Celtrin programerski izziv: Mining FRIk

Peter Us

31. maj 2016

1 Uvod

Cilj naloge je bil napovedovanje časov nalaganja oglasov na mobilnih napravah. Učni podatki, ki so bili na voljo so zajemali 2.500.000 primerov ter 46 atributov. Podatke sem najprej preprocesiral, nato je sledilo učenje in analiza atributov. Za učenje je bila uporabljena knjižnica xgboost, saj "extreme gradient boosting" algoritmi v zadnjem času (poleg nevronskih mrež) prevladujejo v tekmovanjih odkrivanja znanj iz podatkov. Poleg tega lahko iz modela ugotovimo, kateri atributi so najbolj prispevali k odločitvi za napoved - kar je zaželjena lastnost, saj nas v tem tekmovanju ne zanima samo najboljša možna napoved, ampak tudi kateri atributi so k njej najbolj prispevali.

Za implementacijo je bil uporabljen programski jezik Python 2.7. Pomagal sem si s knjižnicami "scikit-learn" in "xgboost". Sledi opis posameznih faz.

2 Predprocesiranje

Ker je naknadno prišlo do spremembe testne množice (odstranjeni so bili osamelci) je bilo to narejeno tudi na učni množici. Odstranjenih je bilo približno 5% učnih primerov z najvišjimi vrednostmi odvisne spremenljivke. Podatki so bili tudi premešani, saj so bili podani v urejenem vrstnem redu, česar si ponavadi ne želimo.

2.1 Predprocesiranje atributov

Atributi z veliko mankajočimi vrednostmi (EXTERNALCREATIVEID, NETWORKTYPE, EXTERNALSUPPLIERID) Ti trije atributi so bili odstranjeni, ker imajo v učni množici preko 2 milijona mankajočih vrednosti.

Binarni atribut (UA_MOBILEDEVICE) Je bil edini binarni atribut v podatkih. Zavzemal je lahko vrednost 1 ali "null". V predprocesiranju je bila vrednost "null" zamenjana z 0.

Numerični atributi (TOPMOSTREACHABLEWINDOWHEIGHT, TOPMOSTREACHABLEWINDOWWIDTH, HOSTWINDOWHEIGHT, HOSTWINDOWWIDTH) Ti atributi so ostali enaki kot so bili, le mankajoče vrednosti so bile zamenjane z najbolj pogosto vrednostjo vsakega atributa.

Timestamp Iz timestamp atributa so bili ustvarjeni naslednji atributi: timestamp, timestamp_dayofmonth, timestamp_dayofweek, timestamp_hour, timestamp_minute, timestamp_second, timestamp_milisecond, timestamp_alteranivestamp.

Geo koordinate (GEO_LAT, GEO_LNG) Iz atributov koordinat so bili ustvarjeni naslednji atributi: GEOIP_LNG, GEOIP_LAT, GEOIP_SUM (vsota lat, lng), GEOIP_PROD (produkt lat, lng) , GEOIP_DIFF (razlika lat, lng).

JSON (FILESJSON, ERRORSJSON) Iz file JSON atributa sta bila narejena atributa "size" in "length". Prvi predstavlja skupno velikost datotek, drugi pa število le-teh. Iz error JSON atributa je bil narejen "length" atribut, ki predstavlja število napak v ERRORJSON atributu.

Missing values Narejen je bil nov atribut, ki predstavlja število mankajočih vrednosti v učnem primeru.

Kategorični atributi (vsi ostali) Vsi kategorični atributi so bili binarizirani po sistemu "one-hot-encoding", kjer za vsak atribut naredimo tolikšno število novih binarnih atributov, kolikor ima atribut možnih kategoričnih vrednosti (oz. enega manj zaradi problema nedoločljivosti). Nato postavimo enico tistemu, ki predstavlja vrednost atributa v danem primeru, ostali pa imajo vrednost 0. Za kategorične atribute z velikim številom diskretnih vrednosti sem vzel samo kategorije, ki vsebujejo vsaj 100000 primerov. Tu se z mankajočimi vrednostmi nisem posebej ukvarjal. Če je imel atribut manj kot 100000 mankajočih vrednosti se le-te niso upoštevale. Če pa jih je imel več, sem za njih naredil svoj atribut.

Na ta način dobimo 835 atributov. Zaradi uporabe "one-hot-encoding" metode ima velika večina atributov vrednost 0. Zato so bili podatki v nadaljevanju uporabljeni v "sparse matrix" formatu, kjer shranimo samo pozicije in vrednosti elementov matrike, kjer vrednost ni enaka 0. Na tak način celotna učna množica zavzame približno 5GB pomnilnika. To pomeni, da bi lahko enak pristop uporabili tudi na večjih podatkovnih zbirkah, brez potrebe po ogromnih količinah pomnilnika.

3 Učenje

Kot že omenjeno, je bil za učenje uporabljen "eXtreme Gradient Boosting", implementiran v knjižnici "xgboost". Slabost xgboost modela je ta, da ima model veliko število parametrov. Iskanje primernih parametrov in učenje je bilo narejeno na sledeči način:

1. Najprej izmed vseh podatkov vzamemo 300000 za učno in 300000 za testno množico (zaradi hitrosti ne uporabljamo vseh podatkov) - testna množica je potrebna za "early stopping"

2. S privzetimi parametri in pomočjo "early stopping" metode poiščemo primerno število uporabljenih dreves (parameter: "n_estimators")

3. S pomočjo cross-validacije za izbrano število dreves poiščemo primerne parametre dreves (parameteri: “max_depth”, “min_child_weight”, “gamma”, “subsample”, “colsample_bytree”).

4. S pomočjo cross-validacije za izbrane zgornje parametre poiščemo primerni stopnji L1 in L2 regularizacije.

5. Zmanjšamo learning rate modela. Uporabimo model z zgoraj izbranimi parametri na vseh podatkih (le da znova poiščemo primerno število dreves).

Za cross-validacijo je bila uporabljena metoda RandomizedSearchCV iz “scikit-learn” knjižnice, ki išče primerne parametre po podanem prostoru parametrov naključno. Tako cel postopek cross-validacije traja le nekaj 10 minut.

Čas učenja na celotni množici je odvisen od števila izbranih dreves. Na lastnem računalniku učenje na celotni množici z 1000 drevesi traja približno 40 minut.

4 Analiza atributov

Kot že omenjeno, je prednost izbire modela xgboost tudi ta, da nam omogoča analizo atributov. Spodaj so predstavljene relativne vrednosti “pomembnosti” atributov v naučenem xgboost modelu:

feature name	score
TOPMOSTREACHABLEWINDOWHEIGHT	0.0704352463388
TIMESTAMP	0.0579234524347
timestamp_second	0.0431417593086
timestamp_minute	0.0387387017817
timestamp_alteranivestamp	0.0376311003542
GEOIP_LNG	0.0351150674816
GEOIP_LAT	0.0332690651024
TOPMOSTREACHABLEWINDOWWIDTH	0.0325169900589
GEOIP_SUM	0.0317785891072
timestamp_hour	0.0292899044181
GEOIP_PROD	0.0285651775581
HOSTWINDOWHEIGHT	0.0270336792879
GEOIP_DIFF	0.026254256061
FILESJSON_size	0.0189796392775

Tabela 1: Normalizirane f vrednosti atributov.

Pomembno je omeniti, da zgornji rezultati morda ne kažejo čisto prave slike. Atribute, ki so bili binarizirani, sem namreč zamenjal z več atributi. Za pravo predstavo bi toraj morali sešteti f-vrednosti vseh atributov, ki izhajajo iz istega atributa. Če to storimo, so rezultati sledeči:

feature name	score
TIMESTAMP	0.211131257606
GEOIP	0.19498215531
TOPMOSTREACHABLEWINDOWHEIGHT	0.0704352463388
PLATFORMVERSION	0.0396138436504
ACCOUNTID	0.0337613324035
UABROWSERVERSION	0.0329135387182
TOPMOSTREACHABLEWINDOWWIDTH	0.0325169900589
FILESJSON	0.0302060685619
HOSTWINDOWHEIGHT	0.0270336792879
CDNNAME	0.0213315830496
UAVENDOR	0.0205931820979
CAMPAIGNID	0.0188292242688
UAPLATFORMVERSION	0.0173797705487
EXTERNALPLACEMENTID	0.0166687177804

Tabela 2: Normalizirane seštete f vrednosti skupin atributov, kjer ime predstavlja vse izpeljanke iz atributa, omenjene v poglavju Procesiranje.

5 Rezultat

Na leaderboardu je bil dosežen score: 2.9760241295. Postopek predprocesiranja in treniranja modela je bil identičen opisanemu v poglavjih "Predprocesiranje" in "Učenje".

6 Ostalo

Še nekaj omembe vrednih stvari, ki se niso obnesle:

- Napovedovanje za vsak dan posebej (14 modelov) se je izkazalo za slabše.
- Uporaba nevronskih mrež (Keras, Theano) - z dostopom do strežnika s primerno opremo (grafične kartice) bi se dalo priti do dobrih rezultatov. Na lastnem računalniku se obnese slabše in počasneje od xgboost.
- Stohastični gradientni spust (linear SVM) in random forest metode. Se obnesejo slabše od nevronskih mrež/xgboost.

7 Možne izboljšave

- V primeru vsaj še enega modela, ki bi dajal podobono dobre rezultate kot xgboost, bi rezultate lahko izboljšali z uporabo stackinga modelov.
- Ker se Geo_lat, Geo_lng in timestamp atributi izkažejo kot najbolši, bi bilo vredno poskusiti dodati še kakšne atribute, ki so izpeljani iz njih.