A mesterséges intelligencia térhódítása: Gépi tanulás

Algoritmusuk illesztése, legjobb keresése egy osztályozási problémára, bevezetés a machine learning világába

István Fehér
Eszterházy Károly Egyetem
Eger, Hungary
stevenwhite997@gmail.com

***Abstract*—Életünk telis-tele van fontos döntésekkel, ezek közül az egyik legfontosabb, hogy mibe fektetjük kemény munkánk árán megszerzett javainkant, pénzünket. Minden vásárlás egy befektetés is egyben, éppen ezért sokan azt kívánjuk bárcsak előre láthatnánk egy-egy esetleges befektetés/vásárlás előtt, hogy a jövőben örömünket leljük-e benne, elégedettek leszünk-e vele. Ezt a témát tárgyalja majd konkrét példán keresztűl ez a tanulmány. A mesterséges intelligencia hatalmas iramban fejlődik napjainkban, ez új sohasem látott lehetőségek tárházát nyítja meg nekünk, még az előbb felvetett problémára is van megoldása. Egy a problémánkra illeszkedő adathalmazt fogunk megvizsgálni, mely kíváló betekintést ad mind az autóvásárlás kritériumai és az utánna bekövetkező elégedettségünk mértékének megállapítására, mind a gépi tanulás és algoritmusok összehasonlítására.**

***Index Terms*—Adathalmaz vizsgálata, Kereszt validálás, Algoritmusok tesztelése, Legjobb kiválasztása, Ellenőrzés**

I. Bevezetés

A fent olvasott összefogalaló után térjünk át egy valós példára, legyen ez most az autóvásárlás. A gépi tanulásnak hála, mégha ez kissé furán is hangzik, akár szinte előre megnézhetjük az elégedettségünket egy vásárolni kívánt autóval kapcsolatban. Rengeteg szempont alapján választunk magunknak autót, a vizsgált adathalmaz azonban 6 meghatározó paraméter alapján minden autó-t egy elégedettégi osztályba sorol. Ez a 6 paraméter pedig a következő: vásárlási ár, fenntartási költségek, ajtók száma, férőhelyek száma, csomagtartó mérete és az jármű biztonsága. Ezek alapján vontak le összefüggéseket az adathalmaz készítői, és derítették ki, hogy a járművek tulajdonosai mennyire elégedettek az autójukkal. Továbbiakban nem is szeretném tovább taglalni milyen fontos tehát, hogy jó/ magunknak megfelelő autót válasszunk. A célkítűzésem, pedig nem más, mint megtalálni a legjobb algoritmust ami a legnagyobb pontossággal (minimum 95 %!) be tudja sorolni az alább említett paraméterek alapján a gépjárművet az adathalmazból betanult módon. Emelett a hiba lehetőségére is figyelmet fordítok, minimalizálom azt. Rengeteg tanulmánytól eltérően mi itt most nem egy adott, meghatározott algoritmussal szeretnénk eredményt elérni, és azt fejlesztgetni, tovább csiszolni (boostolni), hanem a csoportosítási/besorolási problémához keresünk számos, ismert algoritmust, és ezeket ”versenyeztetjük”, ha úgy tetszik nem kell algoritmus szakértőnek lennünk, mert ez a tanulány érintőlegesen mindent megmagyaráz, tehát nem ragaszkodunk egyetlen algoritmushoz sem, egyszerűen a legjobbat választjuk, vagy akár többet is, ha esetleg több is megfelel majd elvárásainknak.

II. Módszerek

A fent leírtakban szereplő adathalmaz megtalálható az UCI Machine learning weboldalon, Car Evaluation Data Set néven. Az 1997-es évben került megosztásra, a még mindig naprakész infomáció, ami hatalmas népszerűségnek örvend, az egyik leggyakrabban elemzett adathalmaz! Méreteit tekintve nem mondható sem túlzottan egyszerűnek, sem bonyolultnak, 1728 sort tartalmaz, 7 attribútummal rendelkezik, egy osztályba soroló adatszerkezet, tökéletes alany a gépi tanuló algoritmusok összehasonlítására, a mesterséges intelligencia kibontakozására!

Eredetileg az adatunk az alábbi módon tündökölt:

```
vhigh,vhigh,2,2,small,low,unacc
vhigh,vhigh,2,2,small,med,unacc
vhigh,vhigh,2,2,small,high,unacc
vhigh,vhigh,2,2,med,low,unacc
vhigh,vhigh,2,2,med,med,unacc
vhigh,vhigh,2,2,med,high,unacc
vhigh,vhigh,2,2,big,low,unacc
vhigh,vhigh,2,2,big,med,unacc
vhigh,vhigh,2,2,big,high,unacc
vhigh,vhigh,2,4,small,low,unacc
vhigh,vhigh,2,4,small,med,unacc
vhigh,vhigh,2,4,small,high,unacc
```

Fig. 1. Az adat eredeti szerkezete.

Nos ez egy laikusnak nagyon szimpatikusnak tűnhet, hiszen érhetőnek tűnik (amennyiben tudjuk melyik paraméter melyik oszlopnak felel meg) és nem csak rengeteg numerikus értékkel találja szemben magát. Valóban vonzó ez a szerkezet, de sajnos a gépi tanulással nem kompatibilis, mivel a gépi tanulás még mindig sokkal gyorsabb, jobban tanul számokból, mintsem szöveg/String értékekből.

Ennek következményeként az adatokat át kell konvertánunk egységes numerikus értékekre. Az alábbi módón rendeltem össze az értékeket:

a Dataframe replace parancs használatával.
pl : (df = df.replace(’unacc’, 1)).

unacc	1
acc	2
good	3
vgood	4
low	1
med	2
high	3
vhigh	4
more	5
small	1
5more	6
med	2

Fig. 2. Összerendelési táblázat.

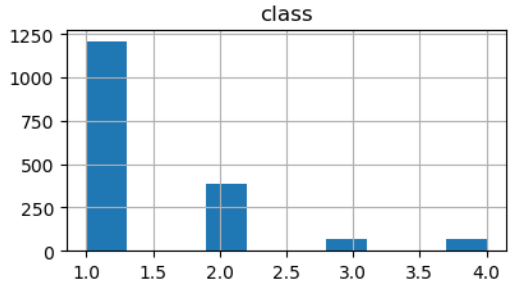

Fig. 4. Adatok eloszlása a cél csoportokban.

Továbbá fejléceket adtam az adathalmazhoz a könyebb azonosíthatóság végett ,az eredeti adathalmaz fejlécek nélküli volt.

Ekkor az adatszerkezetünk a következőképpen nézett ki:

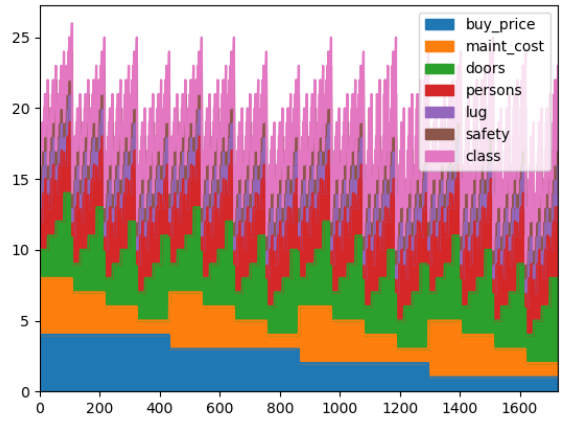

Fig. 3. Az adathalmaz eredeti szerkezete.

Nos ebből az ábrából aligha vonhatunk le messzire menő következtetésket, de érdemes tudnunk, hogy ez volt a kiindulási pontunk.

Mindenképpen megemlítendő dolog még, a python, pycharm és a fontosabb könyvtárak verzóinak ismerete (számos algoritmus, keresztvalidálási módszer) meghívása különböző képpen történik a különböző verziókban! Ezért ezeket a verziókat itt közlöm:

A. Verziószámok

- Python: Python 3.6.6 (v3.6.6:4cf1f54eb7, Jun 27 2018, 03:37:03
- PyCharm: JetBrains PyCharm Community Edition 2018.3.4 x64
- Scipy: 1.2.1
- Numpy: 1.16.3
- matplotlib: 3.0.3
- pandas: 0.24.2

B. Adatok csoportokba rendezése

Az adataink eloszlása kezdetekben az alábbi módon szemléltethető:

Mint az jól látható a legtöbb autót elfogadhatatlannak (1 = unacceptable) nak tartják.

Az osztályba/csoportba rendezés problémáját többféle irányból is megközelítettem, először a Kmeans-t hívtam segítségűl és ezzel próbáltam meg klaszterezni és ezáltal csoportokba szervezni az adatokat. Az adatokat így csoportokba szerveztem. Ezután DBScan-t használtam, amivel az alábbi adatokat kaptam:

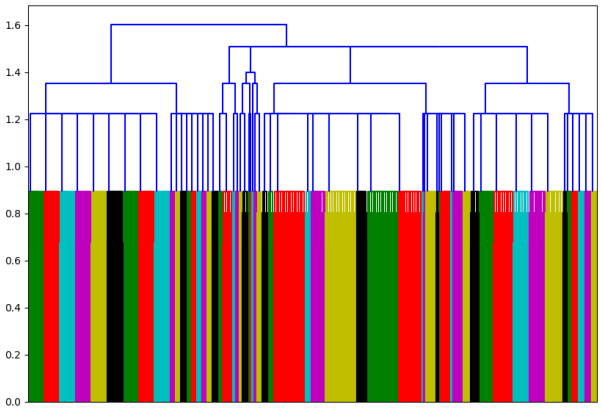

Fig. 5. DBSCAN

Ezután a keresztvalidálási módszerrel kezdtem el az adathalmazra illeszkedő legjobb algoritmus felkutatását.

Ennél a módszernél az adatokat, azaz a 7 paramétert 2 csoportra avagy tömbre osztottam. Az egyik X tömb tartalmazta az első 6 attribútumot, az Y tömb pedig csak az osztályba tartozást tartalmazza. Ezt a 2 tömböt is további résztömbökre oszotottuk, egy finomítani kívánt (train) és egy ellenőrző (validation) résztömbökre.

Ezután a scikit learn model selection, traint test split metódusát hívjuk meg erre a 4 paraméterre, mégpedig 7-es seed értékkel, és az arányokat a 80/20 nál határozzuk meg az adat 4/5-ét finomítjuk a maradék 20% -ot pedig ellenőrző összegnek visszatartjuk.

A pontosság megállapítására k-fold cross validation -t használtunk, ez annyit jelent, hogy az adatainkat 10 részre osztja fel, majd ebből 9 en végez számításokat azaz ”train”-el, a maradék 1 részt visszatartja ellenörzés ”validálás” céljából,

aztán ezeket ismételgeti amíg minden kombinációba nem került.

Említésre méltó továbbá, hogy többféle scoringot is használhatunk.

C. Algoritmusok

Ezt követően algoritmusok legkülönbözőbb, legszínesebb kavalkádját kutattuk fel, amelyek kompatibilisek voltak a mi problémánkkal, célunkkal. Mindegyiket a már fent említett 10-fold cross validation el használtam, a megbízható eredmény/pontosság kedvéért.

Ezek a következők:

1) Logisztikus regresszió: Ami megbecsüli egy bizonyos esemény (a függő változó) bekövetkezésének valószínűségét. A logisztikus regressziót leggyakrabban egy esemény előrejelzésére használják.

2) Lineáris diszkriminancia-analízis: Lineáris diszkriminancia-analízis a statisztikában, minta-felismerésben és gépi tanulásban használt módszer, amely a független változók olyan lineáris kombinációját képes megtalálni, amely a függő változó alapján kialakított csoportokat a lehető legjobban megkülönbözteti (diszkriminálja). A diszkriminancia-analízis szorosan kapcsolódik a varianciaanalízishez és a regresszióanalízishez, amelyek úgyszintén egy függő változót igyekeznek kifejezni más változók lineáris kombinációjaként. A logisztikus regresszió közeli ”rokona”. Ez az algoritmus is független változóban folytonos, függő változóban kategórikus. Ez az amilyen típusú algoritmusokra nekünk szükségünk van!

3) Gaussian Naive Bayes: Nagyon hasonló az előző algoritmusokhoz, de abban mégis eltér, hogy minden paramétert elkülönítve, külön értékel ki, egymástól teljesen függetlenül. Minden változót ugyanolyan mértékben végkimenetel befolyásolónak tart.

4) K-nearest neighbors: Most kicsit más irányból közelítjük meg a megoldásunkat. Ez az algoritmus egy nem paraméteres minta felismerő algoritmus, amit főként regresszió-ra és osztályozsára használhatunk. A legegyszerűbb, kezdő machine learning algoritmusnak tartják. A szomszédos elemek közelsége alapján dönti el melyik csoportba fog tartozni egy vizsgált objektum.

5) Support Vector Machine: Egy jelentős figyelmet kiváltó osztályozási módszer a tartóvektor-gép (SVM – Support Vector Machine). A módszer a statisztikai tanulás elméletéből származik és ígéretes tapasztalati eredményeket mutat sok gyakorlati alkalmazásban. A módszer egy másik egyedi jellege, hogy a döntési határt a tanulóesetek egy részhalmazának segítségével reprezentálja, amelyeket tartóvektoroknak (support vector) nevezünk.

6) RandomForestClassifier: A döntési fára épül, döntési fák sokasága. Az egyes attribútumok értékei alapján a mintákat hierarchikusan csoportosítjuk. A levelek: osztálycímkék.

7) Decision Tree Classifier: A döntési fa (decision tree) is igen nagy népszerűségnek örvend a mesterséges intelligencia és a gépi tanulás terén. Működéséről elég ha most annyit tudunk, hogy a célja egy olyan modell létrehozása, amely minnél nagyobb pontossággal tudja megbecsüli a célváltozó értékét (class) a bemeneti változó függvényében. A fa minden levél eleme egy bemeneti paraméternek felel meg.

III. Eredmények

Az előbb felsorolt 7 algoritmust összevetve az alábbi pontosságokat kaptam (10-fold validation) -al:

Egy fontos tényező van még azonban a cross validation és az algoritmusokon kívűl is, ez peidg a scoring változó a cross validation-ben. Általában az a mondás tartja, hogy kiegyensúlyozott (balanced) adathalmazokon elég a ”sima” accuracy-t használni a scoring értékeként, esetünkben viszont sajnos az adathalmaz a legkevésbé sem kiegyensúlyozott, azaz a különböző osztályokba sorolható példányok száma jelentősen eltér. Vizsgáljuk meg tehát mind a 2 scoring móddal az algorimusaink eredményét! Legyünk naivak és feltételezzük, hogy az adathalmazunk kiegyensúlyozott. Ekkor a accuracy scoringgal az eredményeink:

A. Accuracy Scoring

```
Algorithm comparison:
LR: 0.794526 (0.024051)
LDA: 0.824179 (0.025851)
NB: 0.767016 (0.029916)
KNN: 0.930529 (0.023429)
SVM: 0.948629 (0.017819)
RFC: 0.955135 (0.012047)
DTC: 0.976113 (0.011718)
```

Fig. 6. Algoritmusok pontossága.

Az első érték jelenti az átlagos pontosságot, a második az eltérést.

Szemléltetve ez az alábbi módón fest:

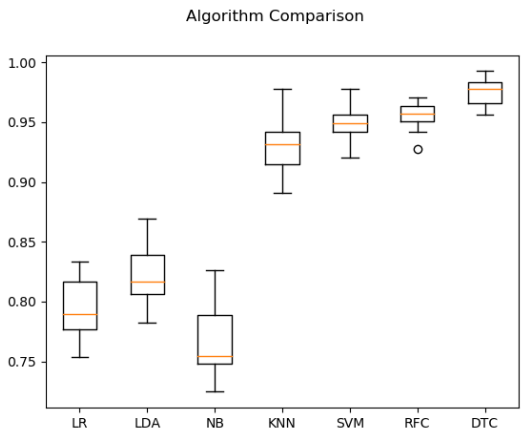

Fig. 7. Boxplot ábrázolás

De vajon a scoring mekkora szerepet játszott ebben? Most megtudhatjuk! Balanced Accuracy Scoring:

```
Algorithm comparison:
LR: 0.431882 (0.057562)
LDA: 0.518571 (0.052870)
NB: 0.625922 (0.041293)
KNN: 0.764210 (0.086830)
SVM: 0.868757 (0.082901)
RFC: 0.883599 (0.086307)
DTC: 0.963345 (0.035630)
```

Fig. 8. Algoritmusok pontossága.

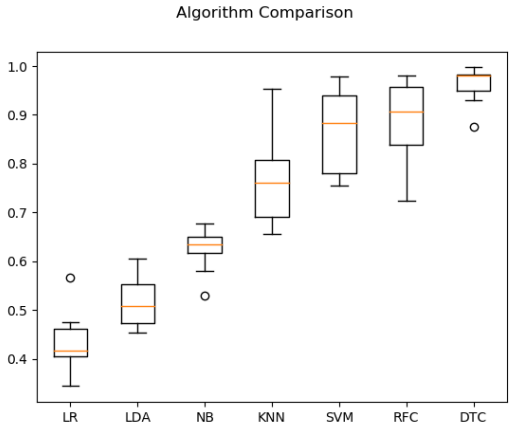

Fig. 9. Boxplot diagramm.

Mindenképpen fontos tényező, hogy most csak minimálisan voltak felparaméterezve függvényeink! Na de mi produkálhatott ekkora eltérést a scoringokban? A következő fejezetben megtudhatjuk.

Most pedig leellenőrizzük a befutó (DTC) algoritmusunk, a predict függvénnyét lefuttatva a validációs tömbünkön.

```
0.9884393063583815
[[244   1   1   0]
 [  1  76   0   0]
 [  0   1   7   0]
 [  0   0   0  15]]
              precision    recall  f1-score   support

           1       1.00      0.99      0.99       246
           2       0.97      0.99      0.98        77
           3       0.88      0.88      0.88         8
           4       1.00      1.00      1.00        15

    accuracy                           0.99       346
   macro avg       0.96      0.96      0.96       346
weighted avg       0.99      0.99      0.99       346
```

Fig. 10. Final Check-Predict-Validation

IV. Kiértékelés

Eredményeinket tekintve a Decision Tree Classifier (Döntési Fa Osztályozó) algoritmus volt az abszolút befutó, minden kiértékelési módszernél. El is érte a célkitűzésünket, a 95%-os pontosságot! Továbbá a második helyen végző Random Tree Classifier (Véletlen Erdő osztályozó) sem sokkal maradt el elvárásainktól. Általánosan elmondható tehát, hogy a mi problémánkra a döntési fa szerkezetek a legmegfelelőbbek a (Random Tree Classifier is döntési fákra épül). Most azonban tekintsünk vissza a scoring miatti jelentős eltérésre, hogyan volt lehetséges, hogy ekkora mértékben befolyásolta algoritmusaink pontosságát?

Nos, az adatszerkezetünk kiegyensúlyozottlanságának hiánya vezetett ide. Túlnyomó részt 1 osztályba tartoznak a példányok (az elfogadhatatlan, unacc csoportba).

Elfogadhattuk volna a accuracy scoring által kapott rendkívűl kecsegtetőnek tűnő pontosságokat, akkor majdnem 3 algoritmussal is elértük volna a kitűzött 95%-os pontosságot, de ez az adatszerkezetünk ismeretében teljes félrevezetés lett volna.

Örömmel látjuk viszont, hogy megvan a ”befutó”, keresett algoritmusonk. Érdekesség még továbbá, hogy ezt az eredményt mindenféle paraméterezés, testreszabás és boosting nélkül érte el, sőt a végső ellenörzés során még felül is múlta magát, mégjobb pontosságot mutatott a validációs tömbön!

Továbbá, egy nagyon érdekes végső következtetésre is jutottunk, mégpedig minden paraméter befolyásolja, természetesen, hogy egy példány melyik osztályba sorolható, és ezek a paraméterek összefüggésben is állnak egymással, viszont van 1 kiemelkedő szerepű bemeneti paraméter, ami még jelentősebben meghatározza, hogy melyik osztályba fog egy egyed tartozni, ez pedig nem más mint a safety, azaz az autó biztonsági szintje.

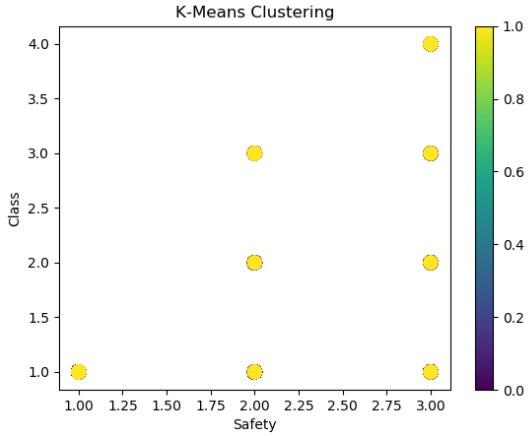

Fig. 11. Kmeans Clustering Visualization

Ahogyan azt az alábbi ábrán is láthatjuk, ha egy autó nem biztonságos akkor az semmilyen más paraméteri kombinációkkal sem lehet mégcsak elfogadható (acc) besorolású sem. Az ábrán egyébként egy 2 klaszteres K-means klaszterezés látható.

A célunkat (95%-os pontosságú becslés) tehát elértük a Decision Tree Classifier (Döntési Fa Osztályozó) algoritmussal és további fontos következtetésekre is jutottunk. A felhasznált, ismertetett módszereket és algoritmusokat bármilyen más adatszerkezetre is bátran felhasználhatjuk, működni fog, mindaddig amíg az egy osztályba rendezési/sorolási probléma. Más problémákhoz másmilyen eljárásokra van szükség.