Váradi Fruzsina
VD4JNP

Nagy mennyiségű adatfeldolgozás

Projektmunka - Dokumentáció

Az adatfeldogozás a *"data57.cvs"* adathalmaz alapján történt.

Az állomány egy híres kozmetikai márka 2014-es Facebook post adatait tartalmazza.

A projektmunka során az *Anaconda Navigator Jupyter Notebook* fejlesztői környezetet használtam.

Importált könyvtárak:

- Pandas
- Numpy
- Seaborn
- Pyplot (Matplotlib)
- Axes3D (mpl_toolkits.mplot3d)
- KMeans (sklearn.cluster)
- sklearn
- LinearRegression, LogisticRegression (sklearn.linear_model)
- train_test_split (sklearn.model_selection)
- preprocessing (sklearn)
- accuracy_score (sklearn.metrics)

A.) Adatvizualizáció és klaszterezés

A1.) Az adathalmaz megismerése, a benne szereplő attribútumok bemutatása és általános jellemzése

Beolvastam az adathalmazt és elmentettem egy *"df"* vátozóba.
Ahhoz hogy lássuk, hogy néz ki néhány tipikus sor, a *df.head* parancsot használtam.
Az attribútumok megismeréséhez az egyik legjobb eszköz az információs panel: *df.info()*.
Ebből kiolvashatjuk, hogy az adatbázis 19 oszlopot és 344 sort tartalmaz. A 19 oszlopból 2 float64(), 16 int64() és 1 object() típusú adatokat tartalmaz. Ez nekünk tökéletes, csupán egyetlen oszlop, a "Type" tartalmának típusa *object*, viszont ez nem fog gondot jelenteni, és ezen kívül csak numerikus adatok szerepelnek az adatbázisban.
Az információs panel tartalmazza oszloponként a "nem null" attrinútumok számát. Láthatjuk, hogy van néhány üres cella ("share" és "like" oszlopokban), ezért ezt is orvosolni fogom.
A *df=df.drop([244])* utasítással eltávolítom a 244. sort, mivel ez a sor egy extrém kiugró értéket tartalmaz.
Továbbá a *df.descibe()* parancsot használva néhány statisztikai leírót kaphatunk meg, melyekre esetleg szükségünk lehet. Innen megtudhatjuk az oszloponkénti átlagokat, minimumokat, maximumokat, kvartiliseket, stb.

Váradi Fruzsina
VD4JNP

A2.) Előfeldolgozás, adattisztítás, pl. hiányzó adatok vagy extrém értékek feltérképezése, illetve kezelése, adatkonverzió

Bizonyos oszlopot átneveztem annak érdekében, hogy egyszerűbb legyen a használatuk.:

- "Lifetime Post Total Reach" → "Post Nézettség"
- "Lifetime Post Total Impressions" → "Post Feltűnés"
- "Lifetime Engaged Users" → "Egyedi fálhasználók click"
- "Lifetime Post Consumers" → "Összes felhasználó click"
- "Lifetime Post Consumptions" → "Clickek"
- "Lifetime Post Impressions by people who have liked your Page" → "Post Feltűnés-követők"
- "Lifetime Post reach by people who like your Page" → "Post Nézettség-követők"
- "Lifetime People who have liked your Page and engaged with your post" → "Követők click"

Az oszlopok adattípusai numerikusak, adatkonverzióra nincs szükség.
df.isnull().sum() parancs segítségével feltérképeztem, hogy mely oszlopokban vannak hiányzó adatok. Látjuk, hogy a "like" (1) és "share" (4) oszlopokban.
Egy *mean* változóba eltároltam a "like" oszlop átlagát, majd a hiányzó cellákat feltöltöttem ezzel az értékkel. Ugyanezt megismételtem a "share" oszloppal is. Így a *df.isnull().sum().sum()* parancs 0 értékkel tér vissza.
Ezután a "Total Interactions" oszlop alapján (ez az oszlop tartalmazza a likeok, kommentek és megosztások összegét) csökkenő sorrendbe rendeztem a táblázatot, így láthatjuk, hogy melyek voltak a legnépszerűbb posztok. Majd, hogy egy összehasonlítást végezzek megnéztem az oszlop átlagát és a maximumát.
Kíváncsi voltam, hogyan alakul a "Total Interactions" oszlop áltaga, ha az adott poszt fizetett reklám volt, vagy sem ("Paid" oszlop 1 vagy 0). Egyértelművé vált, hogy a fizetett content-ek iránt magasabb az érdeklődés.
Végül categória és típus szerint végeztem egy összegző táblázatot.

A3.) Különböző vizualizációs eszközök használata az adathalmaz, illetve az attribútumok között fennálló kapcsolatok feltárására, két vagy több attribútum együttes vizsgálata alkalmas plotok felhasználásával és a kapott eredmények értelmezése

Az adathalmaz oszlopai között több releváns információt és összefüggést is találhatunk, ezért több vizualizációs eszközt is hasnzálni fogok.
Először megvizsgáltam a benyomásokat. Összehasonlítottam a " Post Feltűnés" és a " Post Feltűnés-követők" oszlopokat, hogy megtudjam van e hatása a követőknek a népszerűségre. A sűrűségfüggvények nem normális eloszlásúak, és a dobozábrák is ezt sugallják. Vannak kiugró értékek is.
Majd megnéztem a bejegyzések click számának eloszlását. A 2000 alatti értékek dominálnak. Egy átlagos bejegyzés 1289.9273255813953 ckick-et kapott, viszont kiugró értékeket itt is bőven találhatunk.
Kördiagram segítségével kirajzoltattam az eloszlást a bejegyzés típusa szerint, külön fizetett és nem fizetett posztok alapján. A fényképes bejegyzések dominálnak mindkét kategóriában.
Készítettem egy hisztogramot, mely az első 8 "legforgalmasabb" órákat tartalmazza, csökkenő sorrendben, kategóriákra bontva. Tehát ezekben az órákban posztoltak a legtöbbet.

Jól látható, hogy az 1-es kategóriából posztolnak a legtöbbet (különleges kínálatok és tartalom), de pl 4 órakor a 2-es kategória dominál (termék és márka reklámok).
Egy eloszlásfüggvényt használva megtudhatjuk a like-ok számának alakulásaát is, mejből kiderül, hogy 0 és 250 között van a legtöbb like.
A következő grafikon tartalmazza hónaponként a "Page total likes" (azoknak az embereknek a száma akik like-olták az oldalt) alakulását. Kivehető, hogy a hónapok előrehaladásával emelkedik a like-ok száma. Láthatunk lapultabb és csúcsosabb, illetve jobbra és balra elnyúló értékeket is. Szinte egyik sem simmeltrikus, tehát nem normális eloszlású.
A következő táblázat egy korrelációs táblázat, mely az értékek közötti függőségeket (vagy épp függetlenségeket) tartalmazza. *sns.heatmap(…)* parancs használatával szemléletesebbé tettem a táblázatot. Így könnyebben kivehetők azok a metszetek, amik függőséget mutatnak. Kétségtelenül azt mondhatjuk, hogy a "like" és "Total Interactions" oszlopok között determinisztikus/ függvényszerű lineáris kapcsolat van. Továbbá a következő párok között szoros a kapcsolat:

- "Követők click" (azoknak az embereknek a száma, akik like-olták az oldalt és click-eltek a poszton valahova) és "Összes felhasználó click" (az emberek száma, akik click-eltek a poszton valahova): 0,97
- "Post Nézettség-követők" (azoknak az embereknek a száma, akik láttak egy posztot azért, mert like-oltál az oldalt) és "Post Feltűnés-követők" (az oldalt like-olt emberek ráhatásainak a száma): 0,95
- "Post Month" és a "Page total likes": 0,94
- illetve a "comment", "like", "share" és "Total Interactions" oszlopok mindegyike között: 0,95 ; 0,93 ; 0,89 ; 0,88 ; 0,86.

Mivel ezeken a helyeken magas a korrelációs együttható értéke, ezekkel az attribútumokkal érdemes foglalkozni a későbbiekben.
A *sns.pairplot(df.loc[:,'comment':'Total Interactions'])* parancs használatával egy 4x4-es táblázatba kirajzolom az összefüggéseket. Mint azt már az előbbi korrelációs táblázatból is megtudhattuk, a "like" és "Total Interactions" oszlopok között lineáris a kapcsolat (nagyjából egy vonalban helyezkednek el a pontok).

A4.) Klaszterezésre használható algoritmus kiválasztása, bemutatása, illetve annak az adathalmazra történő alkalmazása, a kapott eredmények értelmezése (szükség esetén pl. a Scikit-learn dokumentáció is használható).

Klaszterezéshez én a *k-közép* módszert választottam. Ez a módszer megpróbálja megkeresni és idealizálni a felhasználó által megadott számú klasztert (k) és inden egyes ponthoz megpróbálja megtalálni azt a klasztert, aminek a középpontjához ő a legközelebb van, tehát távolságot mér (euklideszi távolság).
A "like" és "share" oszlopok pontfelhő diagramjából indultam ki.
Először 4-nek választottam a klaszterek számát: *kmeans = KMeans(n_clusters=4).* Kiírattam a centrálpontokat: *kmeans.cluster_centers_,* végül kiplotoltattam.
Nem voltam biztos abban, hogy jól választottam a klaszterek számát, ezért lefutattam a k-könyök módszer függvényt, melyből arra következtettem, hogy lehet jobb lett volna 5 klaszter. A beadott projektmunka az 5 klaszteres változatot tartalmazza.

Kíváncsiságból megnéztem a függvényeket 1-től 11-ig számú klaszterekkel, ellenőrzés képpen is.
Végül egy 3D-s vizualizációt is kipróbaltam, mivel a “comment” oszlop a “like ” és a “share” oszlopokkal is szoros kapcsolatban van.
A klaszterezés segítségével megjósolhatjuk egy poszt helyét. Például ha egy poszt 800 like-ot, 50 megosztást és 20 commentet kap, akkor az a második klaszterbe sorolódik:
kmeans.predict([[800,50,20]]) → array([2]).

B.) Regresszió

B1.) Az előző rész eredményeit felhasználva, a megfelelő attribútumok kiválasztása és azok indoklása
Az A.) alpont alapján tudjuk, hogy a “like” és a “Total Interactions” oszlopok között lineáris a kapcsolat. Ezért ezekre az attribútumokra fogunk lineáris regressziót alkalmazni. Így a folyotonos adatok alapján, ha ismerjük pl a “like” értékét, meg tudjuk mondani a “Total Interactions” értékét is.
A “like” és “Clickek” oszlopok plot diagramján azt látjuk, hogy eléggé szétszórtak a pontok, tehát nem függenek egymástól. Ezért ezekre az attribútumokra fogunk logisztikai regressziót alkalmazni.

B2.) Lineáris regresszió alkalmazása folytonos attribútum esetén, az eredmények értelmezése, illetve azok felhasználása
Átadtam az X értékének a “like” oszlop adatait, és az y-nak a “Total Interactions” oszlopét.
Majd *a reg = LinearRegression()* és *reg.fit(X,y)* parancssorokkal ráillesztettem a lineáris függvényt. A kapott eredményt (ami egy a pontok helyzeteit követő vonal kell legyen) ráteszem a plot diagramra, így leellenőrzöm az eredményt.
Végül a *sns.regplot(X,y)*-t használva a szórást is feltüntettem, ez a mi esetünkben igen kicsinek mondható.
A regressziót úgy használatjuk, hogy megadunk egy új X értéket és a *reg.predict(Xnew)* segítségével megkapjuk a várható y értéket.

B3.) Logisztikus regresszió alkalmazása diszkrét attribútum esetén, az eredmények értelmezése, illetve azok felhasználása
Hosszas próbálgatások után még mindig nem sikerült egy olyan diszkrét osztályt találnom az adatbázisban, ami megfelelne az attribútumok különválasztására, ezért létrehoztam egy új “Popular” oszlopot. Ha az adott Facebook bejegyzés ”Total Interactions” értéke nagyobb mint 450, vagy a “Clickek” értéke nagyobb mint 2000 a “Popular” értéke 1, különben 0. Tehát 1-esek lesznek a népszerű oszlopok. Ez alapján fogom kategorizálni a pontokat.
Átadtam az X értékének a “like” és “Összes felhasználó click” oszlopok adatait, és az y-nak a “Popular” oszlopét.
Majd *a reg = LogisticRegression()* és *reg.fit(X,y)* parancssorokkal ráillesztettem a logisztikus regressziós függvényt.

A kiplotolt eredményen jól látható a határvonal a népszerű és a kevésbé népszerű bejegyzések között. Természetesen vannak átfedések is, mivel ez egy egyenes.
Megjósolhatjuk, hogy melyik kategóriába fog esni egy olyan bejegyzés, melynek 50 like-ja és 200 click-je volt (a kevésbé népszerűbe).

C.) Osztályozási módszerek, gépi tanulás

C1.) Az előző részek eredményeit felhasználva, a megfelelő attribútumok kiválasztása és azok indoklása

Az y-nak a "Category" oszlopot választottam, mert ezek diszkrét attribútumok. Ennek a megjóslása lesz a cél.

Az egyik modell x értékei az adatbázis numerikus attribútumai lesznek. Ezek alapján fogjuk meghatározni az y változókat.

Egy másik modell x értékeit a korrelációs táblázat alapján választottam ki. Azok az oszlopok, melyek jobban meghatározzák a "Category"-t.

Létrehozunk egy új oszlopot, melyek az y értékek alapján lettek meghatározva:

- 1 → action
- 2 → product
- 3 → inspiration

Majd definiáljuk a változókat.

C2.) Előfeldolgozás, pl. a kiválasztásra került attribútumok értékeinek normalizálása

Megvizsgáltam az X értékeit dobozdiagramok segítségével és eszrevesszük, hogy jelentős eltérések vannak a tartományok között. Mivel a kiugró értékeket nehezebb egymáshoz hasonlítani, ezért az eltéréseket közelebb kell hozni egymáshoz. Erre a standardizálást használtam. Ennek során az eredeti várható értékeket 0-vá tesszük és átskálázzuk a szórás értékét úgy, hogy standard normalizáljuk az eredményt (egyeseket kapjunk).

Átskáláztam az X tanulóhalmaz értékeit: *scaler=preprocessing.StandardScaler()*

scaler.fit(X_train).

Lekérdeztem a szórásokat és a skálázási faktorokat (*scaler.mean_, scaler.scale_).* Mivel nagyok, megfelelnek a standardizálásnak.

Normalizáltam az X értékeket: *X_train_normalized=scaler.transform(X_train).*
Ellenőrzésképp lekérdeztem a szórásokat, és látom hogy 1-esek.

Ismét dobozdiagram segítségével kirajzoltam az X értékeit és beláthatjuk, hogy már "szebbek", mert mindegyik 0 várható érték körül van és nagyjából megfelelő szórással próbál értékeket hozzárendelni.

Ugyanezeket a lépéseket végeztem el az X teszthalmazra is hogy megkapjuk a normalizált X teszthalmaz értékeit.

C3.) A tanuló algoritmus használatához az adathalmaz tanuló és teszthalmazra történő felbontása, a választott paraméterértékek indoklása

Felbontottam az X és Y2 ertékeket tanuló- és teszthalmazokra (*X_train,X_test,Y2_train,Y2_test=train_test_split(X,Y2,train_size=0.8,stratify=Y2)*), majd logisztikai regressziókat illesztettem rájuk.

Majd lekérdeztem a prediktált értékeket a tesztelemekre.

Ezeket a lépéseket elvégeztem a szűrt adattáblámon is.

C4.) Az osztályozásra alkalmazott modell(ek) rövid bemutatása, az adathalmazon elvégzett tanítás eredménye, a kapott paraméterértékek bemutatása, értelmezése

C5.) Kiértékelés, különböző teljesítménymértékek használata

Végezetül lekérdeztem a modellek tesztjeinek pontosságát, pl:
accuracy_score(Y2_test,predicted)

Az első modellt a teljes adathalmazra nézve vizsgáltam, hogy miként határozzák meg az egyes címkéket. A tesztelésnél 54%-os, a tanulásnál pedig 32%-os pontosságot kaptunk, ami a tanulás szempontjából nem elég kielégítő.

A második modellt az eredeti értékek normalizált értékeire vizsgáltam. A tesztelésnél és a tanulásnál is 38%-os pontosságot kaptunk, ami egyik szempontjából sem elég kielégítő.

Végül a harmadik modellt a célváltozóval legjobban korreláló értékekre futtattam. A tesztelésnél 54%-os, a tanulásnál pedig 55%-os pontosságot kaptunk, ami mindkét szempontból kielégítő.

Ha összességében választanunk kell, akkor a 3. modellt választjuk, mert az a legpontosabb tanulás és tesztelés szempontjából is.