KVALITETA AUTOMOBILA I STROJNO UČENJE

Iris Trgovec,
Prirodoslovno-matematički fakultet
Matematički odsjek
2023./2024.

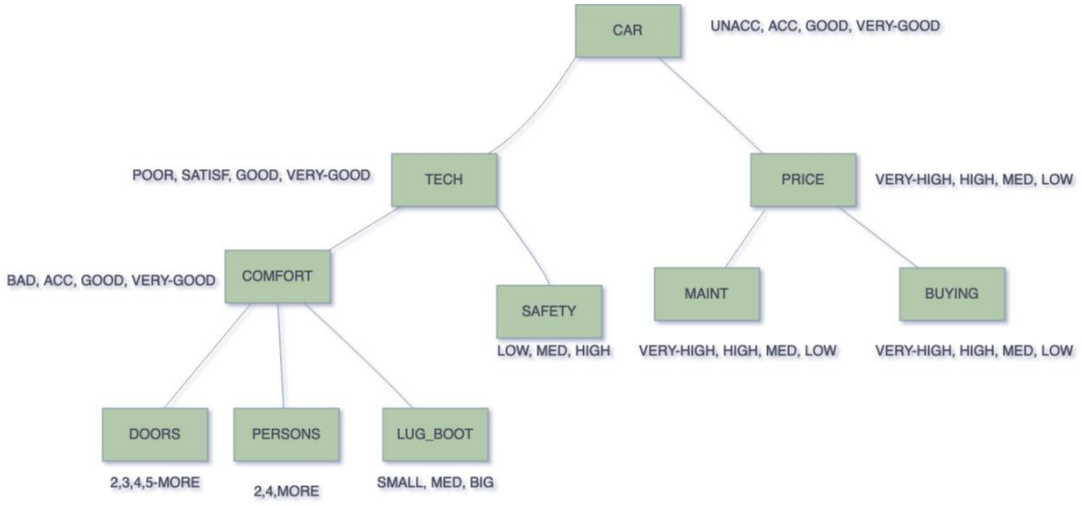
SADRŽAJ

UVOD

Razmatranje ključnih faktora prije kupovine automobila izuzetno je bitno kako biste donijeli ispravnu odluku. Jedan od najvažnijih faktora je sigurnost vozila. Proučavanje sigurnosnih aspekata pomaže odabrati automobil koji će zaštititi Vas, ali i suvozače. Bitnu ulogu imaju cijena vozila, troškovi održavanja, tehnološke karakteristika vozila…
Automobil treba biti prilagođen svakodnevnim potrebama, stoga će pažljivo razmatranje svih navedenih elemenata osigurati da će kupovina automobila biti dobro promišljena te prilagođena traženim preferencijama.

OPIS PROBLEMA I PODATAKA

U ovom radu promatrat ćemo bazu podataka pronađenu na stranici http://archive.ics.uci.edu/dataset/19/car+evaluation [1]. Baza se sastoji od podataka o 1728 automobila, pri čemu su svi podaci kategorijalni.



Slika 1 odnos između podataka

Zbog jednostavnosti i činjenice da je baza podataka na engleskom jeziku, koristit ćemo engleske oznake. Kao što je prikazano na prethodnoj slici, kvaliteta auta (unaccepted, accepted, good, very-good) ovisi o dva glavna kriterija, cijeni (price) i tehnološkim karakteristikama (tech).
Oznake za kvalitetu auta, poredane od lošije prema najboljoj:

- unacc (skraćeno od unaccepted - neprihatljiv)
- acc (skraćen od accepted - prihatljiv)
- good (dobar)
- vgood (skraćeno od very good – jako dobar)

Nadalje, cijena ovisi o prodajnoj cijeni (buying) i cijeni održavanja (maint) pri čemu su njihove vrijednosti označene s very-high (jako visoka), high (visoka), med (medium – osrednja) i low (niska).
Tehnološke karakteristike ovise o udobnosti (comfort) i sigurnosti (safety). Udobnost ovisi o dvije diskretne varijable, broju vrata (varijabla doors) i broju osoba (varijabla persons) koje stanu u auto, te o veličini prtljažnika (lug_boot) koja može poprimiti vrijednosti small, medium (med) i big.

Što se tiče faze u kojoj pripremamo podatke za daljnju analizu, ona će biti izuzetna jednostavna zbog činjenice da nemamo nedostajućih vrijednosti u bazi.

Cilj ovog projekta je uz upravo navedene kategorijalne podatke o autu odrediti hoće li automobil biti ocijenjen kao unacc (neprihatljiv), acc (prihatljiv), good (dobar) ili vgood(jako dobar). S obzirom na to da je riječ o klasifikaciji, koristit ćemo stablo odluke i slučajne šume. Iz dane baze odvojit ćemo dio podataka (20%) na kojem ćemo evaluirati točnost odabranih metoda nadziranog strojnog učenja.

STROJNO UČENJE

Strojno učenje (engl. *Machine learning*) predstavlja granu umjetne inteligencije koja se bavi razvijanjem sustava koji mogu poboljšavati svoje perfomanse pomoću iskustva. Postoji više vrsta strojnog učenja, a u ovom projektu bavit ćemo se nadziranim učenjem, točnije, klasifikacijom.
Promotrit ćemo dva različita algoritma za klasifikaciju, stablo odluke i slučajne šume. Usporedit ćemo njihovu uspješnost u predviđanju rezultata u programskom jeziku Python. Podatke ćemo, pomoću funkcije train_test_split() iz biblioteke sklearn, podijeliti na četiri grupe, X_train, Y_train, X_test i Y_test, pri čemu prve dvije grupe čine train set, a zadnje dvije test set. Također, odnos u veličini između train seta i test seta bit će 80:20. Algoritmi za klasifikaciju implementirani su u biblioteci sklearn te su jednostavni za korištenje. Svaki algoritam prvo treba inicijalizirati, pomoću .fit naredbe istrenirati model na train setu te zatim pomoću naredbe .predict predvidjeti rezultate za test set. Nakon toga ćemo korištenjem matrice zabune (confusion matrix) i funkcije accuracy_score evaluirati točnost modela na našim podacima.

Da bismo mogli koristiti navedene algoritme, prvo trebamo kodirati kategorijalne varijable u numeričke. Za to ćemo koristiti ***One-Hot Encoding***, koji će nam pomoći zaobići problem ordinalnosti koji se može pojaviti kad kategorijalne varijable imaju prirodan poredak (npr. low – med – high). One-Hot Encoding implementiran je u Python biblioteci Pandas, a poziva se pomoću sljedećeg koda:

```
one_hot_encoded_data = pandas.get_dummies(data, columns = ['buying'])
```

pri čemu paramter columns sadrži imena stupaca varijabli koje želimo kodirati.

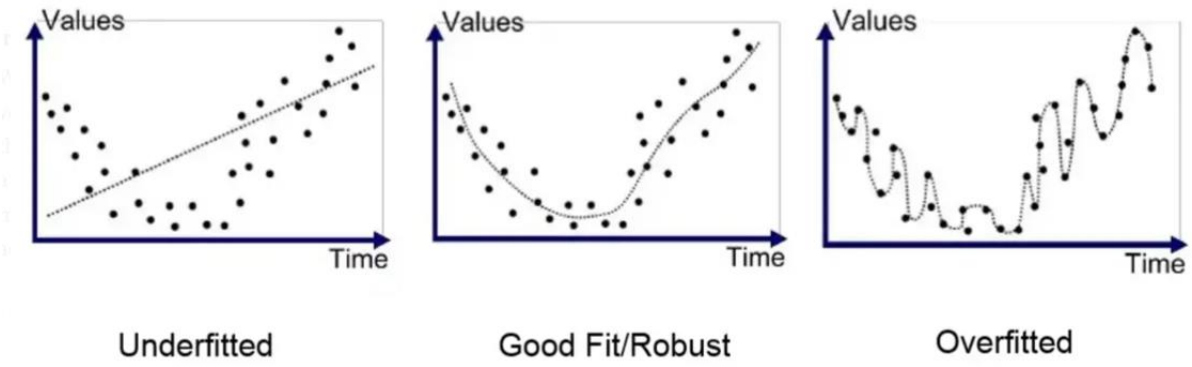
Stablo odluke (engl. *Decision tree*) algoritam je koji se koristi za rješavanje problema klasifikacije i regresije. Na temelju ulaznih podataka gradi model u obliku stabla, pri čemu svaki čvor u stablu predstavlja testiranje određenog atributa, a listovi sadrže klasifikaciju početnog podatka.

```
from sklearn.tree import DecisionTreeClassifier     #Decision tree is
already implemented in the sklearn library
classifier = DecisionTreeClassifier(criterion="entropy",random_state=0)
             #criterion is function to measure the quality of a split

classifier.fit(X_train, Y_train)  #we have to train our model on the
train set data

y_pred = classifier.predict(X_test)
```

Slika 2 isječak koda gdje se koristi stablo oduke

Slučajna šuma (engl. *Random forest*) je algoritam koji koristi više stabala odlučivanja kako bi klasificirao ulazi podatak. Algoritam na početku izdvaja slučajne podskupove podataka iz skupa za treniranje te za svaki izdvojeni podskup konstruirao stablo odlučivanja. Za konačnu odluku odabire se rezultat koji se najčešće pojavljivao. U modelu se mogu podešavati brojni parametri, poput broja stabala odluke u šumi koje želimo generirati (varijabla n_estimators), funkcije koja mjeri kvalitetu podjele (varijabla criterion). Criterion može poprimiti vrijednosti iz skupa {„gini“, „entropy“, „log_loss“}. Postoji još parametara koji se mogu podešavati, a još ćemo izdvojiti max_depth koja je po defaultu postavljena na None, a označava maksimalnu dubinu stabla.

Algoritam slučajne šume, zbog većeg broja stabala odluka koji generira, sporiji je od algoritma stabla odluka. No, najčešće daje točnije rezultate od stabla odluke jer, zbog toga što se temelji samo na nekim podacima, smanjuje mogućnost da dođe do 'overfittinga'. Naime, do 'overfittinga' dolazi kad se model previše prilagodi podaci na kojima uči pa može postati manje točan na novim, do tad neviđenim, podacima.



Slika 4 prikaz overfittinga

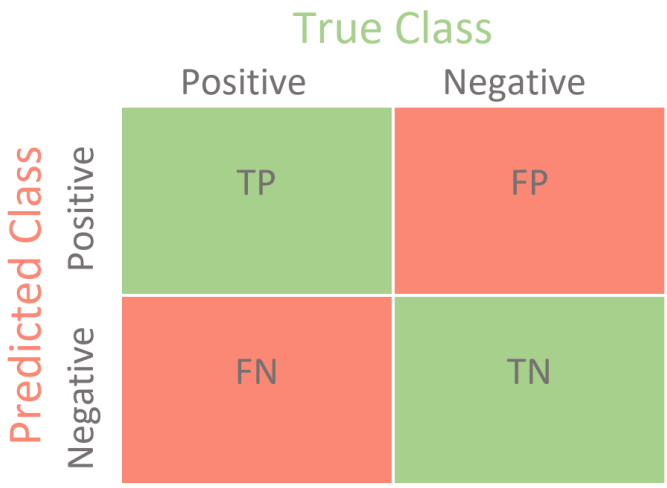
S obzirom na to da algoritam slučajne šume ima velik broj parametara koji se mogu mijenjati, naći ćemo najbolje za naše podatke pomoću funkcije GridSearchCV() te ćemo zatim s tim parametrima kreirati model slučajne šume. Taj postupak naziva se 'hyperparameter tuning', a sastoji se od pronalaženja kompromisa između točnosti i kompleksnosti modela. Ideja je pronaći parametre koji će imati visoku točnosti, ali bez pojave 'overfittinga' kod train seta podataka te ćemo na taj način dobiti model koji će se jako dobro ponašati na novim podacima.

```
from sklearn.model_selection import GridSearchCV

param_grid = {
    'n_estimators': [25, 50, 100, 150, 200, 250],
    'max_features': ['sqrt', 'log2', None],       # possible parameters
    'max_depth': [5,10,20],
    'max_leaf_nodes': [5,10,20]
}

grid_search = GridSearchCV(RandomForestClassifier(random_state=0),
                           param_grid=param_grid)      #in grid_search
are best parameters
grid_search.fit(X_train, Y_train)    #fitting them on our train set
best_param= grid_search.best_estimator_
```

Slika 4 isječak koda za pronalaženje najboljih parametara

USPOREDBA MODELA

Postavlja se pitanje kako provjeriti koji je od navedenih klasifikacijskih modela dobar za određeni problem. U tu svrhu koristit ćemo ***matricu zabune*** (engl. *confusion matrix*).



Slika 4 matrica zabune, 2x2

Vrijednosti koje su pojavljuju u matrici zabune (u slučaju kad rezultat može poprimiti samo dvije vrijednosti) su sljedeće:

- TP (engl. *true positive*) – broj instanci koje su stvarno pozitivne i klasificirane su kao pozitivne od strane modela
- FP (engl. *false positive*) –broj instanci koje su negativne, ali klasificirane kao pozitivne
- FN (engl. *false negative*) –broj instanci koje su negativne i klasificirane kao negativne
- TN (engl. *true negative*) – broj instanci koje su pozitivne i klasificirane kao negativne

Na sličan način se matrica zabune može definirati kad rezultat može poprimiti n različitih vrijednosti, pri čemu će tada dimenzija matrice biti n x n.

Točnost (engl. *accuracy*) definira se kao zbroj elemenata na glavnoj dijagonali matrice zabuna, tj. broj ispravno klasificiranih primjera podijeljen s ukupnim brojem primjera. U Pythonu se točnost modela računa pomoću funkcije accuracy_score(Y_test, Y_pred). Upravo spomenuta funkcija vraća broj između 0 i 1, a čim je broj bliže 1, to je model točnije predvidio rezultate na novim podacima.

U sljedećoj tablici prikazana je točnost za promatrane algoritme na danoj bazi podataka.

Algoritam	Accuracy_score()
Stablo odluke	0.95664
Slučajne šume (s GridSearchCV)	0.942196

Tablica 1 usporedba točnosti algoritama

ZAKLJUČAK

Iako model stabla odluke ima malo veću točnost, bolje je koristiti slučajne šume jer je to kompleksniji model te će se bolje prilagoditi na generalizirane podatke, odnosno, manje je podložan overfittingu u usporedbi sa stablom odluke.

Stablo odluke može biti dobar algoritam za jednostavne probleme, ali na složenijim skupovima podataka možda neće biti toliko efikasan. Slučajne šume preporučuju se za korištenje na velikim skupovima podataka i bazama podataka s puno stupaca.

Nadalje, zbog pretraživanja više kombinacija parametara i izgradnje većeg broja stabala, slučajne šume su sporiji algoritam od stabla odluke.

Literatura

https://www.qwak.com/post/xgboost-versus-random-forest [pristupljeno: 4.1.2024.]
https://machinelearningmastery.com/one-hot-encoding-for-categorical-data/ [pristupljeno: 4.1.2024.]
https://www.geeksforgeeks.org/ml-one-hot-encoding-of-datasets-in-python/ [pristupljeno: 4.1.2024.]
https://tahera-firdose.medium.com/fine-tuning-your-random-forest-classifier-a-guide-to-hyperparameter-tuning-d5ceab0c4852 [pristupljeno: 4.1.2024.]
https://en.wikipedia.org/wiki/Decision_tree_learning [pristupljeno: 4.1.2024.]
https://scikit-learn.org/stable/modules/tree.html [pristupljeno: 4.1.2024.]
https://www.javatpoint.com/machine-learning-random-forest-algorithm [pristupljeno: 4.1.2024.]
https://en.wikipedia.org/wiki/Random_forest [pristupljeno: 4.1.2024.]