Mercari - Price Suggestion Challenge

Bojana Đerić, Ivan Emanuel Pavlov,
Tatjana Ramljak, Lara Rajković

19. travnja 2019.

1 Uvodni opis problema

Mercari[1] je japanska aplikacija za prodaju i kupnju stvari. Korisnici aplikacije mogu prodati bilo što, bilo korišteno ili ne: odjeću, igračke, elektronsku i sportsku opremu... Njihova misija je da prodaja stvari postane lakša od kupovine. Kako bi to postigli htjeli su da aplikacija predloži cijenu korisnicima kada stave novi predmet na prodaju.

U tu svrhu otvoreno je natjecanje na Kaggleu. Zadatak je predvidjeti cijenu predmeta s obzirom na informacije koje je korisnik unio o tom predmetu. Informacije koje su dane u *listingu* su: naslov, u kakvom je stanju predmet, kategorija, marka, je li *shipping* plaćen od strane prodavača ili nije te opis predmeta. U skupu za učenje dana je i cijena predmeta. Model ćemo razvijati na skupu za učenje te ga zatim testirati na skupu za testiranje. Oba skupa podataka su dostupna na Kaggleu.

2 Cilj i hipoteze istraživanja problema

Cilj istraživanja problema je napraviti model koji daje što manju pogrešku *(Root Mean Squared Logarithmic Error)* na skupu za testiranje. Pogreška koju minimiziramo je dana u pravilima natjecanja.

3 Materijali, metodologija i plan istraživanja

U svrhu rješavanja problema nadamo se izgraditi nekoliko različitih modela. Potom bi kreirali ansambl kojime bi htjeli poboljšati prediktivnu moć. Trenutno imamo neke ideje, ali pred nama je još puno istraživanja tako da još nemamo jasnu sliku o svim detaljima provedbe projekta pogotovo onima koji se tiču dubokog učenja.

Nakon eksploratorne analize provest ćemo *feature engineering* tj. kvalitetnu konstrukciju i odabir značajki za model strojnog učenja. Koristit ćemo neke metode iz *Scikit-learn* biblioteke (te moguće nekih drugih kao *WordBatch*) za

[1] https://www.mercari.com/

provedbu analize teksta da bi generirali *featurese* od nestrukturiranih značajki. Podaci *name* i *item_description* su tekstualni podaci koje ćemo pomoću TF-IDF metode pretvoriti u numeričke vrijednosti. *PCA* metodom ćemo smanjiti broj dobivenih varijabli. Podatke *item_condition_id*, *category_name*, *brand_name* i *shipping* tretirat ćemo kao kategorijske varijable. Pokušat ćemo napraviti redukciju broja kategorijskih varijabli pomoću *MCA (Multiple Correspondance Analysis)*. Zatim ćemo koristiti višedimenzionalnu linearnu regresiju uz normalizaciju u svrhu dinamičke predikcije cijene. Početni model koji dobijemo mijenjat ćemo i nadograđivati kako bismo što više smanjili pogrešku.

U kontekstu dubokog učenja, nakon upoznavanja s metodama istoga nadamo se probati neke pristupe za koje smo primjetili da su česti: višeslojni perceptron (MLP), konvolucijske neuronske mreže, povratne neuronske mreže (RNN) itd.

Koristit ćemo programski jezik *Python*. Neki od paketa koje ćemo koristiti su *matplotlib*, *numpy*, *pandas*, *seaborn*, *wordcloud*, *textblob*, *sklearn*, *tensorflow*, *keras*...

Mjera pogreške modela je *RMSLE (Root Mean Squared Logarithmic Error)* koja će biti izračunata na Kaggleu nakon što priložimo cijene podataka skupa za testiranje. Ona se dobiva po formuli

$$\sqrt{\frac{1}{n}\sum_{i=1}^{n}(log(p_i+1)-log(a_i+1))^2}$$

pri čemu su
n - broj podataka u skupu,
p_i - predviđene cijene,
a_i - stvarne cijene.

4 Pregled dosadašnjih istraživanja

Kako je problem preuzet s Kagglea, na stranici su ponuđena razna rješenja. Neki od korištenih algoritama su *MLP (multilayer perceptron)*, *Light Gradient Boosting*, *Convolutional Neural Networks(CNN)* , *Random forest*, *Ridge regresija*, *Embedding-Sum-MLP*.

Rješenja su uglavnom koristila razne kombinacije algoritama. Nedostatke postojećih istraživanja nam je jako teško odrediti jer se vidi da je uloženo mnogo rada u svako od njih, no pokušat ćemo poboljšati modele boljim *feature engineeringom* i redukcijom dimenzije. Pobjednički tim imao je RMSLE jednak 0.37758, dok je najveći moguć RMSLE koji su imali posljednje rangirani korisnici jednak 99.00000 što je posljedica ekstremnog overfittanja ili vjerojatnije neuspjelog izvršenja koda kod validacije rješenja koncem natjecanja.

Zanimljivo je i da je kod trebao biti izvršen uglavnom u rasponu od jednog sata, što je predstavlja očito ograničenje u pogledu mogućih rješenja. Također, natjecanje je bilo Kernel tipa, što bi značilo da se završni kod morao "odvrtiti" u Kaggle kernelu u rasponu od 1h, te nije bilo dozvoljeno "scrappanje" dataseta

s Kagglea, vjerojatno u svrhu omogućavanja ravnopravnijih uvjeta natjecanja ljudima koji nemaju pristup skupom hardwareu.

Možda bitno za navesti je i to da se Kaggle kernel vrti na relativno jakom hardwareu (CPU i GPU) s dovoljno memorije, koji je izvan kupovne moći većine korisnika Kagglea, tako da s naše strane to nama ne bi trebalo biti ograničenje.

5 Očekivani rezultati predloženog projekta

Kroz izradu projekta se nadamo isprobati metode koje su obrađene na predavanju, te produbiti trenutno znanje istih. Također se nadamo naučiti nove metode i pristupe, kao nadogradnju na postojeće ili moguće alternativu.

Kao konačni rezultat projekta očekujemo predati model koji daje relativno malu pogrešku i dobro predviđa cijenu proizvoda.

Literatura

[1] Yaser S. Abu-Mostafa, Malik Magdon-Ismail i Hsuan-Tien Lin. *Learning From Data*. 2012.

[2] Benjamin Bengfort, Tony Ojedaa i Rebecca Bilbro. *Applied Text Analysis with Python: Enabling Language-Aware Data Products with Machine Learning*. 2018.

[3] Aurélien Géron. *Hands-On Machine Learning with Scikit-Learn and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems*. 2017.

[4] Susan Li. *A Complete Exploratory Data Analysis and Visualization for Text Data*. 19. ožujka 2019. URL: `https://towardsdatascience.com/a-complete-exploratory-data-analysis-and-visualization-for-text-data-29fb1b96fb6a`.

[5] Mercari. *Mercari Price Suggestion Challenge*. 18. veljače 2018. URL: `https://www.kaggle.com/c/mercari-price-suggestion-challenge/`.

[6] Antti Puurula, Anders Topper i Cheng-Tsung Liu. *Wordbatch*. 3. siječnja 2019. URL: `https://github.com/antttti/Wordbatch`.

[7] Scikit-learn. *Feature extraction*. URL: `https://scikit-learn.org/stable/modules/feature_extraction.html`.