ANALIZA VELIKIH PODATAKA

školska 2024/2025 godina

Vežba 9: Logistička regresija i binarna klasifikacija

Šta je klasifikacija?

Klasifikacija je vrsta problema u kom model pokušava da **odredi kojoj klasi pripada neka instanca**. Umesto da predviđa brojčanu (kontinuiranu) vrednost, model daje procenu kategorije kojoj podatak pripada.

Primeri:

- Da li će korisnik otkazati uslugu (da/ne)
- Da li je email spam (da/ne)
- Da li je recenzija pozitivna ili negativna

Na ovom kursu radimo **binarne klasifikacione zadatke** – gde imamo **dve klase** (npr. 0 i 1).

Osnove logističke regresije

Za razliku od linearne regresije, koja predviđa realnu (kontinuiranu) vrednost, logistička regresija predviđa verovatnoću da neka instanca pripada klasi 1. Model koristi **sigmoidnu funkciju** da transformiše linearnu kombinaciju ulaznih promenljivih u verovatnoću.

Matematički zapis:

$$\hat{p} = \frac{1}{1 + e^{-z}}, \quad \text{gde je } z = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_n x_n$$

Gde je:

- $\hat{p}$ – procenjena verovatnoća da primer pripada klasi 1
- z – linearna kombinacija parametara i ulaznih promenljivih
- β_0 – intercept (slobodan član)
- $\beta_1, \beta_2, \ldots, \beta_n$ – koeficijenti modela
- $x_1, x_2, \ldots, x_n$ – ulazne promenljive (feature-i)

Odluka o klasi

Ako je predikcija verovatnoće veća od 0.5, model klasifikuje primer kao klasu 1, u suprotnom kao 0. Ova prag-vrednost (threshold) od 0.5 je podrazumevana, ali se u praksi može prilagoditi u zavisnosti od problema (npr. u medicini se često koristi niži prag kako bi se smanjio broj promašenih pozitivnih slučajeva).

klasa = {
  1, ako $\hat{p} > 0.5$
  0, ako $\hat{p} \leq 0.5$
}

Pretpostavke i karakteristike modela

Pretpostavka	Opis
Nezavisnost ulaznih promenljivih	Nema multikolinearnosti.
Veza između ulaznih podataka i logit funkcije je linearna	Ne između X i Y direktno, već između X i log-odnosa verovatnoća.
Nema autokorelacije	Posebno važno kod vremenskih podataka.

Log-odds i verovatnoće

Logistička regresija modeluje **log-odds** (logaritam odnosa verovatnoće da je nešto klasa 1 naspram klase 0):

$$\log\left(\frac{p}{1-p}\right) = \beta_0 + \beta_1 x_1 + \cdots + \beta_n x_n$$

To omogućuje **interpretaciju koeficijenata**: svaki β_i pokazuje **koliko se povećava log-odds** kada se x_i poveća za jednu jedinicu. Ako je β_i pozitivan, veće vrednosti x_i povećavaju verovatnoću klase 1, i obrnuto.

Gde je:

- p – verovatnoća da instanca pripada klasi 1
- $\frac{p}{1-p}$ – *odds* (odnos verovatnoće da je nešto klasa 1 prema klasi 0)
- $log(\frac{p}{1-p})$ – *log-odds* (logaritam odnosa verovatnoća), tj. logit funkcija
- β_0 – slobodan član (intercept)
- $\beta_1,\beta_2,\ldots,\beta_n$ – koeficijenti modela
- $x_1,x_2,\ldots,x_n$ – ulazne promenljive (features)

Intuicija iza sigmoid funkcije

Logistička regresija koristi **sigmoid (ili logističku) funkciju**, koja pretvara bilo koji realan broj u interval (0, 1). Zamislimo to kao "meki prelaz" između klasa. Na primer:

- Ako je izlaz blizu 0.9, model je prilično siguran da podatak pripada klasi 1.
- Ako je blizu 0.1, model smatra da je to gotovo sigurno klasa 0.
- Ako je rezultat blizu 0.5 – model je "nesiguran" i klasifikacija može biti pogrešna.

Šta ako su klase neuravnotežene?

U mnogim realnim problemima, broj primera iz jedne klase može biti znatno veći od druge. Na primer:

- 95% korisnika **nije** otkazalo uslugu,
- 5% jeste.

Model koji uvek predviđa da korisnik neće otkazati uslugu ima **tačnost od 95%**, ali **ne detektuje korisnike koji jesu otkazali**, što je najvažnije za poslovnu odluku. Takav model je praktično beskoristan u ovom kontekstu.

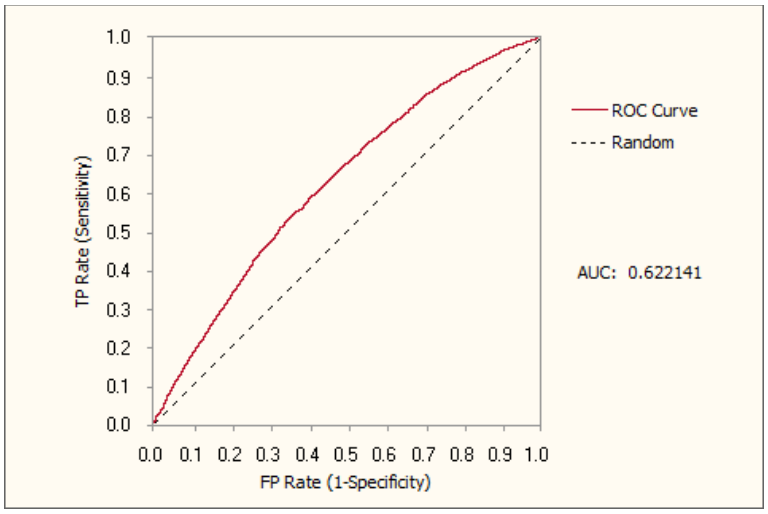
Zato je važno koristiti **dodatne metrike** koje bolje opisuju ponašanje modela kod neuravnoteženih klasa:

1. **Precision (preciznost)**
   Procenat tačno predviđenih pozitivnih primera među svima koje je model označio kao pozitivne.
   Precision = True Positives / (True Positives + False Positives)

2. **Recall (odziv ili osetljivost)**
   Procenat stvarnih pozitivnih primera koje je model uspešno identifikovao.
   Recall = True Positives / (True Positives + False Negatives)

3. **F1-score**
   Harmonijska sredina između precision i recall. Koristi se kada je važno pronaći dobar balans između tačnosti i potpunosti.
   F1-score = 2 x (Precision * Recall / Precision + Recall)

4. **ROC AUC (Area Under the Curve)**
   Pokazuje sposobnost modela da razlikuje između klasa za sve moguće pragove (thresholds). Vrednost AUC blizu 1 označava izvrstan model. Što je ROC AUC veći, to je model bolji u razlikovanju pozitivnih i negativnih primera, bez obzira na konkretan prag.



ROC AUC – šta zapravo meri?

ROC AUC (Area Under Curve) meri koliko dobro model razlikuje pozitivne i negativne klase.

- AUC = 1 → savršen model
- AUC = 0.5 → potpuno slučajna odluka (kao bacanje novčića)
- AUC < 0.5 → lošije nego slučajno (možda su klase zamenjene)
- AUC > 0.5 → model je bolji od slučajnog, što znači da ima sposobnost da prepozna pozitivne i negativne primere bolje nego slučajni odabir. Što je vrednost AUC bliža 1, to je model precizniji u razlikovanju klasa.

ROC kriva pokazuje sve moguće kombinacije tačne pozitivne stope i lažno pozitivne stope za različite granične vrednosti verovatnoće.

Kada koristiti logističku regresiju (a kada ne)?

Pogodno:

- Kada je cilj klasifikacija u dve klase.
- Kada postoji linearna veza između ulaza i izlaza (u logit prostoru).
- Kada želite interpretabilan model.

Nepogodno:

- Za višeklasa klasifikaciju (mada postoji "multinomial logistic regression").
- Kada su odnosi između promenljivih i ciljne promenljive **nelinearni**.
- Kada su podaci jako kompleksni i zahtevaju nelinearne razgraničenja (tu je bolje koristiti SVM, Random Forest, XGBoost itd.).

Kako pravilno pripremiti podatke za logističku regresiju?

Pre nego što primenimo logističku regresiju, važno je razumeti značaj pripreme podataka. Model uspešno funkcioniše ako su ulazni podaci kvalitetni i relevantni. To uključuje:

- Čišćenje podataka i uklanjanje nedostajućih vrednosti,
- Standardizaciju ili normalizaciju numeričkih promenljivih,
- Kodiranje kategorijskih promenljivih u numeričke (npr. one-hot encoding).
- Izbor i kreiranje relevantnih osobina (features) koje najbolje opisuju problem.

Podsećanja radi, one-hot encoding se koristi za transformaciju kategorijskih promenljivih u numerički format, tako što svakoj kategoriji dodeljuje poseban binarni (0/1) vektor, omogućujući modelima da ih pravilno interpretiraju.

Praktičan primer – binarna klasifikacija (otkazivanje usluge)

Koristićemo skup podataka o korisnicima telekoma (churn.csv)

```
import pandas as pd
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, confusion_matrix,
classification_report, roc_auc_score, roc_curve
import matplotlib.pyplot as plt
import seaborn as sns

df = pd.read_csv("churn.csv")

# Pretprocesiranje
df['Churn'] = df['Churn'].map({'Yes': 1, 'No': 0})
df = df.dropna()

# One-hot encoding za kat. promenljive
df = pd.get_dummies(df, drop_first=True)

X = df.drop('Churn', axis=1)
y = df['Churn']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)

model = LogisticRegression(max_iter=1000)
model.fit(X_train, y_train)

y_pred = model.predict(X_test)
y_proba = model.predict_proba(X_test)[:, 1]
```

Evaluacija modela

Accuracy

```
acc = accuracy_score(y_test, y_pred)
print(f"Tačnost modela: {acc:.2f}")
```

Confusion Matrix

```
cm = confusion_matrix(y_test, y_pred)
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues')
plt.xlabel("Predikcija")
plt.ylabel("Stvarna klasa")
plt.title("Matrica konfuzije")
plt.show()
```

Termin	Objašnjenje
TP (True Positive)	Ispravno predviđeni pozitivni
TN (True Negative)	Ispravno predviđeni negativni
FP (False Positive)	Pogrešno klasifikovani negativni kao pozitivni
FN (False Negative)	Pogrešno klasifikovani pozitivni kao negativni

Classification Report

```
print(classification_report(y_test, y_pred))
```

Sadrži:

- Precision (tačnost za pozitivnu klasu)
- Recall (koliko stvarnih pozitivnih je uhvaćeno)
- F1-score (harmonična sredina)

ROC kriva i AUC

```
fpr, tpr, _ = roc_curve(y_test, y_proba)
plt.plot(fpr, tpr, label=f"AUC = {roc_auc_score(y_test, y_proba):.2f}")
plt.plot([0, 1], [0, 1], linestyle='--')
plt.xlabel("False Positive Rate")
plt.ylabel("True Positive Rate")
plt.title("ROC kriva")
plt.legend()
plt.show()
```

Interpretacija koeficijenata modela

```
coeffs = pd.DataFrame({
    "Promenljiva": X.columns,
    "Koeficijent": model.coef_[0]
}).sort_values(by="Koeficijent", ascending=False)

print(coeffs)
```

- Pozitivni koeficijenti → veća šansa za otkazivanje
- Negativni koeficijenti → manja šansa

Može se dodatno koristiti np.exp() da se dobije **odds ratio**.

Balansiranje klasa sa SMOTE-om

Za balansiranje klasa možemo koristiti i tehniku koja se zove **SMOTE** (*Synthetic Minority Over-sampling Technique*). Ona balansira klase tako što **veštački generiše nove primere manjinske klase**.

SMOTE ne kopira postojeće primere, već koristi **interpolaciju između njih** kako bi generisao **nove, realistične uzorke** iz manjinske klase. Na taj način pomaže modelu da nauči ravnotežu između klasa i izbegne pristrasnost ka većinskoj klasi.

Primer koda – upotreba SMOTE-a u Pythonu

```
from imblearn.over_sampling import SMOTE
from sklearn.model_selection import train_test_split

# Podela na trening i test skup
X_train, X_test, y_train, y_test = train_test_split(X, y, stratify=y,
random_state=42, test_size=0.2)

# Primena SMOTE samo na trening skupu
smote = SMOTE(random_state=42)
X_resampled, y_resampled = smote.fit_resample(X_train, y_train)

# Proverimo novu raspodelu klasa
import numpy as np
unique, counts = np.unique(y_resampled, return_counts=True)
dict(zip(unique, counts))
```

Izlaz može izgledati ovako: {0: 3200, 1: 3200}

Time smo **uravnotežili klase** i omogućili modelu da uči ravnopravno o obe.