ANALIZA VELIKIH PODATAKA

školska 2025/2026 godina

Vežba 13: XGBoost

XGBoost (Extreme Gradient Boosting) je moćan i efikasan algoritam za mašinsko učenje koji se koristi za rešavanje regresionih i klasifikacionih problema. Zasnovan je na principu *boosting* tehnike, gde se više jednostavnih modela (najčešće stabala odlučivanja) kombinuje da bi se dobio snažan prediktivni model. Svako sledeće stablo se trenira tako da ispravi greške koje je napravilo prethodno stablo, čime se postepeno poboljšava ukupna tačnost modela.

Za razliku od algoritama kao što je **Random Forest**, koji koristi *bagging* (treniranje više stabala paralelno i kombinovanje njihovih rezultata), XGBoost koristi *boosting*, što znači da modeli rade **sekvencijalno** – jedno za drugim. Ideja je da se fokus stavi na one primere koje prethodni modeli nisu dobro klasifikovali, pa se sledeći modeli treniraju sa više pažnje upravo na tim greškama.

Ova tehnika omogućava XGBoost-u da postigne visoku tačnost čak i sa relativno jednostavnim modelima, jer se greške kontinuirano smanjuju kroz više iteracija.

📌 Karakteristike XGBoost algoritma

- Radi i sa numeričkim i sa enkodiranim kategorijalnim podacima.
- Koristi gradijentni boosting – trenira niz slabih modela i kombinuje ih u jak model.
- Ugrađena L1 i L2 regularizacija za smanjenje overfittinga.
- Podržava paralelizaciju tokom treniranja.
- Brz, efikasan i skalabilan – idealan za velike skupove podataka.
- Ima napredne funkcije za obradu nedostajućih vrednosti i ponderisanje uzoraka.

🧠 Zašto koristiti XGBoost?

XGBoost je jedan od najuspešnijih modela na Kaggle takmičenjima jer:

- Pruža visoku tačnost i dobre generalizacione sposobnosti.
- Ima kontrolu nad kompleksnošću modela putem regularizacije.
- Efikasno koristi resurse – trenira brže od standardnih boosting modela.
- Dobro se ponaša i sa nelinearnim i sa interaktivnim osobinama.
- Ugrađena mogućnost ranog zaustavljanja i podešavanja težina za neuravnotežene klase.

📌 Kako XGBoost funkcioniše

1. **Inicijalna predikcija**: Model počinje sa jednostavnom predikcijom (npr. srednja vrednost).
2. **Računanje reziduala (grešaka)**: Za svaki uzorak računa se koliko je trenutna predikcija daleko od stvarne vrednosti.
3. **Treniranje novog stabla**: Novo stablo se trenira da predvidi te greške (reziduale).
4. **Ažuriranje modela**: Nove predikcije se dodaju prethodnim uz kontrolisanu brzinu učenja (*learning rate*).
5. **Ponavljanje**: Proces se ponavlja dok se model ne stabilizuje ili dostigne maksimalan broj iteracija.

🧠 Značaj osobina (Feature Importance)

Jedna od važnih prednosti XGBoost-a je mogućnost analize značaja ulaznih osobina (feature-a). Ovo pomaže da se razume koje karakteristike najviše utiču na predikcije modela, što je ključno za interpretaciju rezultata i unapređenje modela.

XGBoost nudi nekoliko načina da se oceni značaj osobina:

Gain

Gain predstavlja **prosečan doprinos određene osobine u smanjenju greške modela** svaki put kada se ta osobina koristi za deljenje podataka (split) u stablu odlučivanja.

U XGBoost-u, cilj svakog grananja (split-a) u stablu je da **poveća tačnost modela** tako što smanjuje grešku – na primer, u klasifikaciji to može biti log-loss, a u regresiji srednja kvadratna greška (MSE). Kada model odlučuje **kojom osobinom da podeli čvor**, bira onu koja će najviše doprineti smanjenju te greške. Taj doprinos nazivamo **gain**.

👉 **Viši "gain" znači da osobina više doprinosi ukupnoj tačnosti modela** i ima veći značaj.

Cover

Cover meri **koliko uzoraka (instanci)** prolazi kroz određeno grananje (split) u stablu koje koristi konkretnu osobinu. Drugim rečima, ne meri samo koliko puta je osobina korišćena za deljenje čvorova, već i **koliko je podataka obuhvaćeno tim deljenjem**.

Ako se neka osobina koristi u čvoru koji se nalazi visoko u stablu (bliže korenu), tada mnogo uzoraka prolazi kroz taj čvor i samim tim osobina ima **veći "cover"**.

Ako se osobina koristi niže u stablu (u dubini), uticaće na manji broj uzoraka i njen "cover" će biti manji.

Na primer:

- Recimo da imamo 10.000 recenzija. Ako se osobina „broj slika u recenziji“ koristi za deljenje stabla i kroz to deljenje prođe 6.000 recenzija, ta osobina će imati veći "cover" u poređenju sa osobinom „dužina recenzije“, koja se možda koristi samo za 500 recenzija.

SHAP vrednosti (SHapley Additive exPlanations)

SHAP je naprednija i preciznija metoda koja daje **količinski doprinos svake osobine u donošenju konkretne predikcije** za svaki uzorak.

📌 SHAP vrednosti dolaze iz teorije igara (Shapley vrednosti) i omogućavaju da se za **svaku pojedinačnu predikciju** vidi koliko je koja osobina „dodala“ ili „oduzela“ od izlazne vrednosti modela.

Na primer:

Ako je model predvideo da je neki gost ostavio **rizičnu recenziju**, SHAP može pokazati da su **niska ocena**, **negativna sentiment analiza** i **često pojavljivanje istog korisnika** najviše doprineli toj predikciji.

Suprotno tome, za legitimne recenzije, SHAP može pokazati da su pozitivne ocene i stabilno ponašanje korisnika „gurali“ predikciju ka sigurnoj zoni.

✅ **Prednosti SHAP vrednosti:**

- **Transparentnost** – možete videti tačno koje osobine i koliko su uticale na odluku modela.
- **Vizualizacija** – moguće je crtati grafike koje pokazuju efekte osobina na veliki broj predikcija (npr. summary plot).
- **Korisno u analizi anomalija** – možete otkriti neobične obrasce ponašanja u podacima i objasniti ih

Ukratko, dok su Gain i Cover brze metode za ocenu značaja osobina u celini, SHAP vrednosti omogućavaju mnogo dublje i pojedinačno razumevanje ponašanja modela, što je posebno važno u osetljivim aplikacijama poput detekcije prevara, medicine ili finansija.

📏 **Priprema podataka**

Za razliku od Random Forest-a, XGBoost **nije potpuno neosetljiv na skaliranje**, ali i dalje može dobro da radi bez standardizacije u mnogim slučajevima.

- Kategorijalne osobine treba enkodirati (npr. One-Hot ili Label Encoding).
- Nedostajuće vrednosti se automatski detektuju i model ih tretira posebno.
- Dobro je ukloniti visoko korelisane osobine koje mogu doprineti redundantnosti.

🎯 **Evaluacija performansi**

Za regresione zadatke se koriste iste metrike iste kao i kod Random Forest-a: **MSE**, **RMSE**, **MAE**, **R^2.**

⚙️ Parametri XGBoost modela

Neki od ključnih hiperparametara uključuju:

- n_estimators: Broj stabala (boosting rundi)
- learning_rate: Brzina učenja (obično između 0.01 i 0.3)
- max_depth: Maksimalna dubina svakog stabla
- subsample: Procenat podataka za svako stablo (kontrola overfittinga)
- colsample_bytree: Procenat osobina po stablu
- reg_alpha, reg_lambda: L1 i L2 regularizacija
- gamma: Minimalno smanjenje greške da bi došlo do grananja

🧪 Kada koristiti XGBoost?

XGBoost je idealan kada:

- Ti je potrebna visoka preciznost.
- Radiš sa srednje velikim do velikim skupovima podataka.
- Imaš kompleksne nelinearne odnose između osobina.
- Želiš model koji se može lako podešavati.
- Potrebna ti je robusnost na outliere i šum.

🧪 Praktični primer u Pythonu – XGBoost Regressor

I ovde koristimo poznati housing dataset.

```
# Učitavanje biblioteka

import pandas as pd

from sklearn.model_selection import train_test_split

from sklearn.preprocessing import OneHotEncoder

from sklearn.compose import ColumnTransformer

from sklearn.pipeline import Pipeline

from sklearn.metrics import mean_squared_error
```

```
import matplotlib.pyplot as plt

import seaborn as sns

from xgboost import XGBRegressor


# Učitavanje Boston housing podataka

data = pd.read_csv("housing.csv")


# Odvajanje osobina i ciljne promenljive

X = data.drop("medv", axis=1)

y = data["medv"]


# Definisanje kategorijalnih i numeričkih osobina

categorical_features = ['chas']

numerical_features = [col for col in X.columns if col not in

                      categorical_features]


# Preprocesor - OneHotEncoder samo za 'chas'

preprocessor = ColumnTransformer([

    ('cat', OneHotEncoder(), categorical_features)

], remainder='passthrough')


# Kreiranje pipeline-a sa XGBoost modelom

model = Pipeline([

    ('preprocessor', preprocessor),

    ('regressor', XGBRegressor(n_estimators=100, learning_rate=0.1,

                               max_depth=4, random_state=42))
```

```
])

# Podela podataka na trening i test skup
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
                                                    random_state=42)
# Treniranje modela
model.fit(X_train, y_train)

# Predikcija i evaluacija performansi
predictions = model.predict(X_test)
print("MSE:", mean_squared_error(y_test, predictions))

# Analiza značaja osobina
feature_names = list(model.named_steps['preprocessor'].transformers_[0][1].
                     get_feature_names_out()) + numerical_features
importances = model.named_steps['regressor'].feature_importances_
importance_df = pd.DataFrame({"Feature": feature_names,
                              "Importance": importances})
importance_df = importance_df.sort_values(by="Importance", ascending=False)

# Vizualizacija značaja osobina
sns.barplot(x="Importance", y="Feature", data=importance_df)
plt.title("Značaj osobina u XGBoost modelu")
plt.tight_layout()
plt.show()
```