Petre-Antica Eduard-Dominic

Competiție SII

Detalii Implementare

GitHub link: https://github.com/Eduard-Petre/SII-Competitie

1. Preprocesarea datelor

Pentru a realiza preprocesarea datelor (de la setul de date inițial la valori numerice), am realizat un Pipeline ce cuprinde toți pașii necesari:

a. Ștergerea coloanelor nenecesare

Am scos din setul de antrenare coloanele care nu aveau relevanță pentru model : „preț”, „id” și „marca” (scoasă întrucât aceasta se afla deja în coloana „model”)

b. Transformarea coloanelor categorice

Pentru coloanele categorice ce au 3+ valori posibile, dar nu foarte multe, am realizat encodarea lor folosit OneHotEncoding. Coloanele care au fost astfel transformate sunt „combustibil”, „transmisie”, „caroserie”, „culoare” și „opiuni_culoare”

Coloanele categorice care au doar 2 valori posibile („cutie_de_viteze”) au fost encodate într-o singură coloană, cu 2 valori întregi (1 – „Automata”, 0 – „Manuala”)

c. Encodarea modelului

Pentru a realiza encodarea modelului mașinii, am scos „_” dintre marcă și model în sine. După aceea, am antrenat un model Word2Vec pe coloana „model” a setului de antrenare cu embeddings cu dim=50. Astfel, mașinile care au același model / marcă vor avea vectorii de embeddings mai apropiați decât restul mașinilor și se pot face asocieri mai bune între prețuri și model / marcă.

d. Encodarea addon-urilor

Întrucât numărul de addon-uri existente în setul de antrenare este unul relativ redus (183), am creat câte o coloană pentru fiecare addon posibil care are valoarea 1 sau 0, în funcție de faptul dacă mașina are sau nu addon-ul respectiv.

2. Antrenarea modelului

Întrucât acest task este unul de predicție, iar reprezentarea pe care am ales-o a setului de date conține foarte multe coloane cu valori binare (mașina are sau nu are caracteristica respectivă), am ales să folosesc un model RandomForestRegressor, cu criteriul „squared_error”

și 200 de arbori de estimare ce pot avea o adâncime maximă de 321. Întrucât RandomForestRegressor este un model de ansamblu de arbori de decizie, acesta nu are nevoie ca datele de intrare să fie normalizate / standardizate, așa că am sărit peste acest pas.

Pentru testarea modelului, 10% din setul de antrenare a fost rezervat pentru testarea acestuia, iar pe acesta s-a obținut un scor R2 egal cu 0.957. După testare, întregul set din „train.json” a fost folosit pentru antrenare. Atât setul de antrenare, cât și cel de testare se găsesc în folderul „input”, iar în folderul „output” se găsește rezultatul predicției pe setul din fișierul de testare.