku regresiju kao:

```
train_X_pca = pca.fit_transform(train_X)
test_X_pca = pca.transform(test_X)
classifier = LogisticRegression(random_state = 4)
classifier.fit(train_X_pca, train_y)
pred_y = classifier.predict(test_X_pca)
```

Rezultati dobijeni konstrukcijom ovog modela bitno se ne razlikuju od prethodno navedenih, što možemo videti na osnovu grafičkog prikaza matrice konfuzije:



Slika 7: Matrica konfuzije za logističku regresiju

Kao i do sad, kao mere kvaliteta u cilju procene performansi modela logističke regresije koristimo funkcije evaluacije *F1 Score*, *AUC ROC* i *Precision*. Njihovom primenom dobijamo sledeće rezultate:

```
mere_kvaliteta(test_y, pred_y)
```

F1 Score : 0.44

AUC ROC : 0.63

Precision : 0.05

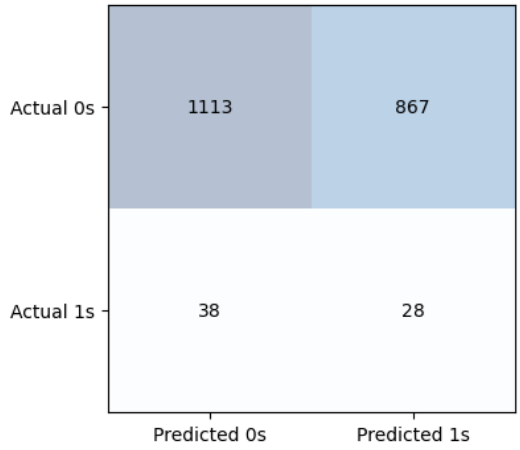
Kako se rezultati dobijeni modelom logističke regresije značajno ne razlikuju od prethodno dobijenih vrednosti, možemo analogno izvesti zaključke o kvalitetu modela.

Valja napomenuti i da možemo kombinovati *Analizu glavnih komponenti* PCA i prethodno navedenu tehniku *oversampling*-a pod nazivom SMOTE. Ovim postupkom dobijamo uravnoteženu bazu na kojoj možemo primeniti *kNN* algoritam.

Demonstriraćemo opisani postupak na podacima iz naše baze podataka korišćenjem sledećih funkcija:

```
train_X_pca2, train_y_pca2 = smote.fit_resample(train_X_pca, train_y)
model = KNeighborsClassifier(n_neighbors=150)
model.fit(train_X_pca2, train_y_pca2);
pred_y = model.predict(test_X_pca)
```

Kao i do sada, tačnost modela dobijenog kombinovanjem ove dve metode i *kNN* algoritma proveravamo preko matrice konfuzije. Posmatrajmo njen grafički prikaz:



Slika 8: Matrica konfuzije za PCA i SMOTE tehniku

Kao i do sad, kao mere kvaliteta u cilju procene performansi našeg modela koristimo funkcije evaluacije *F1 Score*, *AUC ROC* i *Precision*. Njihovom primenom dobijamo sledeće rezultate:

```
mere_kvaliteta(test_y, pred_y)
```

F1 Score : 0.38

AUC ROC : 0.49

Precision : 0.03

Možemo primetiti rezultati funkcija evaluacije modela dobijenog korišćenjem *Analize glavnih komponenti* PCA i SMOTE tehnike u kombinaciji sa *kNN* algoritmom za nijansu gori od rezultata ostalih modela koje smo koristili.

2.3.2 UMAP - Uniformna aproksimacija i projekcija prostora

Uniformna aproksimacija i projekcija prostora, tj. *Uniform Manifold Approximation and Projection* UMAP je algoritam koji se koristi za redukciju dimenzionalnosti, vizualizaciju i analizu visokodimenzionih podataka.

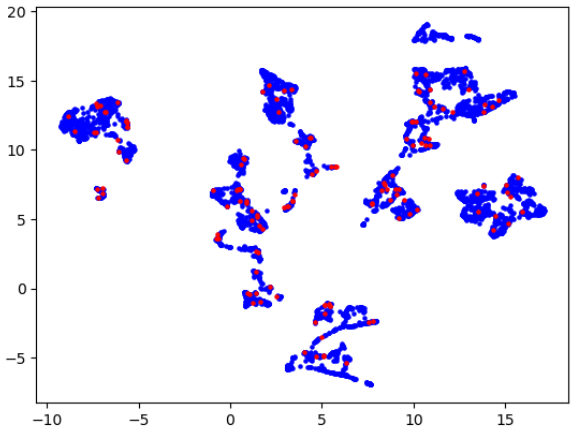
Najčešće za novu dimenziju podataka uzimaju se brojevi 2 ili 3 kako bi se podaci mogli grafički predstaviti. Ovim algoritmom prvo smanjujemo dimenziju cele baze, a zatim konstruišemo trening i test skup.

Glavna mana UMAP-a leži u tome što nemamo mogućnost direktnog ubacivanja novih podataka u model, već je prilikom dodavanja novih podataka potrebno ispočetka ponoviti konstrukciju modela.

Primenićemo UMAP algoritam na realne podatke iz naše baze. Za početak ćemo svesti bazu na dve dimenzije, pa podeliti podatke na trening i test skupove na sledeći način:

```
umap_ = umap.UMAP(n_components=2)
X_umap = umap_.fit_transform(X)
train_X_umap, test_X_umap, train_y_umap, test_y_umap = train_test_split(
    X_umap,y,test_size=0.3, random_state=42 )
```

Kako je dimenzija nove baze 2, podatke možemo vizuelno prikazati u dvodimenzionalnom sistemu pomoću sledećeg grafika:

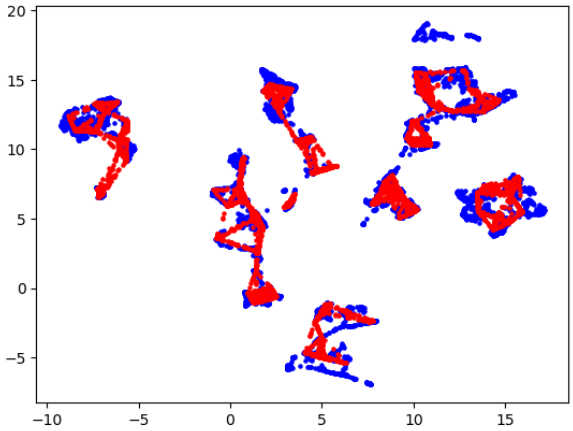


Slika 9: Grafik podataka korišćenjem UMAP algoritma

Pošto na grafiku ne uočavamo jasnu razdvojenost podataka, možemo proširiti trening skup koristeći gorepomenutu *oversampling*-a pod nazivom SMOTE:

```
smote = SMOTE(random_state=42)
train_X_umap2, train_y_umap2 = smote.fit_resample(train_X_umap, train_y_umap)
```

Nakon primene ove tehnike, ponavljamo postupak grafičkog prikazivanja dobijenih podataka i tada dobijamo:

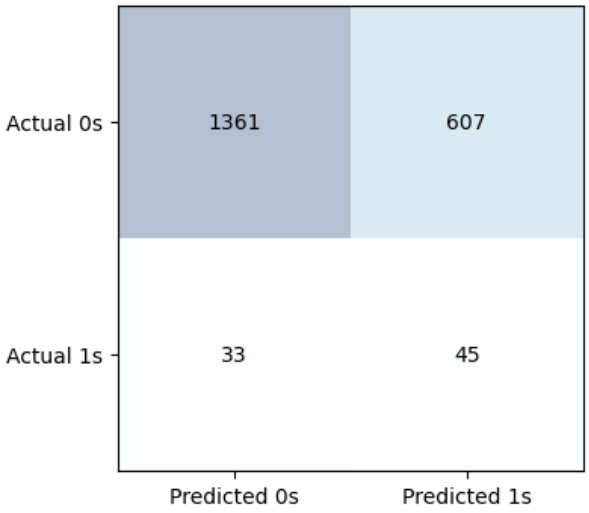


Slika 10: Grafik podataka korišćenjem UMAP algoritma i SMOTE tehnike

Vidimo da i dalje ne postoji jasna razdvojenost između kompanija, ali možemo primetiti da na određenim mestima postoje mali klasteri tačaka iste boje. Ova pojava nam implicira da možemo isprobati kNN algoritam. Kako klasteri koje smo uočili nisu veliki, za vrednost broja k uzećemo $k = 50$ i primeniti algoritam:

```
model = KNeighborsClassifier(n_neighbors=50)
model.fit(train_X_umap2, train_y_umap2);
pred_y = model.predict(test_X_umap)
```

Za određivanje preciznosti modela ponovo koristimo matricu konfuzije. Posmatrajmo njen grafički prikaz:



Slika 11: Matrica konfuzije za UMAP i SMOTE tehniku

Kao i do sad, kao mere kvaliteta u cilju procene performansi našeg modela koristimo funkcije evaluacije *F1 Score*, *AUC ROC* i *Precision*. Njihovom primenom dobijamo sledeće rezultate:

```
mere_kvaliteta(test_y_umap, pred_y)
```

F1 Score : 0.47

AUC ROC : 0.63

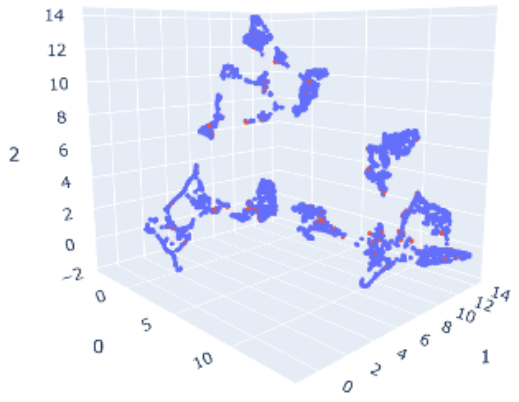
Precision : 0.07

Rezultati koje smo dobili slični su rezultatima dobijenim korišćenjem običnog trening skupa. Ovaj ishod je očekivan, jer je ideja UMAP algoritma da se prilikom preslikavanja tačaka u manjedimenzioni prostor čuvaju odnosi između njihovi rastojanja. Naime, cilj algoritma je da tačke koje su se nalazile blizu u početnom prostoru budu blisko pozicionirane u prostoru nove dimenzije i obratno.

Ponovo ćemo primeniti UMAP algoritam na realne podatke iz naše baze. Sada ćemo svesti bazu na tri dimenzije, pa podeliti podatke na trening i test skupove na sledeći način:

```
umap_ = umap.UMAP(n_components=3)
X_umap3 = umap_.fit_transform(X)
train_X_umap3, test_X_umap3, train_y_umap3, test_y_umap3 = train_test_split(
    X_umap3,y,test_size=0.3, random_state=42 )
colors = train_y_umap3.astype(str)
```

Kako je dimenzija nove baze 3, podatke možemo prikazati u trodimenzionalnom sistemu pomoću sledećeg grafika:

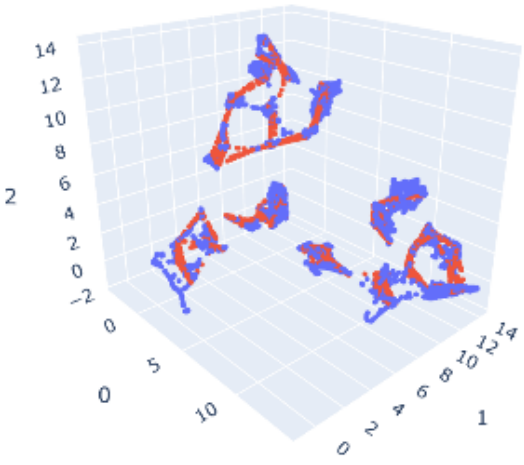


Slika 12: Grafik podataka korišćenjem UMAP algoritma

Kao i malopre, na grafiku ne uočavamo jasnu razdvojenost podatka, te proširujemo trening skup koristeći SMOTE tehniku:

```
smote = SMOTE(random_state=42)
train_X_umap4, train_y_umap4 = smote.fit_resample(train_X_umap3,
    train_y_umap3)
colors = train_y_umap4.astype(str)
```

Nakon primene ove tehnike, ponavljamo postupak grafičkog prikazivanja dobijenih podataka i tada dobijamo:

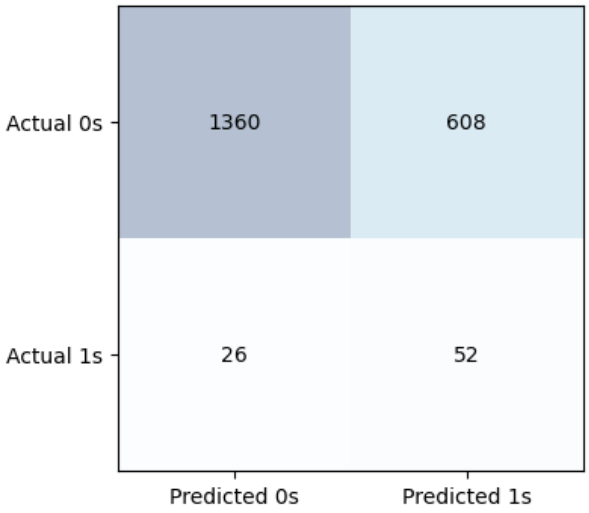


Slika 13: Grafik podataka korišćenjem UMAP algoritma i SMOTE tehnike

Vidimo da i dalje ne postoji jasna razdvojenost između kompanija, ali možemo primetiti da na određenim mestima postoje mali klasteri tačaka iste boje. Ponavljamo postupak odozgo i na novodobijene podatke primenjujemo *kNN* algoritam. Kako klasteri koje smo uočili nisu veliki, za vrednost broja k uzećemo $k = 50$ i primeniti algoritam:

```
model = KNeighborsClassifier(n_neighbors=50)
model.fit(train_X_umap4, train_y_umap4);
pred_y = model.predict(test_X_umap3)
```

Za određivanje preciznosti modela ponovo koristimo matricu konfuzije i njen grafički prikaz:



Slika 14: Matrica konfuzije za UMAP i SMOTE tehniku

Kao i do sad, kao mere kvaliteta u cilju procene performansi našeg modela koristimo funkcije evaluacije *F1 Score*, *AUC ROC* i *Precision*. Njihovom primenom dobijamo sledeće rezultate:

```
mere_kvaliteta(test_y_umap3, pred_y)
```

F1 Score : 0.48

AUC ROC : 0.68

Precision : 0.08

Primećujemo da su rezultati koje smo dobili za ovaj model za nijansu bolji od rezultata modela čija baza ima dimenziju 2, što je bilo i očekivano.

Valja napomenuti da se na skoro identičan način kao UMAP primenjuje i t-SNE algoritam, koji ovom prilikom nećemo detaljnije obraditi.

2.3.3 Kombinacija PCA-a i UMAP-a

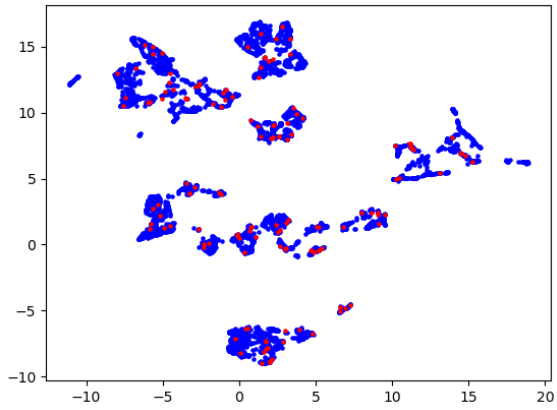
Kao što smo već istakli, prilikom obrade podataka možemo kombinovati i različite metode smanjivanja dimenzije. Na realnim podacima iz naše baze upotrebićemo PCA, a zatim UMAP, te posmatrati podatke dobijene kombinacijom ovih metoda.

Treba istaći da ovaj pristup može biti koristan jer PCA smanjuje dimenzionalnost podataka, a UMAP dodatno smanjuje dimenzionalnost i čuva lokalne strukture podataka.

Za početak, upotrebljavamo PCA i UMAP, te delimo dobijene podatke na test i trening skupove na sledeći način:

```
pca = PCA(n_components=0.99).fit(X)
X_pca = pca.fit_transform(X)
umap_ = umap.UMAP(n_components=2)
X_umap5 = umap_.fit_transform(X_pca)
train_X_umap5, test_X_umap5, train_y_umap5, test_y_umap5 = train_test_split(
    X_umap5,y,test_size=0.3, random_state=42 )
```

Kako je dimenzija nove baze 2, podatke dobijene korišćenjem ove dve metode možemo prikazati u dvodimenzionalnom sistemu pomoću sledećeg grafika:

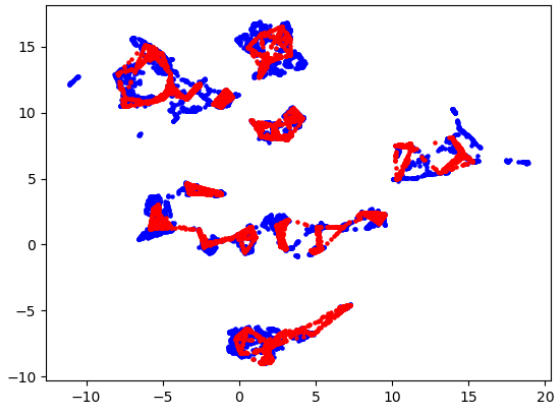


Slika 15: Grafik podataka korišćenjem PCA i UMAP algoritma

Ponovo na grafiku ne uočavamo jasnu rasdvojenost podataka, te proširujemo trening skup koristeći SMOTE tehniku:

```
smote = SMOTE(random_state=42)
train_X_umap6, train_y_umap6 = smote.fit_resample(train_X_umap5,
    train_y_umap5)
```

Nakon primene ove tehnike, ponavljamo postupak grafičkog prikazivanja dobijenih podataka i dobijamo:

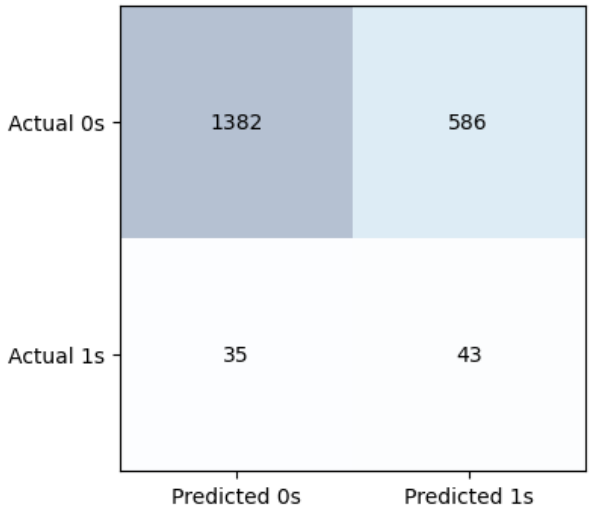


Slika 16: Grafik podataka korišćenjem PCA, UMAP i SMOTE tehnike

Vidimo da i dalje ne postoji jasna razdvojenost između kompanija, ali možemo primetiti da na određenim mestima postoje mali klasteri tačaka iste boje. Ponavljamo postupak odozgo i na novodobijene podatke primenjujemo *kNN* algoritam. Kako klasteri koje smo uočili nisu veliki, za vrednost broja k uzećemo $k = 50$ i primeniti algoritam:

```
model = KNeighborsClassifier(n_neighbors=50)
model.fit(train_X_umap6, train_y_umap6);
pred_y = model.predict(test_X_umap5)
```

Za određivanje preciznosti modela ponovo koristimo matricu konfuzije i njen grafički prikaz:



Slika 17: Matrica konfuzije za PCA, UMAP i SMOTE tehniku

Kao i do sad, kao mere kvaliteta u cilju procene performansi našeg modela koristimo funkcije evaluacije *F1 Score*, *AUC ROC* i *Precision*. Njihovom primenom dobijamo sledeće rezultate:

```
mere_kvaliteta(test_y_umap3, pred_y)
```

F1 Score : 0.47

AUC ROC : 0.63

Precision : 0.07

Matrica konfuzije ovog modela ne pokazuje značajno poboljšanje u odnosu na prethodne tehnike koje smo primenili. Takođe, vrednosti funkcija evaluacije su relativno slične ostalim tehnikama koje smo koristili.

Na osnovu ovih rezultata možemo zaključiti da primenjene metode za obradu podataka iz odabrane baze imaju određeni uticaj na performanse modela, ali ne pružaju značajno poboljšanje u predikcijama.Nakon svih korekcija modela, možemo konstatovati da rezultati ipak ne garantuju visoku pouzdanost modela.

3 Metode Klasifikacije

Nadgledano mašinsko učenje može se podeliti na algoritme klasifikacije i regresije.

Regresioni modeli se koriste kada je zavisna promenljiva neprekidnog tipa (npr. želimo da previdimo broj prodatih proizvoda), dok se klasifikacioni koriste kada ja zavisna promenljiva kategoričkog tipa. Njihov cilj je da algoritam nauči da automatski prepozna određene obrasce u podacima koji ukazuju na pripadnost određenoj klasi. Na taj način, klasifikacija može pomoći u rešavanju problema kao što su predviđanje ponašanja kupaca, klasifikacija medicinskih dijagnoza, prepoznavanje slika i drugo.

Među najpoznatije metode klasifikacije spadaju logistička regresija, *Random Forest*, SVM (*Support Vector Machine*), te algoritam k najbližih suseda (kNN) i naivni Bajes. U okviru rada smo se već fokusirali na kNN algoritam, a logističku regresiju smo takođe već primenili kod PCA metode. Pored toga, smatramo da zbog prirode podataka logistička regresija ne bi dala dobre rezultate, pa ćemo se nadalje fokusirati na ostale metode.

3.1 Support Vector Machine (SVM)

Metoda potpornih vektora (eng. Support Vector Machine - SVM) predstavlja metodu nadgledanog mašinskog učenja koja se može koristiti i za regresiju, ali je mnogo zastupljenija kod problema klasifikacije.

Cilj SVM metode je da pronađe optimalnu hiperravan u N-dimenzionom prostoru (N predstavlja broj atributa) koja jasno razdvaja podatke različitih klasa (hiperravan je potprostor čija je dimenzija za jedan manja od njegovog originalnog prostora, na primer hiperravan dvodimenzionalnog prostora je prava, a hiperravan trodimenzionalnog prostora je ravan). Ukoliko podaci nisu razdvojivi, problem se rešava tako što se podaci transformišu u prostor veće dimenzije u kojoj postaju razdvojivi. Ovaj postupak poznat je kao "kernel trick", a jezgrom odnosno kernelom, se naziva funkcija koja se koristi za ovu transformaciju. Neki od uobičajenih kernela u SVM-u su:

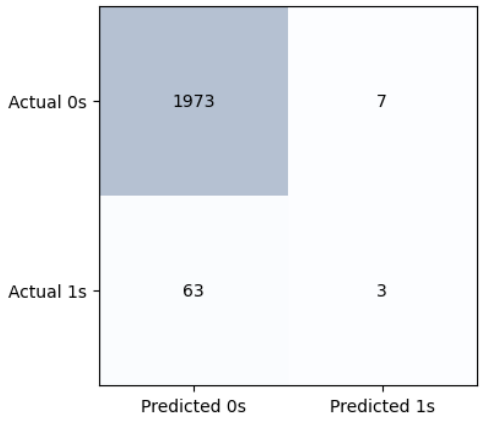
- *Gausovo jezgro*
$$k(x_i, x_j) = e^{-\gamma \|x_i - x_j\|^2}$$
- *Gausovo radijalno jezgro (RBF)*
$$k(x_i, x_j) = e^{\frac{\|x_i - x_j\|^2}{2\sigma^2}}$$
- *Polinomijalno jezgro*
$$k(x_i, x_j) = (x_i^T x_j + 1)^d$$
- *Sigmoidno jezgro*
$$k(x_i, x_j) = tanh(\alpha x_i^T x_j + c)$$

Nakon što smo pronašli optimalnu hiperravan, SVM može klasifikovati nove, neoznačene podatke tako što će ih projektovati na hiperravansku površ i odrediti kojoj klasi pripadaju.

Pokazaćemo primenu SVM metode koristeći Gausovo radijalno jezgro sa hiperparametrom $C = 100$ koji reguliše jačinu regularizacije na sledeći način:

```
svc = svm.SVC(C=100, kernel = 'rbf').fit(train_X, train_y)
pred_y = svc.predict(test_X)
```

Posmatramo matricu konfuzije ovog modela i njenu grafičku reprezentaciju:



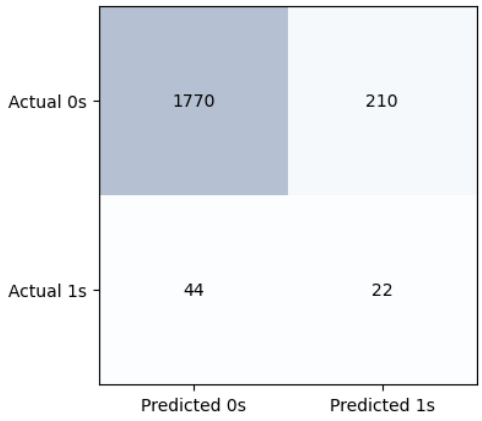
Slika 18: Matrica konfuzije za SVM metodu

Primećujemo da da matrica konfuzije ima veliki broj grešaka prve vrste, što smo istakli kao veliku manu svakog modela.

Primenićemo sada SVM metodu na podatke koji su prethodno prošli SMOTE tehniku obrade, koristeći funkciju:

```
svc = svm.SVC(C=100, kernel = 'rbf').fit(train_X_smote, train_y_smote)
pred_y = svc.predict(test_X)
```

Posmatramo matricu konfuzije ovog modela i njenu grafičku reprezentaciju:



Slika 19: Matrica konfuzije za SVM metodu i SMOTE tehniku

Kao i do sad, kao mere kvaliteta u cilju procene performansi SVM modela u kombinaciji sa SMOTE tehnikom koristimo funkcije evaluacije *F1 Score*, *AUC ROC* i *Precision*. Njihovom primenom dobijamo sledeće rezultate:

```
mere_kvaliteta(test_y, pred_y)
```

F1 Score : 0.54

AUC ROC : 0.61

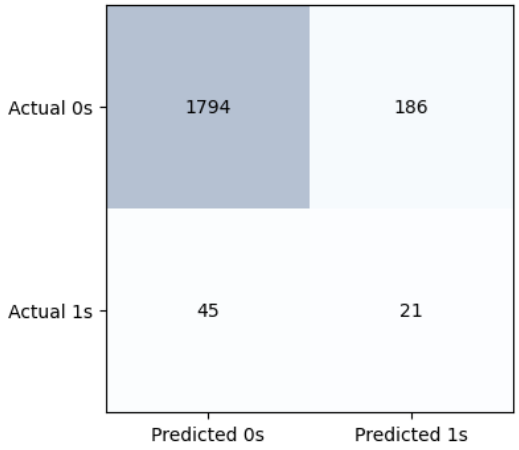
Precision : 0.09

Primećujemo da su rezultati primetno bolji, tj. da se broj grešaka druge vrste povećao primenom SMOTE tehnike. Na osnovu vrednosti funkcija evaluacije zaključujemo da model ima određeni nivo efikasnosti, ali predikcije koje pravi ne mogu biti pouzdane.

Posmatrajmo sada primenu SVM metode na podacima na kojima su prethodno izvršeni PCA i SMOTE tehnika:

```
svc = svm.SVC(C=100, kernel = 'rbf').fit(train_X_pca2, train_y_pca2)
pred_y = svc.predict(test_X_pca)
```

Posmatramo matricu konfuzije ovog modela i njenu grafičku reprezentaciju:



Slika 20: Matrica konfuzije za SVM metodu, PCA i SMOTE tehniku

Kao i do sad, kao mere kvaliteta u cilju procene performansi SVM modela u kombinaciji sa PCA i SMOTE tehnikom koristimo funkcije evaluacije *F1 Score*, *AUC ROC* i *Precision*. Njihovom primenom dobijamo sledeće rezultate:

```
mere_kvaliteta(test_y, pred_y)
```

F1 Score : 0.55

AUC ROC : 0.61

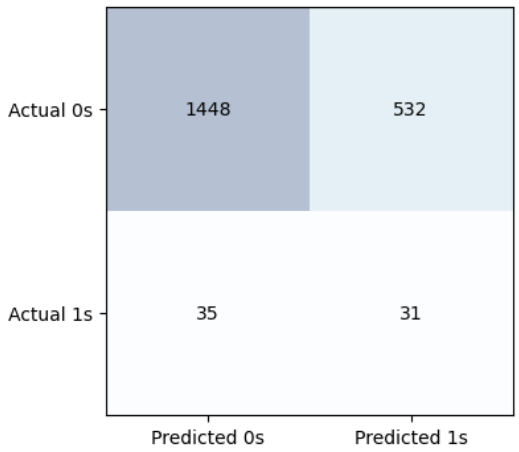
Precision : 0.10

Na osnovu metrice konfuzije i već navedenih mera kvaliteta primećujemo da je razlika u kvalitetu modela zanemarljivo mala u odnosu na kvalitet SVM modela u kombinaciji sa SMOTE tehnikom. Ipak, obe metode daju bolje rezultate od metoda primenjenih na neuravnoteženim podacima.

Razmotrićemo kako izbor jezgra utiče na preciznost modela. Ponovićemo SVM metodu u kombinaciji sa SMOTE tehnikom, koristeći polinomijano jezgro na sledeći način:

```
svc = svm.SVC(C=100, kernel = 'poly').fit(train_X_smote, train_y_smote)
pred_y = svc.predict(test_X)
```

Posmatramo matricu konfuzije ovog modela i njenu grafičku reprezentaciju:



Slika 21: Matrica konfuzije za SVM metodu i SMOTE tehniku

Kao i do sad, kao mere kvaliteta u cilju procene performansi SVM modela u kombinaciji SMOTE tehnikom koristimo funkcije evaluacije *F1 Score*, *AUC ROC* i *Precision*. Njihovom primenom dobijamo sledeće rezultate:

```
mere_kvaliteta(test_y, pred_y)
```

F1 Score : 0.47

AUC ROC : 0.60

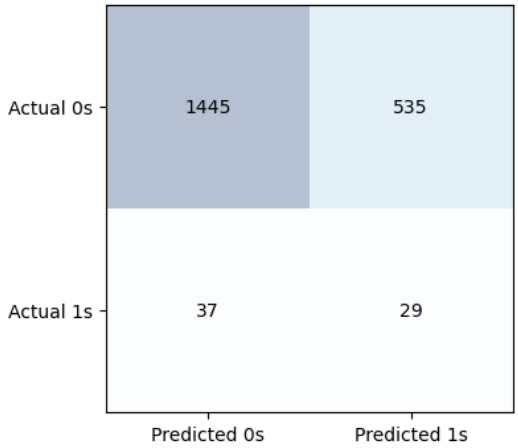
Precision : 0.06

Matrica konfuzije SVM metode u kombinaciji sa SMOTE tehnikom korišćenjem polinomijalnog jezgra razlikuje se u odnosu matrice modela koji koriste drugačiji izbor jezgra. Međutim, vrednosti funkcije evaluacije ovog modela se skoro neprimetno razlikuju u odnosu na prethodne.

Posmatrajmo sada primenu SVM metode na podacima na kojima su prethodno izvršeni PCA i SMOTE tehnika:

```
svc = svm.SVC(C=100, kernel = 'poly').fit(train_X_pca2, train_y_pca2)
pred_y = svc.predict(test_X_pca)
```

Posmatramo matricu konfuzije ovog modela i njenu grafičku reprezentaciju:



Slika 22: Matrica konfuzije za SVM sa polinomijalnim jezgrom

Kao i do sad, kao mere kvaliteta u cilju procene performansi SVM modela u kombinaciji sa PCA i SMOTE tehnikom koristimo funkcije evaluacije *F1 Score*, *AUC ROC* i *Precision*. Njihovom primenom dobijamo sledeće rezultate:

```
mere_kvaliteta(test_y, pred_y)
```

F1 Score : 0.46

AUC ROC : 0.58

Precision : 0.05

Na osnovu metrice konfuzije i navedenih mera kvaliteta primećujemo da je razlika u kvalitetu modela nastalog upotrebom SVM metode u kombinaciji sa PCA i SMOTE tehnikom korišćenjem polinomijalnog jezgra mala u odnosu na modele koji koriste drugačiji izbor jezgra.

3.2 XGBoost - Algoritam ektremnog gradijentnog pojačavanja

Algoritam ekstremnog gradijentnog pojačavanja, tj. *Extreme Gradient Boosting* je jedan od najefikasnijih i najpreciznijih algoritama za rešavanje problema klasifikacije i regresije.

Ovaj algoritam ima sposobnost da kombinuje veliki broj slabih modela, često predstavljenih u formi stabla odlučivanja, a zatim stvori jak i robustan model. Zasniva se na iterativnom procesu treniranja koji omogućava svakom stablu da se fokusira na greške prethodno izgrađenih stabala, čime se postepebo poboljšavaju performanse modela.

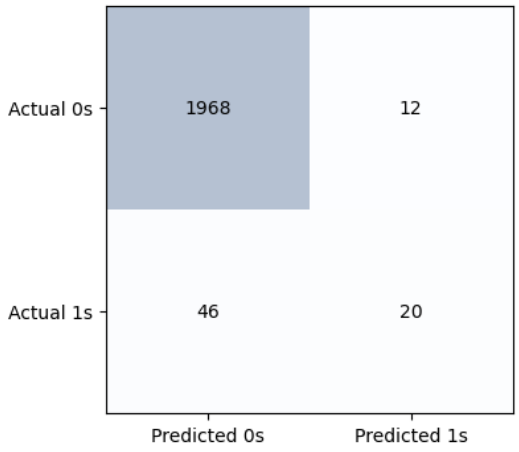
Pored gorenavedenog, XGBoost algoritam ima ugrađenu podršku za paralelizaciju, što omogućava istovremeno izvršavanje računskih operacija na više procesora ili računarskih čvorova. Ova performansa smanjuje vreme potrebno za treniranje modela i poboljšava efikasnost prilikom obrade velikog broja podataka.

Takođe, algoritam daje informaciju o važnosti prediktora u modelu, te je lakše identifikotivati ključne faktore koji utiču na rezultate modela. XGBoost uključuje i različite tehnike regularizacije, te se lako prilagođava različitim problemima i skupovima podataka.

Primenićemo sada XGBoost algoritam na realne podatke iz naše baze na sledeći način:

```
xgb = xgboost.XGBClassifier(n_estimators=100, max_depth=5)
xgb.fit(train_X, train_y)
pred_y = xgb.predict(test_X)
```

Posmatramo matricu konfuzije ovog modela i njenu grafičku reprezentaciju:



Slika 23: Matrica konfuzije za XGBoost metodu

Donosimo iste zaključke kao prilikom analize SVM modela primenjenog na neuravnoteženim podacima.

Kao i do sad, kao mere kvaliteta u cilju procene performansi XGBOOST modela koristimo funkcije evaluacije *F1 Score*, *AUC ROC* i *Precision*. Njihovom primenom dobijamo sledeće rezultate:

```
mere_kvaliteta(test_y, pred_y)
```

F1 Score : 0.70

AUC ROC : 0.65

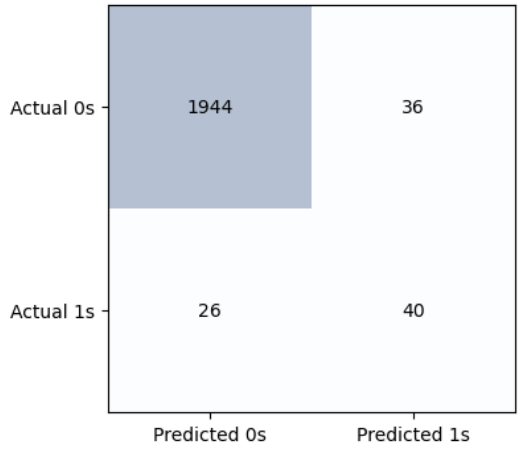
Precision : 0.62

Vidimo da su vrednosti funkcija evaluacije puno bolje u odnosu na prethodne. Ovo ukazuje da model nastao korišćenjem XGBoost metode ima primetno bolje performanse u odnosu na modele nastale korišćenjem ostalih metoda.

Razmotrimo sada implementaciju XGBoost metode na podacima na kojima je prethodno izvršena SMOTE tehnika:

```
xgb = xgboost.XGBClassifier(n_estimators=100, max_depth=5)
xgb.fit(train_X_smote, train_y_smote)
pred_y = xgb.predict(test_X)
```

Posmatramo matricu konfuzije ovog modela i njenu grafičku reprezentaciju:



Slika 24: Matrica konfuzije za XGBoost metodu i SMOTE tehniku

Kao i do sad, kao mere kvaliteta u cilju procene performansi SVM modela u kombinaciji sa PCA i SMOTE tehnikom koristimo funkcije evaluacije *F1 Score*, *AUC ROC* i *Precision*. Njihovom primenom dobijamo sledeće rezultate:

```
mere_kvaliteta(test_y, pred_y)
```

F1 Score : 0.77

AUC ROC : 0.79

Precision : 0.53

Vrednosti funkcija evaluacije ovog modela su malo bolje u odnosu na rezultate modela napravljenog na osnovu neuravnoteženih podataka. Ipak, preciznost predikcije modela napravljenom na podacima korišćenjem SMOTE tehnike je manja od preciznosti modela napravljenog na osnovu neuravnoteženih podataka.

3.3 Random Forest - Algoritam slučajne šume

Slučajna šuma, tj. *Random Forest* je važan algoritam mašinskog učenja koji se takođe koristi za probleme klasifikacije i regresije.

Ovaj algoritam kombinuje više stabala odlučivanja u cilju postizanja boljeg rezultata. Naime, on generiše više stabala odlučivanja, pri čemu svako stablo radi sa nasumično izabranim podskupom podataka i prediktora. Valja napomenuti da je, zbog ovakvog načina implementacije, algoritam otporan na *overfitting*, tj. preteranu prilagođenost modela. Kombinovanjem rezultata više stabala odlučivanja, Random forest uspeva da smanji disperziju i poveća stabilnost predikcija.

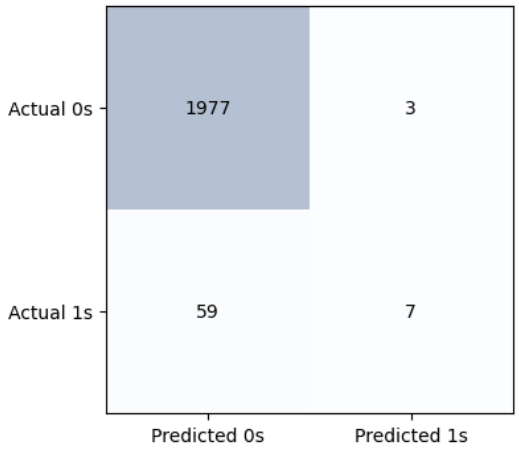
Random forest ima sposobnost rada sa podacima visoke dimenzije i velikim brojem prediktora. Kao i XGBoost, može proceniti važnost prediktora u okviru modela. Dakle, na osnovu odluka o razdvajanju podataka koje stabla donose, algoritam može izračunati važnost svakog prediktora i identifikovati one koji najviše doprinose predikciji.

Na nivou ansambla se može zadati broj stabala koja se treniraju (*n_ estimators* parametar), kao i svojstva koja prate stabla (kriterijum homogenosti, maksimalna dubina stabla, maksimalni broj atributa itd). Svojstvom *max_ depth* se može uticati na veličinu podskupa instanci nad kojim se stabla treniraju, dok je praksa da se kroz *random_ state* parametar prati slučajnost u podelama kako bi eksperimenti mogli da se reprodukuju.

Razmotrimo prvo kakve nam rezultate daje ovaj model kada su podaci neuravnoteženi.

```
model_forest = ensemble.RandomForestClassifier(n_estimators=20, max_depth=3,
    random_state=7)
model_forest.fit(train_X, train_y)
pred_y = model_forest.predict(test_X)
```

Posmatramo matricu konfuzije ovog modela i njenu grafičku reprezentaciju:

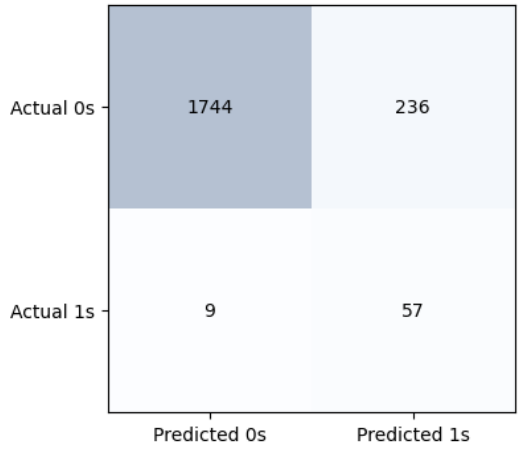


Slika 25: Matrica konfuzije za Random Forest metodu

Ovi rezultati nisu zadovoljavajući jer je broj grešaka prve vrste znatno veći od broja grešaka druge vrste. U cilju poboljšanja našeg modela, pogledajmo šta će se desiti ako Random forest metod primenimo na podatke prethodne izbalansirane pomoću SMOTE tehnike.

```
model_forest = ensemble.RandomForestClassifier(n_estimators=20, max_depth=3,
    random_state=7)
model_forest.fit(train_X_smote, train_y_smote)
pred_y = model_forest.predict(test_X)
```

Posmatramo matricu konfuzije ovog modela i njenu grafičku reprezentaciju:



Slika 26: Matrica konfuzije za Random Forest metodu u kombinaciji sa SMOTE tehnikom

Vidimo da smo sada broj grešaka prvog tipa sveli na minimum i time značajno poboljšali preciznost našeg modela. Razmotrimo sada još i sledeće mere kvaliteta: funkcije evaluacije *F1 Score*, *AUC ROC* i *Precision*:

```
1 mere_kvaliteta(test_y, pred_y)
```

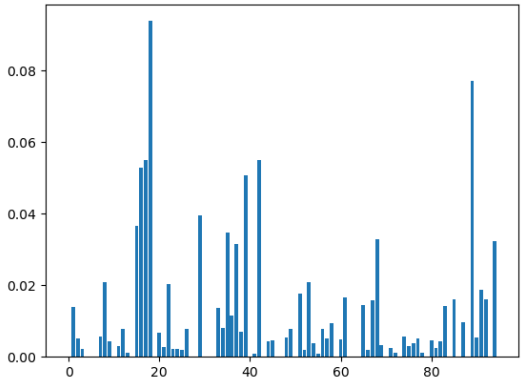
F1 Score : 0.63

AUC ROC : 0.87

Precision : 0.19

Vrednosti funkcije evaluacije dobijene ovim metodom su jako dobre, te možemo reći da nam je ovaj metod pružio dosad najveću preciznost i najmanju grešku prve vrste.

Random forest algoritam tokom treniranja modela, može proceniti važnost svakog atributa na osnovu toga koliko doprinosi smanjenju nečistoće (*impurity*) ili gubitka informacija u modelu. Pošto su rezultati koje Random forest metod daje jako značajni, pogledajmo procenu važnosti atributa koju nam on daje. Na osnovu grafika možemo videti da za najvažnije atributie smatra *Net Value Per Share (C)* i *Gross Profit to Sales*, dok neke atribute smatra potpuno nevažnim.



Slika 27: Procena važnosti atributa za random forest model

4 Zaključak

U okviru ovog projekta imali smo prilike da pokažemo kako se mašinsko učenje i njegovi algoritmi mogu primeniti za predviđanje bankrota kompanije mnogo pre nego što se on desi.

Prilikom analize podataka upoznali smo se sa važnim konceptima mašinskog učenja poput redukcije dimenzionalnosti, izbora modela, evaluacije performansi i slično. U toku izrade projekta upoređivali smo rezultate dobijene različitim metodama, grafički ih prikazivali i donosili zaključke.

Na osnovu dobijenih rezultata možemo zaključiti da metode koje smo obradili imaju određeni uticaj na performanse modela predikcije bankrota. Međutim, primena većine metoda ne dovodi do značajnog poboljšanja u predikcijama. Kompleksnost obrane baze koja se sastoji od neuravnoteženih podataka i ostali faktori implicirali su malu pouzdanost modela predviđanja.

Ipak, možemo konstatovati da smo najbolje modele dobili primenom Random forest modela na podatke izbalansirane korišćenjem SMOTE tehnike.

Važno je napomenuti da je baza koju smo koristili,kao i podaci koji su u njoj nalaze, usko vezana za prilike na tajvanskom berzi i da su tako dobijeni rezultati neprimenljivi za predviđanje bankrota kompanija u drugim državama. Da bismo uspešno predvideli poslovanje kompanije, bilo bi neophodno da skupimo dovoljno informacija o kompanijama iz iste države ili o kompanija koje posluju pod sličnim uslovima, te da nad tim podacima pravimo model koji će biti uspešan u prognoziranju.

Literatura

[1] Mladen Nikolić, Anđelka Zečević (2019). *Mašinsko učenje*. Matematički fakultet., 1st ed.

[2] Bojana Milošević (2019). *Linearni statistički modeli*. Matematički fakultet., 1st ed.

[3] Mihajlo Srbakoski (2023). Vežbe iz predmeta: *Statistički softver 4*. Matematički fakultet.

[4] Saurabh Raj (2023). *Company Bankruptcy Prediction*.

[5] Vishal Karmalkar, Shalin Shah, Akshay Rajhans (2022). *Bankruptcy Prediction on Real World Dataset using Machine Learning Algorithms*.

[6] Baza podataka korišćena za izradu projekta: *Company Bankruptcy Prediction*.

.