ANALIZA VELIKIH PODATAKA

školska 2024/2025 godina

Vežba 12: Random Forest

Random Forest je **ansambl model** koji se koristi za probleme regresije i klasifikacije. Radi se o kolekciji **odluka stabala (decision trees)**, gde svako stablo daje svoju predikciju, a konačna odluka se donosi **glasanjem** (kod klasifikacije) ili **prosekom predikcija** (kod regresije).

Model koristi princip **bagging-a (bootstrap aggregating)**, gde se svako stablo trenira na nasumičnom uzorku podataka i koristi nasumični podskup osobina prilikom svakog razdvajanja čvora. Ova metoda značajno smanjuje varijansu i rizik od pretreniranosti, koji je čest kod pojedinačnih stabala.

📌 Karakteristike Random Forest algoritma

- Radi i sa **numeričkim** i sa **kategorijalnim** podacima.
- Nije osetljiv na **skaliranje** podataka (nije potrebno standardizovati podatke).
- Može da se koristi za probleme sa **mnogo osobina** i kada postoji **međuzavisnost između osobina**.
- Ima **mehanizam za procenu značaja osobina**.

🧠 Zašto koristiti Random Forest?

Random Forest je popularan zato što:

- **Smanjuje overfitting**, za razliku od pojedinačnih stabala koja su sklona preučenosti.
- **Dobro podnosi nelinearne odnose** i kompleksne interakcije između promenljivih.
- **Može da radi sa velikim brojem osobina** i ne zahteva pretpostavke o distribuciji podataka.
- **Dobro podnosi podatke sa greškama i odstupanjima.**

📌 Kako Random Forest funkcioniše

1. **Bootstrap uzorkovanje**: Za svako stablo se nasumično uzima uzorak sa ponavljanjem iz trening skupa.
2. **Treniranje stabla**: Svako stablo se trenira na svom uzorku. Međutim, prilikom svakog grananja, stablo bira najbolju osobinu **samo iz nasumično odabranog podskupa osobina**, čime se smanjuje korelacija između stabala.
3. **Agregacija predikcija**: Nakon treniranja, predikcije svih stabala se kombinuju – prosekom (regresija) ili većinskim glasanjem (klasifikacija).

🧠 Značaj osobina (Feature Importance)

Random Forest može izračunati značaj svake osobine na osnovu:

- **Impurity-based** metode (Gini ili Entropy za klasifikaciju, varijansa za regresiju): meri koliko je neka osobina smanjila nečistoću u drveću.
- **Permutation importance**: meri pad performansi modela kada se vrednosti osobine permutuju (nasumično izmešaju), što ukazuje koliko je osobina važna za tačnu predikciju.
- **SHAP vrednosti** (napredna metoda): Kvantifikuju doprinos svake osobine za pojedinačne predikcije.

Ove metode pomažu u interpretaciji modela i identifikaciji najuticajnijih osobina.

📏 Priprema podataka

Random Forest modeli **nisu osetljivi na skaliranje podataka**, jer se grananja u stablu zasnivaju na poretku vrednosti, a ne na njihovoj veličini.

Takođe, **ne zahtevaju kodiranje binarnih ili kategorijalnih osobina** uvek, jer ih mnoge implementacije podržavaju direktno. Ipak, za bolju interoperabilnost, preporučuje se korišćenje **One-Hot enkodiranja**.

One-hot enkodiranje se primenjuje kada biblioteka (kao što je scikit-learn) ne podržava rad direktno sa stringovima. Na primer, kategorija Boja sa vrednostima crvena, plava, zelena bi se pretvorila u 3 binarne kolone. Ovo pomaže modelu da razume diskretne kategorije bez nametanja veštačkog redosleda.

🎯 Evaluacija performansi

Za regresione probleme, performanse se najčešće ocenjuju pomoću metrika kao što su:

- **MSE (Mean Squared Error)**
- **RMSE (Root Mean Squared Error)**
- **MAE (Mean Absolute Error)**
- **R^2 (koeficijent determinacije)**

Za klasifikacione probleme koriste se:

- **Accuracy (tačnost)**
- **Precision, Recall, F1-score**
- **ROC-AUC skor**

⚙️ Parametri Random Forest modela

Neki od ključnih hiperparametara uključuju:

- n_estimators: Broj stabala u šumi (više → bolje performanse, ali sporije izvođenje)
- max_depth: Maksimalna dubina stabla
- max_features: Broj osobina koji se uzima u obzir pri svakom grananju
- min_samples_split i min_samples_leaf: Kontrolišu minimalni broj uzoraka za deljenje čvora i postojanje lista
- bootstrap: Da li koristiti bootstrap uzorkovanje (obično True)

🧪 Kada koristiti Random Forest?

Random Forest je odličan izbor kada:

- Postoji veliki broj osobina i nemaš sigurnost koje su relevantne
- Postoji složen odnos između ulaza i izlaza
- Želiš model koji automatski detektuje značajne osobine i toleriše šum
- Nemaš striktnu potrebu za interpretabilnošću (stabla su lako objašnjiva, ali Random Forest nije toliko transparentan)
- Potreban ti je model koji pruža visoku tačnost bez prevelike potrebe za podešavanjem parametara

🧪 Praktični primer u Pythonu

U sledećem primeru koristićemo **Random Forest Regressor** da bismo predvideli prosečnu cenu kuća (medv) koristeći *Boston housing* dataset.

📦 Boston Housing dataset

Boston Housing je klasični skup podataka u mašinskom učenju koji se koristi za regresione zadatke. Cilj je predvideti **prosečnu cenu kuća u različitim delovima Bostona** na osnovu više socio-ekonomskih i geografskih karakteristika.

Sadrži ukupno **506 primera (redova)** i **13 osobina (kolona)** koje opisuju karakteristike svake lokacije, kao i **jednu ciljnu promenljivu (medv)** koja predstavlja prosečnu cenu kuće u hiljadama dolara.

🔍 Glavne osobine uključuju:

- crim: stopa kriminala po glavi stanovnika
- zn: procenat zemljišta za stambene objekte
- indus: procenat nekomercijalnih poslovnih površina
- chas: da li se nalazi uz reku Čarls (binarno: 0 ili 1)
- nox: koncentracija azot oksida (zagađenost)
- rm: prosečan broj soba po stanu
- age: procenat objekata izgrađenih pre 1940.
- dis: udaljenost do poslovnog centra
- rad: indeks pristupa autoputevima
- tax: porez na imovinu
- ptratio: odnos učenika i nastavnika u školama
- black: demografski indikator (% afroameričkog stanovništva)
- lstat: procenat siromašnog stanovništva
- medv: **ciljna promenljiva**, srednja vrednost cene kuće (u hiljadama $)

```
# Učitavanje biblioteka

import pandas as pd

from sklearn.model_selection import train_test_split

from sklearn.ensemble import RandomForestRegressor
```

```
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

# Učitavanje podataka
data = pd.read_csv("housing.csv")

# Definisanje osobina i ciljne promenljive
X = data.drop("medv", axis=1)
y = data["medv"]

# Kategorijalne i numeričke kolone
categorical_features = ['chas']
numerical_features = [col for col in X.columns if col not in
                      categorical_features]
# Preprocesor (bez skaliranja jer Random Forest to ne zahteva)
preprocessor = ColumnTransformer([
    ('cat', OneHotEncoder(), categorical_features)
], remainder='passthrough')

# Kreiranje pipeline-a
model = Pipeline([
    ('preprocessor', preprocessor),
    ('regressor', RandomForestRegressor(n_estimators=100, random_state=42))
```

```
])

# Podela skupa
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
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
sns.barplot(x="Importance", y="Feature", data=importance_df)
plt.title("Značaj osobina u Random Forest modelu")
plt.tight_layout()
plt.show()
```