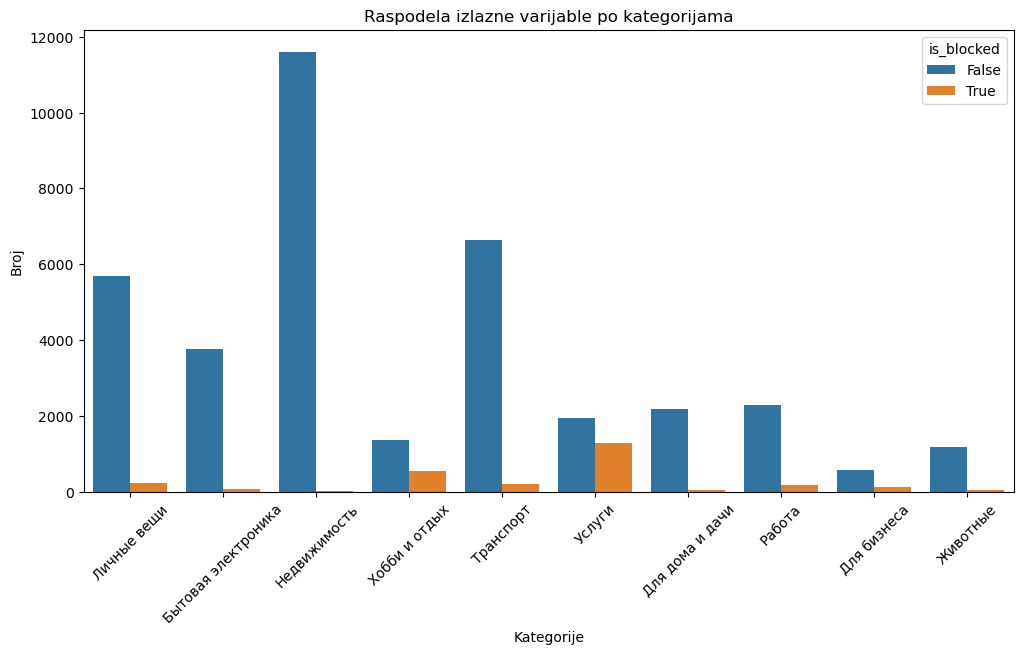
Avito.ru – „The Hunt for Prohibited content“

# Opis problema

Avito.ru, ruski je sajt koji spaja kupce sa prodavcima dobara i usluga širom Rusije. Avito sadrži više od 22 miliona oglasa i veoma je popularan, poslovanje samog sajta i kompanije u mnogome zavisi od kvalitetnog sadržaja. Zbog popularnosti sajta svakog dana se dešava da se jave ponuđači ilegalnih proizvoda i/ili usluga, kako po zakonu Rusije tako po Avito – ovim internim pravilima. Avito je stoga angažovao tim moderatora koji nastoje da blokiraju ilegalni, tj. neprimereni sadržaj. Sa velikim rastom, postaje kompleksno moderirati oglase ručno i tu mašinsko učenje stupa na scenu – potrebno je napraviti model koji će iz moderatorskih iskustava naučiti kako da klasifikuje oglas.

# Opis i razumevanje podataka

Sam problem se svodi na problem nadgledanog mašinskog učenja – binarne klasifikacije, u kome se predviđa varijabla „is\_blocked“ koja sadrži klasu opservacije (blokiran/neblokiran oglas). Ostatak skupa podataka se sastoji od tekstualnih podataka poput kategorije i podkategorije i to: 10 kategorija proizvoda i usluga sa 45 podkategorija istih. Među tekstualnim podacima su i naslov oglasa, opis oglasa kao i dodatni atributi oglasa koji su dati u JSON formatu.



Najveći broj opservacija, sa ujedno najvećim disbalansom između blokiranih i neblokiranih oglasa nalazi se u kategoriji nekretnina, međutim skoro sve instance su neblokirani oglasi – ova kategorija je kasnije loše uticala na performanse modela pa je izbačena iz krajnje analize. Iz dalje analize izbačeni su id oglasa kao i atributi „is\_proved“ i „close\_hours“, naglašeno je da su dati atributi visoko korelisani sa izlaznom promenljivom i da se ne smeju koristiti u daljoj analizi[[1]](#footnote-1).

Nedostajuće vrednosti su se javljale samo kod tekstualnih podataka i to 614 u varijabli „title“i 1136 u „attrs“. Vrednosti su zamenjene modusom kolone korišćenjem klase „SimpleImputer“ iz „scikitlearn“ biblioteke.

# Priprema podataka

Nakon popunjavanja nedostajućih vrednosti usledila je priprema podataka i to odvojeno u podskupovima numeričkih i kategoričkih (uglavnom tekstualnih) podataka.

Najveći izazov je bila veličina samog skupa podataka koji je odmah na početku smanjen slučajnim uzorkovanjem. Sledeći problem je bila vizualizacija i razumevanje nekih korelacija između kategorija i podkategorija – inicijalni plan je tekao tako što bi se skup podataka delio i posmatrao iz ugla kategorija i podkategorija, tako da su napravljene metode koje su podatke na ruskom za kategorije i podkategorije pretvarale u engleski radi vizualizacije i razumevanja. Postojao je i pokušaj da se ceo skup podataka prevede preko Google Translate API – ja, međutim ovakav pristup nije pokazao rezultate jer se sam API „bunio“ zbog mnogo brojki, abrevacija i skraćenica u ruskom jeziku a u našim tekstualnim podacima.

Kategorički atributi su dalje bili obrađivani u kontekstu kolone „attrs“ gde je pokušano da se atributi izvuku iz JSON formata i dodele posebnim klasama. Ovaj deo pripreme podataka je bio izazovan zbog neuniformnosti JSON formata i različitih atributa – kategorije malih potrepština za kuću i kategorija vozila nemaju istu strukturu JSON-a, tako da su podaci vađeni u veliki broj zasebnih kolona i u kolonama u kojima data opservacija nema vrednost ona je bila popunjavana sa praznom string vrednošću.

Ovo je naposletku, napravilo problem u količini potrebne memorije da se izvrši vektorizacija tekstualnih podataka pa se vektorizacija bazirala na manjem broju reči, međutim performanse kasnije izgrađenih modela nisu se pokazale bolje sa ovakvim prikazom kolone atributa. Kolona „attrs“ je na kraju zadržala originalno stanje i kao takva, zajedno sa kolonama naslov i opis bila podvrgnuta vektorizaciji i to **tfidf** pristupu. U samoj vektorizaciji ograničio sam se na maksimum od 600 reči po koloni što je dodalo novih 1800 kolona(varijabli) u posmatrani skup podataka.

U numeričkim atributima je ispitana korelacija – nije postojalo značajne korelacije između bilo kog para atributa, kako su atributi bili na različitim skalama i imali ekstremne vrednosti uklanjani su autlajeri iz atributa „phone\_cnt“ i „price“ metodom „winsorize“ pošto su metode „zscore“ i „boxplot“ ostale bez efekta na autlajerima kategorije price.

Numerički podaci su na samom kraju pripreme bili skalirani korišćenjem RobustScaler – a iz paketa „scikitlearn“.

# Treniranje algoritama, evaluacija, optimizacija hiperparametara i ansambl algoritmi

Pripremljeni podaci su korišćeni u četiri različita modela mašinskog učenja – stablu odlučivanja, logističkoj regresiji, K najbližih suseda i Naivnom Bajesu. Model logističke regresije sa maksimalnih 500 iteracija je dao najbolje rezultate u početku, model Naivnog Bajesa je imao najlošije performanse – mali broj podataka iz pripremljenog skupa je zadovoljavao pretpostavku normalnosti, stablo odlučivanja i K najbližih suseda bili su na sredini.

Korišćene evaluacione metrike su tačnost – radi identifikacije procenta tačno klasifikovanih opservacija, kao i ROC AUC mera radi identifikacije sposobnosti modela da kvalitetno razdvoji klase. Za evaluiranje korišćena je „kros-validacija“ sa 5 „fold“-ova.

Za samo poboljšanje performansi modela je korišćena „grid“ pretraga – u modelu K najbližih suseda ovo je dalo značajna poboljšanja u performansama sa hiperparametrom k=11. Za ostale modele korišćeni su algoritmi ansambli koji su takođe dali bolje rezultate i to „bagging“ i „boosting“ za stablo kao i za Naivnog Bajesa, gde je „boosting“ pokazao bolji učinak. Korišćen je i voting klasifikator za ova dva modela po „soft voting“ principu, rezultati su bili malo bolji.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | ROC AUC trening | ROC AUC test | Acc. trening | Acc. test |
| Logistička regresija | 0.9798 | 0.9645 | 0.9604 | 0.9516 |
| Naivni Bajes | 0.8390 | 0.8096 | 0.7130 | 0.7064 |
| Stablo odlučivanja | 0.8248 | 0.8173 | 0.9426 | 0.9405 |
| kNN | 0.9777 | 0.9380 | 0.9535 | 0.9471 |
| Bagg1(so) | 0.8704 | 0.8617 | 0.9452 | 0.9437 |
| Bagg2(nb) | 0.9079 | 0.8730 | 0.7718 | 0.7647 |
| Boost1(so) | 0.9495 | 0.9434 | 0.9625 | 0.9346 |
| Boost2(nb) | 0.9286 | 0.9181 | 0.9230 | 0.8734 |
| „Voting“ klasifikator | 0.9395 | 0.9082 | 0.7254 | 0.7173 |

# Sa više znanja i vremena…

Pisanje koda sa Pipeline – ovima je nešto na čemu moram da poradim, nisam dovoljno vešt u tome i pravilo mi je probleme pri samom kodiranju tako da sam se odlučio da ih ne koristim iako to nije nimalo standardan način pisanja koda i naravno time sam izbegao adresiranje problema curenja informacija. Kako su centralni deo podataka činili tekstualni podaci posvetio bih se više obradi i vektorizaciji tih podataka kada bih znao više o samim metodama i tehnikama, naime malo sam uradio na polju selekcije atributa za modele, verovatno da bi dobra strategija bila trenirati modele na nekim podskupovima podataka i kasnije napraviti ansambl nad istim. Takođe hteo da probam je treniranje neuronske mreže pomoću biblioteke Tensorflow, međutim i tu mi je falilo znanja.

Takođe jedna od stvari koja mi je pala na pamet je korišćenje varijabli koje su direktno povezani sa izlaznom u kreiranju neke nove izlazne varijable i samim tim prebacivanje modela iz domena binarne klasifikacije u domen problema regresije respektivno sa predviđanjem do koje mere je neki oglas neprikladan ili ilegalan.

Tim YoLo

Miloš Jolović 2020/0179

1. Ovde sam dobio ideju da se problem možda može transformisati u problem regresije gde bi ocenjivali stepen neprikladnosti oglasa, kombinovanjem ova dva atributa sa trenutnom izlaznom varijablom. [↑](#footnote-ref-1)